Meeting Minutes 10 January 2001

Review of Units Specification

Author:
Warren Hedley (Bioengineering Institute, University of Auckland)
Contributors:
David Bullivant (Bioengineering Institute, University of Auckland)
Melanie Nelson (Physiome Sciences Inc.)
Poul Nielsen (Bioengineering Institute, University of Auckland)

1 Introduction

This document quickly sums up the history of development of units for CellML, and gives some justification for some of the changes in the way things have been done. It then goes on to propose some new ways of doing things, with particular regard to the SBML way of doing things. The January 12 scheme, described in Section 3, didn't last longer than three days however, and the subsequent changes proposed on January 15 are described in Section 4.

2 Early History

The CellML team intended to make the specification of units one of their priorities way back in the heady days of early development in 1999. Many hard years spent trying to develop working code from published models where the units had not been correctly or completely specified had led to this becoming a bit of an issue internally. So it was always clear that wherever a variable was first declared, and whenever bare numbers appeared in equations, units should be associated with these entities — the only real question was the best method.

In 1999 and early 2000, the method that had been developed used a **`units`** attribute as shown in Figure 1. Units were made up of the product of the base and derived SI units, where each base quantity was specified with a scaling prefix (e.g., *m* for *milli*) and an exponent. The notation was to put each triplet between square brackets, separating the members with commas. The absence of triplet parts corresponded to the appropriate part of `[0, dimensionless, 1]`. The whole system was incredibly cool because it was concise, and unambiguous, so you could fully specify the units everywhere. Also, the units strings could be easily split and manipulated using perl or XSLT. (Note that in some documentation the order of the triplet was `[quantity, scale, exponent]` and in others it was `[scale, quantity, exponent]` — the latter is more readable, so this is used in the examples below.)

```
<variable name="concentration_of_A" units="[m,mol,1][,l,-1]" />
```

FIGURE 1: The preferred method of specifying units as of early 2000.

In early 2000, some people may have pointed out that this was perhaps hard to approach for non-mathematical people and could potentially make CellML documents hard to read for biologists and the like. So a system was devised for associating human-readable strings with units declarations, where these strings could then be used for associating units with variables and bare numbers. This system is shown in Figure 2.

```
<!-- the <units_abbreviation_table> appears inside a <model> element -->
<units_abbreviation_table>
  <units abbreviation="dimensionless" expanded="[,,]" />
  <units abbreviation="concentration" expanded="[n,mol,1][m,m,-3]" />
</units_abbreviation_table>

<!-- the <variable> element appears inside a <component> element -->
<variable name="A" units="concentration" />
```

FIGURE 2: The units abbreviation scheme proposed early 2000.

In November, Warren met with the SBML[1] development team at the ISCB in Tokyo, Japan. It was agreed that one of the areas where CellML and SBML should be interoperable if possible was the area of units. The SBML team considered the triplet-based form of units definition in CellML too complex to parse, and suggested breaking down the strings into their individual components, each with their own attributes. This seemed reasonable. The SBML spec at that time didn't allow a scale factor on each quantity, so this was added to SBML and then the two specs could basically agree. At the end of the November meeting, the proposed CellML specification of units looked something like that shown in Figure 3.

```
<!-- the <units_abbreviation_table> appears inside a <model> element -->
<units_abbreviation_table>
  <units abbreviation="dimensionless" />

  <units abbreviation="concentration">
    <unit exponent="1" scale="n" type="mol" />
    <unit exponent="-3" scale="m" type="m" />
  </units>
</units_abbreviation_table>

<!-- the <variable> element appears inside a <component> element -->
<variable name="A" units="concentration" />
```

FIGURE 3: The units scheme agreed on after discussions with the SBML team.

3 Units Specification 12 January 2001

One of the key features ensuring robustness and re-usability of CellML components and models is the requirement that all variables and bare numbers have a set of units declared for them. This allows the connection of components and models where the units on variables that are to be mapped to one another are different (assuming that they are still of the same dimensionality), and for the consistency checking of equations.

