Meeting Minutes 6 October 2000

Author:
Warren Hedley (Bioengineering Institute, University of Auckland)
Contributors:
David Bullivant (Bioengineering Institute, University of Auckland)
Scott Lett (Physiome Sciences Inc.)

1 Summary

The lack of a decent method for obtaining the value of state variables at a given time (as described in Section 6 of the 4 October progress report) led to some discussion about a basic paradigm shift in the classification of variables as far as documentation and MathML are concerned. The units system proposed in the "Dave and Warren's CellML Brainstorm" document of 30 May 2000 (currently not archived on the cellml.org[1]) has also been reviewed and included in this document.

2 CellML Variables

First of all, we want to clear up some basic issues regarding variables:

- Variables should not be classified in their own declarations, which could lead to inconsistencies between their declarations and their use in the math. Intelligent software should be able to tell what a variable represents.
- As a result, variables like `time`, `x`, `y` and `z` should not be treated any differently from other variables. They must be declared in every component where they are referenced, and should be mapped to a single variable declaration with a **`public_interface`** attribute value of `out` somewhere.
- Every variable can be thought of as a function of all of the other variables declared in the current component.

It is the last point that gives us an optimal solution for evaluating variables at certain points in time and space, or indeed with respect to any variable declared in the current component.In general a variable is used by inserting its name into a **`<ci>`** element. To use it as a function, we treat it as a recognised symbolic operator, by placing its name into a **`<csymbol>`** element, with other variables as arguments. An example is given in Figure 1.

3 Units

The units information that is required can be split up into single SI base or derived units, each of which contains three bits of information. The first is a scale factor (such as *micro*, or *milli*) which may be indicated by a single letter abbreviation, or a number representing the power of 10 that the unit is to be multiplied by. The second is the type of SI unit, which can be indicated by a single letter or pair of letters. The third is the exponent (or order) of the units. Each of these tuples can be placed in square brackets, with commas separating the terms, and a complete unit specification made up from the product of several tuples. Some examples are shown in Table 1.

For conciseness, the parts of the tuple could have default values of `0`, `dimensionless` and `1` respectively. So the example in the first row of Table 1 could be rewritten as `[,K,]`. However a more

[1] http://www.cellml.org

```
<!--
    The first expression should return the value of Ca at time t_o,
    as needed in the Luo-Rudy II model. Note that the equality is not
    actually evaluated.
  -->
  <apply>
    <csymbol> Ca </csymbol>
    <apply><eq />
      <ci> time </ci>
      <ci> t_o </ci>
    </apply>
  </apply>
```

FIGURE 1: The proposed standard way of accessing variables at past time steps.

XML	Text description
[0,K,1]	degrees Kelvin
[m,V,1]	millivolts
[n,mol,1][m,m,-3]	nano-moles per millimeter cubed (concentration)
[-3,m,1][-3,s,-1]	millimeters per millisecond (permeability)

TABLE 1: Some examples of units in the proposed CellML format.

useful method of abbreviation has been proposed, where a single set of unit macros is defined for an entire **<model>**, and these macros can be used in variable declarations throughout the model, as shown in Figure 2. Obviously, various standard sets of unit abbreviations might be stored in a database (or even defined in the CellML specification), and then referenced in the CellML code, instead of being hardcoded.

```
<model>
  <units_abbreviation_table>
    <units abbreviation="dimless" expanded="[,,]" />
    <units abbreviation="concntn" expanded="[n,mol,1][m,m,-3]" />
  </units_abbreviation_table>

  <component name="myoplasm">
    <variable
        name="pi"
        units="dimless"
        value="3.14159" />
    <variable
        name="Ca_i"
        units="concntn"
        display_name="intracellular calcium concentration" />
    <variable
        name="vol_myo"
        units="[m,m,3]"
        display_name="myoplasmic volume" />
  </component>
</model>
```

FIGURE 2: An example of the definition and subsequent use of a set of unit abbreviations.

The **value** attribute (used on the declaration of the `pi` variable in Figure 2) is the final addition that we made, and it gives the modeller a chance to set the value of variables in the model definition, rather than in an external experiment configuration file. The modeller could use this attribute to suggest values, or to indicate the values that were used to achieve a certain result. The idea is that these values would not be edited by the user but would be overridden using configuration files when performing simulations.