CellML 2.0 Specification [May 2017 Draft]

Authors:

Michael T. Cooling*

Michael Clerx

Jonathan Cooper

David P. Nickerson

Contributors:

Jesús C. Fernández

David Brooks

Koray Atalag

Alan Garny

The authors also wish to acknowledge the significant contribution of the (discontinued) draft CellML 1.2 specification, much of the text of which was incorporated into this specification - although the semantics have changed considerably. The CellML 1.2 specification was itself the result of a collaborative effort by a number of researchers during 2008-2011:

Andrew K. Miller (who wrote the text reused here), Jonathan Cooper, Justin Marsh, Alan Garny, Randall Britten, Poul Nielsen and David P. Nickerson

* corresponding author: mtcooling.research@gmail.com

Contents

1. Definitions

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

The key phrase “information item”, as well as any specific type of information item such as an “element information item”, are to be interpreted as described in XML Information Set.

CellML infoset

> An XML information set containing a hierarchy of information items conforming to the rules described in this document. In this specification such infosets are assumed to be CellML 2.0 infosets.

CellML model

> A mathematical model represented by a hierarchy of one or more CellML infosets, according to the rules described in this document. In this specification, the topmost CellML infoset in this hierarchy is referred to as the top-level CellML infoset.

CellML processing software

> Software which processes CellML in accordance with the rules of this document.

Namespace

> An XML namespace, as defined in Namespaces in XML 1.1.

CellML namespace

> The CellML 2.0 namespace.

CellML 2.0 namespace

> The namespace `http://www.cellml.org/cellml/2.0#`.

MathML namespace

> The namespace `http://www.w3.org/1998/Math/MathML`.

CellML information item

> Any information item in the CellML namespace.

Basic Latin alphabetic character

> A Unicode character in the range U+0041 to U+005A or in the range U+0061 to U+007A.

European numeric character

> A Unicode character in the range U+0030 to U+0039.

Basic Latin alphanumeric character

> A Unicode character which is either a basic Latin alphabetic character or a European numeric character.

Basic Latin underscore

> The Unicode character U+005F.

Whitespace character

> Any one of the Unicode characters U+0020, U+0009, U+000D or U+000A.

2. General matters

2.1. CellML and XML

1. Every CellML infoset SHALL be represented in an XML document which conforms with the well-formedness requirements of XML 1.1.
2. In this document, the remaining provisions relating to CellML infosets SHALL be interpreted as being constraints on the XML information set represented by that CellML infoset.

2.2. Semantically equivalent CellML infosets

1. Two CellML infosets SHALL be deemed semantically equivalent in terms of model content if one can be transformed into the other by making zero or more of the following changes:
    1. Changing the representation of the XML file in ways which do not change the XML information set represented.
    2. Adding, removing, and/or modifying comment information items.
    3. Changing (inserting, removing, and/or modifying) one or more namespace information items, and/or modifying the prefix of one or more information items, without changing the namespace that any information item is in.
    4. The following paragraph applies only to character information items which are the direct child of an element information item in a CellML namespace, or in the MathML namespace.

       Inserting or removing character information items that consist entirely of whitespace characters, changing the number of whitespace characters in such an information item, or changing the number of whitespace characters at the beginning or end of a character information item.

2.3. Character information items

An element information item in the CellML namespace MUST NOT contain any character information items, except for character information items which consist entirely of whitespace characters.

2.4. Use of namespaces

1. Element and attribute information items in a CellML infoset MUST belong to one of the following namespaces, unless explicitly indicated otherwise:
    1. The CellML namespace
    2. MathML 2.0, where:
        1. The element information item or one of its ancestors is an element information item in the MathML namespace, with a local name equal to `math` (the math element information item); and

2. The math element information item forms the top-level of a valid MathML 2.0 tree.

2.5. XML Document Identifiers

1. Any element information item in the CellML namespace MAY contain an unprefixed attribute information item with local name `id`. This attribute information item SHALL be treated as having attribute type ID, as defined in section 3.3.1 of XML 1.1.