CellML provides a dictionary of standard units that may be used in variable declarations and attached to bare numbers in math. This dictionary consists of the base SI units, the standard set of derived SI units, and

[1] http://www.cds.caltech.edu/erato/

some additional units commonly found in the types of biological models defined using CellML. References to these units should make use of the actual *name* of the units, rather than the standard abbreviation, thus avoiding confusion between units and scale factors. The full sets of base and derived SI units are shown in Figure 8 and Figure 9 respectively and the additional units are given in Figure 10. The full list of units that any CellML processing application should understand is given in Figure 4.

ampere[1]	dimensionless[3]	joule[2]	lumen[2]	numberof[3]	sievert[2]
becquerel[2]	farad[2]	katal[2]	lux[2]	ohm[2]	steradian[2]
calorie[3]	gram[3]	kelvin[1]	meter[1]	pascal[2]	tesla[2]
candela[1]	gray[2]	kilogram[1]	metre[1]	radian[2]	volt[2]
celsius[3]	henry[2]	liter[3]	mole[1]	second[1]	watt[2]
coulomb[2]	hertz[2]	litre[3]	newton[2]	siemen[2]	weber[2]

FIGURE 4: The dictionary of units keywords that CellML processing applications are expected to recognise. Keywords marked with a superscript of 1 are base SI units, those with a superscript of 2 are derived SI units, and those with a 3 are additions to the standard units defined purely for the convenience of model authors using CellML.

CellML also provides a facility whereby new units can be defined in terms of the units defined in the dictionary. This functionality allows the creation of complex units (made up of the products of simple units), define imperial units (which are expressed as a scaled version of an SI unit), and even create units that require an offset (such as Fahrenheit.) This allows model authors to work with whatever set of units they feel comfortable in, secure in the knowledge that their models can be integrated with those of other authors using other units.

New units are defined using the **<units>** element, which has a **name** attribute, the value of which is used to reference the units in variable declarations or on bare number elements. The contents of a **<units>** element is a sequence of **<unit>** elements, where each unit corresponds to one of the basic quantities, the product of which will be the final units type.

Every **<unit>** element may contain no content and may have up to five attributes. The most important of these, and the only one which is required, is the **type** attribute. The **type** attribute is used to set the base quantity for the current **<unit>** element, and its value must correspond to a string from the standard units dictionary, or to the value of the name of some previously defined units.

The optional **offset** attribute is used to shift the transformation between the current units and the base unit being referenced by the **type** attribute. This should only be necessary to define the Fahrenheit temperature scale. If the **offset** attribute is not present, it assumes a default value of 0.

The **scale** attribute, if present, can be used to indicate a scale attribute for the unit type. If its value is a letter, it must be from the standard set of unit pre-multiplier symbols given in Figure 5. If its value is an integer, then the type is pre-multiplied by 10 to the power of this number. If no scale attribute value is specified, it is assumed that the unit type stands alone i.e., is pre-multiplied by one.

The combination of **scale**, **offset** and **type** is then raised to some power equal to the value of the **exponent** attribute. This should be an integer. If no **exponent** attribute value is specified, it is assumed that the unit occurs once i.e., the **exponent** attribute has a default value of one. Note that an **exponent** attribute value of `"0"` (zero) has the effect of removing the parent **<unit>** element from the current units.

Finally a **multiplier** attribute can be used to pre-multiply the result so far by a further scale factor, allowing the introduction of floating point scale factors. This could be used, for instance, to define a "*pound*" unit in terms of the SI kilogram.

The **offset** attribute presents some mathematical problems in unit conversion, so some restrictions must be placed on its use. If the **offset** attribute is present on a **<unit>** element, it must be the sole

symbol	name	factor	symbol	name	factor
Y	yotta	10^{24}	d	deci	10^{-1}
Z	zetta	10^{21}	c	centi	10^{-2}
E	exa	10^{18}	m	milli	10^{-3}
P	peta	10^{15}	u	micro	10^{-6}
T	tera	10^{12}	n	nano	10^{-9}
G	giga	10^{9}	p	pico	10^{-12}
M	mega	10^{6}	f	femto	10^{-15}
k	kilo	10^{3}	a	atto	10^{-18}
h	hecto	10^{2}	z	zepto	10^{-21}
da	deka	10^{1}	y	yocto	10^{-24}

FIGURE 5: The set of letters that may be used in the **scale** attribute on a **<unit>** element and the corresponding scale factors that will pre-multiply the unit.