2.6. Specific information items

1. For the purposes of this specification, a specific information item is one of the following (see https://www.w3.org/TR/xml-infoset/#infoitem for definitions):
    1. A document information item;
    2. An element information item;
    3. An attribute information item;
    4. A processing instruction information item;
    5. An unexpanded entity reference information item;
    6. A document type declaration information item;
    7. An unparsed entity information item;
    8. A notational information item.
2. Specific information items MUST NOT appear in a CellML infoset except where explicitly allowed by this specification, or where allowed by a normative specification referenced by this specification.
3. The order in which specific information items appear, as children of an element information item defined in this specification, SHALL NOT affect the semantic interpretation of the CellML model.

3. Data representation formats in CellML

The following data representation formats are defined for use in this specification:

1. A CellML identifier:
    1. SHALL be a sequence of Unicode characters.
    2. SHALL NOT contain any characters except basic Latin alphanumeric characters and basic Latin underscores.
    3. SHALL contain one or more basic Latin alphabetic characters.
    4. SHALL NOT begin with a European numeric character.
    5. SHALL, when comparing two identifiers, be considered identical to another identifier if and only if both identifiers have identical sequences of Unicode characters.
2. A non-negative integer string:
    1. SHALL be a base 10 representation of a non-negative integer.
    2. SHALL consist entirely of European numeric characters.
3. An integer string:
    1. SHALL be a base 10 representation of an integer.
    2. SHALL, when the integer being represented is negative, consist of the basic Latin hyphen-minus character U+002D, followed by the non-negative integer string representation of the absolute value of the integer.
    3. SHALL, when the integer being represented is non-negative, consist of the non-negative integer string representation of the integer.
4. A basic real number string:
    1. SHALL be a base 10 representation of a real number.
    2. SHALL, when the basic real number being represented is negative, begin with the basic Latin hyphen-minus character U+002D as the sign indicator.
    3. MAY contain a single decimal point separator, which SHALL be the basic Latin full stop character U+002E.
    4. SHALL, other than the sign indicator and the decimal point separator, consist only of European numeric characters.
5. A real number string:
    1. SHALL be a base 10 representation of a real number $r = s \cdot 10^e$, where s is the significand, a real number, and e is the exponent, an integer.
    2. The representation of the number SHALL be the representation of the significand followed immediately by the representation of the exponent.
    3. The significand SHALL be represented as a basic real number string.
    4. A non-zero exponent SHALL be represented by an exponent separator character, followed by the integer string representation of the value of the exponent. The exponent separator character SHALL be either the basic Latin 'E' character U+0045 or the basic Latin 'e' character U+0065.
    5. If the exponent is zero, the exponent MAY be represented by an empty string, or MAY be represented according to the rule for non-zero exponent.

4. The `model` element information item

4.1. Top-level of CellML infosets

The top-level element information item in a CellML infoset MUST be an element information item in the CellML namespace with a local name equal to `model`. In this specification, the top-level element information item is referred to as the `model` element.

4.2. Specific information items

1. Every `model` element MUST contain an unprefixed `name` attribute. The value of the `name` attribute MUST be a valid CellML identifier.
2. A `model` element MAY contain zero or more additional specific information item children, each of which MUST be of one of the following types:
    1. A `component` element; or
    2. A `connection` element; or
    3. An `encapsulation` element; or
    4. An `import` element; or
    5. A `units` element;
3. A `model` element MUST NOT contain more than one `encapsulation` element.

5. The `import` element information item

An `import` element information item (referred to in this specification as an `import` element) is an element information item in the CellML namespace with a local name equal to `import`.

5.1. Specific information items

1. Every `import` element MUST contain an attribute information item in the namespace `http://www.w3.org/1999/xlink`, and with a local name equal to `href`. The value of this attribute SHALL be a valid locator `href`, as defined in section 5.4 of the XLink specification. The `href` attribute SHALL be treated according to the XLink specification, by applying the rules for simple-type elements. When describing an `import` element or one of its children, the phrase "imported CellML infoset" SHALL refer to the CellML infoset obtained by parsing the document referenced by the `href` attribute.
2. Every `import` element MAY contain zero or more specific information item children, each of which MUST be of one of the following types:
    1. An `import units` element; or
    2. An `import component` element.
3. The imported CellML infoset MUST NOT be semantically equivalent (see Semantically equivalent CellML infosets) to the importing CellML infoset. Any CellML infoset imported, directly or indirectly, by the imported CellML infoset MUST NOT be semantically equivalent to the importing CellML infoset.

6. The `import units` element information item

An `import units` element information item (referred to in this specification as an `import units` element) is an element information item in the CellML namespace with a local name equal to `units`, which appears as a child of an `import` element.

6.1. Specific information items

1. Every `import units` element MUST contain an unprefixed `name` attribute. The value of the `name` attribute MUST be a valid CellML identifier. The value of the `name` attribute MUST NOT be identical to the `name` attribute of any other `units` element that appears as the direct child of a `model` element or `import units` element in the CellML infoset.
2. Every `import units` element MUST contain an unprefixed `units_ref` attribute. The value of the `units_ref` attribute MUST be a valid CellML identifier. The value of the `units_ref` attribute MUST match the value of the `name` attribute on a `units` element or `import units` element in the imported CellML infoset. The value of the `units_ref` attribute MUST NOT match the value of the `units_ref` attribute on any sibling `import units` element.

7. The `import component` element information item

An `import component` element information item (referred to in this specification as an `import component` element) is an element information item in the CellML namespace with a local name equal to `component`, which appears as a child of an `import` element.

7.1. Specific information items

1. Every `import component` element MUST contain an unprefixed `name` attribute. The value of the `name` attribute MUST be a valid CellML identifier. The value of the `name` attribute MUST NOT be identical to the `name` attribute of any other `component` element or `import component` element in the CellML infoset.
2. Every `import component` element MUST contain an unprefixed `component_ref` attribute. The value of the `component_ref` attribute MUST be a valid CellML identifier. The value of the `component_ref` attribute MUST match the value of the `name` attribute on a `component` element or `import component` element in the imported CellML infoset. See also the Component reference section.

8. The `units` element information item

A `units` element information item (referred to in this specification as a `units` element) is an element information item in the CellML namespace with a local name equal to `units`, and with a `model` element as its parent.

8.1. Specific information items

1. Every `units` element MUST contain an unprefixed `name` attribute. The value of the `name` attribute MUST be a valid CellML identifier.
2. The value of the `name` attribute MUST NOT be identical to the `name` attribute of any other `units` element or `import units` element in the CellML infoset.
3. The value of the `name` attribute MUST NOT be equal to the name of any of the units listed in the Built-in units table.
4. A `units` element MAY contain one or more `unit` element children.

9. The `unit` element information item

A `unit` element information item (referred to in this specification as a `unit` element) is an element information item in the CellML namespace with a local name equal to `unit`, and with a `units` element as its parent.

9.1. Specific information items

1. Every `unit` element MUST contain an unprefixed `units` attribute information item. The value of the `units` attribute MUST be a valid units reference, as defined in the Units reference section.
    1. For the purpose of the constraint in the next paragraph, the `units` element inclusion digraph SHALL be defined as a conceptual digraph which SHALL contain one node for every `units` element in the CellML model. The `units` element inclusion digraph SHALL contain an arc from `units` element *A* to `units` element *B* if and only if `units` element *A* contains a `unit` element with `units` attribute value that is a units reference to `units` element *B*.
    2. The value of the `units` attribute MUST NOT be such that the `units` element inclusion digraph contains one or more cycles (in other words, units definitions must not be circular).
2. A `unit` element MAY contain any of the following unprefixed attribute information items:
    1. The `prefix` attribute. If present, the value of the attribute MUST meet the constraints specified in the Interpretation of units section.
    2. The `multiplier` attribute. If present, the value of the attribute MUST be a real number string.
    3. The `exponent` attribute. If present, the value of the attribute MUST be a real number string.

10. The `component` element information item

A `component` element information item (referred to in this specification as a `component` element) is an element information item in the CellML namespace with a local name equal to `component`, and which appears as a child of a `model` element.