<unit> element within a **<units>** element. That units element then defines a straightforward conversion according to the following formula (where "*Type*" refers to the unit being defined):

$$x\ Type = (\ multiplier\ x + offset\ )(\ scale\ type\ )^{\ exponent}$$

Complex units are the product of numerous basic quantities, and are created by placing several **<unit>** elements inside a single **<units>** element. The conversion between the new units and the product of the units named in the **type** attributes of the **<unit>** elements is given by the following formula:

$$x\ units = [\ m1\ (\ s1\ t1\ )^{\ e1}\ ]\ [\ m2\ (\ s2\ t2\ )^{\ e2}\ ]\ ...\ [\ mn\ (\ sn\ tn\ )^{\ en}\ ]\ x$$

It is not possible to use the **offset** attribute on any **<unit>** element with an **exponent** attribute with value other than `"1"` or `"-1"`, or that is inside a **<units>** element containing another **<unit>** element with a positive **exponent** value. That is, a unit whose conversion involves an offset may not appear in a product on the "top line" of a units definition. Model authors may only create complex units from previously defined simple units that involve an offset in their conversion if they follow similar rules about exponents. These rules also apply to units defined with the `celsius` keyword from the standard dictionary — this unit is calculated with an offset from the base SI unit Kelvin.

Here are some practical examples of the effect of these rules: it is possible to define units corresponding to degrees fahrenheit (in fact these are defined in the CellML fragment in Figure 6). It is also possible to define, for example, inches per degree fahrenheit, but **not** fahrenheit inches or degrees fahrenheit squared. In the latter cases, a unit involving an offset appears in product on the top line of the units definition.

The CellML fragment in Figure 6 contains the definition of two simple units. In practice, software would usually want to perform the inverse transformation: i.e., given a number in the newly defined units, software would want to convert that back into SI units so that it could be used in simulation.

The first **<units>** element is used to define a litre (where we assign the new units the abbreviation `"l"` which doesn't clash with the keyword `litre` from the standard dictionary of units). In the example a litre is defined as 1000 cubic centimetres. It would also be possible to define a litre as one thousandth of a cubic metre or using any number of possible multipliers and scales. The formula we obtain from the **<units>** definition is:

$$x\ \texttt{l} = 1000\ x\ (\ c\ \texttt{metre}\ )^{\ 3}$$

```
<units name="l">
  <unit multiplier="1000" exponent="3" scale="c" type="metre" />
</units>

<units name="fahrenheit">
  <unit multiplier="0.5555556" offset="-32.0" type="celsius" />
</units>
```

FIGURE 6: Some examples demonstrating the use of the **<units>** and **<unit>** elements.

The second **<units>** element is used to define a degrees fahrenheit as a function of degrees celsius. The formula we obtain from this **<units>** element is:

$$x\ \texttt{fahrenheit} = 0.5555556\ (\ x - 32.0\ )\ \texttt{celsius}$$

The definition of some complex units is shown in Figure 7, where the definition of the later units is based on the earlier definitions. In the first units element, second is re-named `time`. In the second units element, `concentration` is defined as milli-moles per litre. Finally, `flux` is defined in terms of change of `concentration` with respect to `time`.

```
<units name="time">
  <unit type="second" />
</units>

<units name="concentration">
  <unit scale="m" type="mole" />
  <unit exponent="-1" type="litre" />
</units>

<units name="flux">
  <unit type="concentration" />
  <unit exponent="-1" type="time" />
</units>
```

FIGURE 7: Further examples of units definition including the definition of complex units.

4 Changes Post January 12 2001

Just when you think you have something good going, Poul Nielsen waltzes (struts?) in and decides he's not happy with it. Not long after we had what we thought was a complete units scheme with documentation suitable for inclusion in the CellML specification (above), Poul started recommending changes. These changes were the result of numerous close readings of the "A Short Introduction To CellML" paper being written for inclusion in the July edition of The Philosophical Transactions of the Royal Society of London

(numerous other changes to CellML resulting from the writing of this paper are described in the January 15 meeting minutes[2].) A quick summary of the changes and the justifications are as follows:

1. The **`scale`** attribute on the **`<unit>`** element was re-named the **`prefix`** attribute. This is what the people at NIST call it, and as we hope to be a respected standards organisation like NIST, we should follow existing standards ourselves wherever possible.
2. The **`type`** attribute on the **`<unit>`** element was re-named the **`units`** attribute. The word "type" was too ontology-like: Poul favoured a re-naming to **`units`**, Warren favoured a re-naming to **`name_ref`**, and Melanie suggested **`units_ref`**. Basically, we wanted something that could be thought of as consistent with everything else, and since we'd already broken consistency with SBML with the previous change, it didn't really matter if we became more inconsistent. Using **`units`** is consistent with the use of **`<variable>`** elements and also all referencing schemes throughout CellML.
3. It was decided that the values of the **`prefix`** attribute on the **`<unit>`** element should be the full name of the prefixes as they appear in the NIST table to be consistent with the values of the **`units`** attribute. So `"milli"` should be used in place of `"m"`. This also neatly sidesteps the mu/u/micro problem.
4. The `numberof` keyword was removed from the units dictionary. Poul didn't like `numberof` because it was a quantity (like "length") rather than a unit, and because there is implied integer behaviour that we possibly don't want. He suggested that the word `items` would be more technically correct. This keyword was left out of the paper, but may yet be re-introduced into the CellML specification after correspondence with the SBML folks.

5 Tables of SI and Additional CellML Units

base quantity	name	symbol
length	metre (or meter)	m
mass	kilogram	kg
time	second	s
electric current	ampere	A
thermodynamic temperature	kelvin	K
amount of substance	mole	mol
luminous intensity	candela	cd

FIGURE 8: The SI base units.

[2] http://www.cellml.org/private/progress_reports/20010115_meeting_minutes.html

quantity	name	symbol	in base units	in other SI units
plane angle	radian	rad	$m \cdot m^{-1}$	
solid angle	steradian	sr	$m^2 \cdot m^{-2}$	
frequency	hertz	Hz	s^{-1}	
force	newton	N	$m \cdot kg \cdot s^{-2}$	J/m
pressure	pascal	Pa	$m^{-1} \cdot kg \cdot s^{-2}$	N/m^2
energy, work, heat	joule	J	$m^2 \cdot kg \cdot s^{-2}$	$N \cdot m$
power, radiant flux	watt	W	$m^2 \cdot kg \cdot s^{-3}$	J/s
electric charge	coulomb	C	$s \cdot A$	$A \cdot s$
electric potential	volt	V	$m^2 \cdot kg \cdot s^{-3} \cdot A^{-1}$	W/A
capacitance	farad	F	$m^2 \cdot kg^{-1} \cdot s^4 \cdot A^2$	C/V
electric resistance	ohm	$\Omega(R)$	$m^2 \cdot kg \cdot s^{-3} \cdot A^{-2}$	V/A
conductance	siemens	S	$m^{-2} \cdot kg^{-1} \cdot s^3 \cdot A^2$	A/V
magnetic flux	weber	Wb	$m^2 \cdot kg \cdot s^{-2} \cdot A^{-1}$	$V \cdot s$
magnetic flux density	tesla	T	$kg \cdot s^{-2} \cdot A^{-1}$	Wb/m
inductance	henry	H	$m^2 \cdot kg \cdot s^{-2} \cdot A^{-2}$	Wb/A
luminous flux	lumen	lm		$cd \cdot sr$
illuminance	lux	lx		$m^{-2} \cdot cd \cdot sr$
activity	becquerel	Bq	s^{-1}	
absorbed dose	gray	Gy	$m^2 \cdot s^{-2}$	J/kg
dose equivalent	sievert	Sv	$m^2 \cdot s^{-2}$	J/kg
catalytic activity	katal	kat	$s^{-1} \cdot mol$	

FIGURE 9: The derived SI units.

quantity	name	symbol	in SI units
dimensionless	dimensionless		
energy, work, heat	calorie	cal	$x \cdot calorie = 4.1868 \cdot x \cdot joule$
temperature	celsius	$^\circ C$	$x \cdot celsius = (x + 273.15) \cdot kelvin$
mass	gram	g	$x \cdot gram = 0.001 \cdot x \cdot kilogram$
volume	litre (or liter)	l	$x \cdot litre = 0.001 \cdot x \cdot metre^3$
amount	numberof		

FIGURE 10: The additional units added to CellML's standard units dictionary to make it convenient for model authors to defi ne basic quantities.