10.1. Specific information items

1. Every `component` element MUST contain an unprefixed `name` attribute. The value of the `name` attribute MUST be a valid CellML identifier. The value of the `name` attribute MUST NOT be identical to the `name` attribute on any other `component` element or `import component` element in the CellML infoset.
2. A `component` element MAY contain one or more specific information item children, each of which MUST be of one of the following types:
    1. A `variable` element; or
    2. A `reset` element; or
    3. A `math` element

11. The `variable` element information item

A `variable` element information item (referred to in this specification as a `variable` element) is an element information item in the CellML namespace with a local name equal to `variable`, and which appears as a child of a `component` element.

11.1. Specific information items

1. Every `variable` element MUST have each of the following unprefixed attribute information items:
    1. The `name` attribute. The value of the `name` attribute MUST be a valid CellML identifier. The value of the `name` attribute MUST NOT be identical to the `name` attribute on any sibling `variable` element.
    2. The `units` attribute. The value of the `units` attribute MUST be a valid CellML identifier, and MUST meet the constraints described in the Effect of units on variables section.
2. Every `variable` element MAY contain one or more of the following unprefixed attribute information items:
    1. The `interface` attribute. If the attribute is present, it MUST have one of the values `public`, `private`, `public_and_private`, or `none`.
    2. The `initial_value` attribute. If the attribute is present, it MUST meet the requirements described by the Interpretation of initial values section.

12. The `reset` element information item

A `reset` element information item (referred to in this specification as a `reset` element) is an element information item in the CellML namespace with a local name equal to `reset`, and which appears as a child of a `component` element.

12.1. Specific information items

1. Every `reset` element MUST have each of the following unprefixed attribute information items:
    1. The `variable` attribute. The value of the `variable` attribute MUST be a variable reference to a variable defined within the `component` element parent of the `reset` element.
    2. The `order` attribute. The value of the `order` attribute MUST be an integer string. The value of the `order` attribute MUST be unique for all `reset` elements for variables that are in the same connected variable set (see Interpretation of map_variables).
    3. A `reset` element MUST contain one or more specific information item children, each of which MUST be a `when` element.

13. The `when` element information item

A `when` element information item (referred to in this specification as a `when` element) is an element information item in the CellML namespace with a local name equal to `when`, and which appears as a child of a `reset` element.

13.1. Specific information items

1. Every `when` element MUST contain an unprefixed `order` attribute. The value of the `order` attribute MUST be an integer string. All sibling `when` elements MUST have a unique value for `order`.
2. A `when` element MUST contain two specific information item children, each of which MUST be `math` elements.

14. The `math` element information item

A `math` element information item (referred to in this specification as a `math` element) is an element information item in the MathML namespace that appears as a direct child of a `component` element, or as a direct child of a `when` element.

14.1. Specific information items

1. A `math` element MUST be the top-level of a content MathML tree, as described in MathML 2.0.
2. Each element information item child of a `math` element MUST have an element-type name that is listed in the Supported MathML Elements table, and where allowable types are listed in that table the element MUST exhibit one of those types.
3. Every variable name given using the MathML `ci` element MUST be a variable reference to a `variable` within the `component` element that the `math` element is contained within.
4. Any MathML `cn` elements MUST each have an attribute information item in the CellML namespace, with a local name equal to `units`. The value of this attribute information item MUST be a valid units reference.

Table: Supported MathML Elements

Element Category	**Element List**
Simple Operands	<ci>, <cn> (allowable types: real (default), e-notation), <sep>
Basic Structural	<apply>, <piecewise>, <piece>, <otherwise>
Relational and Logical Operators	<eq>, <neq>, <gt>, <lt>, <geq>, <leq>, <and>, <or>, <xor>, <not>
Arithmetic Operators	<plus>, <minus>, <times>, <divide>, <power>, <root>, <abs>, <exp>, <ln>, <log>, <floor>, <ceiling>, <min>, <max>, <rem>,
Calculus Elements	<diff>
Qualifier Elements	<bvar>, <logbase>, <degree> (child of <root> or <diff>)
Trigonometric Operators	<sin>, <cos>, <tan>, <sec>, <csc>, <cot>, <sinh>, <cosh>, <tanh>, <sech>, <csch>, <coth>, <arcsin>, <arccos>, <arctan>, <arcsec>, <arccsc>, <arccot>, <arcsinh>, <arccosh>, <arctanh>, <arcsech>, <arccsch>, <arccoth>
Mathematical and Logical Constants	<pi>, <exponentiale>, <notanumber>, <infinity>, <true>, <false>

15. The `encapsulation` element information item

An `encapsulation` element information item (referred to in this specification as an `encapsulation` element) is an element information item in the CellML namespace with a local name equal to `encapsulation`, and which appears as a child of a `model` element.

15.1. Specific information items

1. Every `encapsulation` element MUST contain one or more `component_ref` elements.

16. The `component_ref` element information item

A `component_ref` element information item (referred to in this specification as a `component_ref` element) is an element information item in the CellML namespace with a local name equal to `component_ref`, and which appears as a child of an `encapsulation` element.

16.1. Specific information items

1. Every `component_ref` element MUST contain an unprefixed `component` attribute information item. The value of this attribute MUST be a valid CellML identifier, and MUST match the `name` attribute on a `component` element or an `import component` element in the CellML infoset.
2. Every `component_ref` element MAY in turn contain zero or more `component_ref` element children.
3. `component_ref` elements which are immediate children of `encapsulation` elements MUST each contain at least one `component_ref` element child.

17. The `connection` element information item

A `connection` element information item (referred to in this specification as a `connection` element) is an element information item in the CellML namespace with a local name equal to `connection`, and which appears as a child of a `model` element.

17.1. Specific information items

1. Each `connection` element MUST contain an unprefixed `component_1` attribute. The value of the `component_1` attribute MUST be a valid CellML identifier. The value of this attribute MUST be equal to the `name` attribute on a `component` or `import component` element in the CellML infoset (see Component reference).
2. Each `connection` element MUST contain an unprefixed `component_2` attribute. The value of the `component_2` attribute MUST be a valid CellML identifier. The value of this attribute MUST be equal to the name attribute on a `component` or `import component` element in the CellML infoset (see Component reference). It MUST NOT be equal to the value of the `component_1` attribute.
3. A CellML infoset MUST NOT contain more than one `connection` element with a given pair of `component`s referenced by the `component_1` and `component_2` attribute values, in any order.
4. Every `connection` element MUST contain one or more `map_variables` elements.

18. The `map_variables` element information item

A `map_variables` element information item (referred to in this specification as a `map_variables` element) is an element information item in the CellML namespace with a local name equal to `map_variables`, and which appears as a child of a `connection` element.

18.1. Specific information items

1. Each `map_variables` element MUST contain an unprefixed `variable_1` attribute. The value of the `variable_1` attribute MUST be a valid CellML identifier. The value of this attribute MUST be equal to the `name` attribute on a `variable` element child of the `component` element or `import component` element referenced by the `component_1` attribute on the `connection` element which is the parent of this element.
2. Each `map_variables` element MUST contain an unprefixed `variable_2` attribute. The value of the `variable_2` attribute MUST be a valid CellML identifier. The value of this attribute MUST be equal to the `name` attribute on a `variable` element child of the `component` element or `import component` element referenced by the `component_2` attribute on the `connection` element which is the parent of this element.
3. A `connection` element MUST NOT contain more than one `map_variables` element with a given `variable_1` attribute value and `variable_2` attribute value pair.

19. Interpretation of CellML models

19.1. Interpretation of imports

1. Each `import` element present in a CellML infoset (the importing infoset) SHALL define a new and separate instance of the CellML infoset referenced by the `href` attribute (the imported infoset). See Units reference and Component reference for the specifics of importing units and components.

19.2. Units reference

1. A units reference SHALL be a CellML identifier, and in terms of the semantics of the model content SHALL be interpreted dependent on the context of the CellML model in which it occurs, according to the units referencing rules defined later in this section.
2. A CellML infoset MUST NOT contain a units reference for which no referencing rule can be held to have been followed.
3. The units referencing rules are:
    1. Where there is a `units` element with a `name` attribute identical to the units reference, then the units reference SHALL refer to that `units` element.
    2. Where there is an `import units` element in the CellML infoset, such that the `import units` element has a `name` attribute identical to the units reference, then the units reference SHALL be treated with respect to referencing rules as if the units reference appeared in the imported infoset, in an information item descended from the imported infoset's `model` element, and referred to the `name` specified in the `units_ref` attribute of the `import units` element.
    3. Where the units reference is equal to the value in the 'Name' column of the Built-in units table, then the units reference SHALL be a reference to the built-in unit corresponding to that row of the table.

Table: Built-in units

Name	Multiplier(s)	Unit reduction tuple (dimension, exponent) set
ampere	-	-
becquerel	1	(second, -1)
candela	-	-
coulomb	1, 1	(second, 1), (ampere,1)
dimensionless	-	-
farad	1, 1, 1, 1	(metre, -2), (kilogram, -1), (second, -4), (ampere,2)
gram	0.001	(kilogram,1)
gray	1, 1	(metre, 2), (second, -2)
henry	1, 1, 1, 1	(metre, 2), (kilogram, 1), (second, -2), (ampere, -2)
hertz	1	(second, -1)
joule	1, 1, 1	(metre, 2), (kilogram, 1), (second, -2)

katal	1, 1	(second, -1), (mole, 1)
kelvin	-	-
kilogram	-	-
liter	0.001	(metre, 3)
litre	0.001	(metre, 3)
lumen	1	(candela, 1)
lux	1, 1	(metre, -2), (candela, 1)
meter	1	(metre, 1)
metre	-	-
mole	-	-
newton	1, 1, 1	(metre, 1), (kilogram, 1), (second, -2)
ohm	1, 1, 1, 1	(metre, 2), (kilogram, 1), (second, -3), (ampere, -2)
pascal	1, 1, 1	(metre, -1), (kilogram, 1), (second, -2)
radian	1, 1	(metre, 1), (metre, -1)
second	-	-
siemens	1, 1, 1, 1	(metre, -2), (kilogram -1), (second, 3), (ampere, 2)
sievert	1, 1	(metre, 2), (second, -2)
steradian	1, 1	(metre, 2), (metre, -2)
tesla	1, 1, 1	(kilogram, 1), (second, -2), (ampere, -1)
volt	1, 1, 1, 1	(metre, 2), (kilogram, 1), (second , -3), (ampere, -1)
watt	1, 1, 1	(metre, 2), (kilogram, 1), (second, -3)
weber	1, 1, 1, 1	(metre, 2), (kilogram, 1), (second, -2), (ampere, -1)

19.3. Interpretation of units

1. The `units` element SHALL be interpreted as the product of its `unit` element children, according to the following rules:
    1. The prefix term is a conceptual property of `unit` elements. If the `unit` element does not have a `prefix` attribute information item, the prefix term SHALL have value 0.0. If the `prefix` attribute information item has a value which is an integer string, then the value of the prefix term SHALL be the numerical value of that string. Otherwise, the `prefix` attribute information item MUST have a value taken from the 'Name' column of the Prefix values table, and the prefix term SHALL have the value taken from the 'Value' column of the same row.
    2. The exponent term is a conceptual property of `unit` elements. If a `unit` element has no `exponent` attribute information item, the exponent term SHALL have value 1.0. Otherwise, the value of the `exponent` attribute information item MUST be a real number string, and the value of the exponent term SHALL be the numerical value of that string.

    3. The multiplier term is a conceptual property of `unit` elements. If a `unit` element has no `multiplier` attribute information item, the multiplier term SHALL have value 1.0. Otherwise, the value of the `multiplier` attribute information item MUST be a real number string, and the value of the multiplier term SHALL be the numerical value of that string.
    4. The relationship between the product, *P,* of numerical values given in each and every child `unit` element units, to a numerical value, *x,* with units given by the encompassing `units` element, SHALL be

$$x[u_x] = \frac{1}{\left({10^{p_1}}^{e_1}..{10^{p_n}}^{e_n}\ m_1..m_n\right)}\left[\frac{u_x}{u_1^{e_1}..u_n^{e_n}}\right]P\left[u_1^{e_1}..u_n^{e_n}\right]$$

       where: u_x denotes the units of the `units` element; p_i, e_i, m_i, and u_i refer to the prefix, exponent and multiplier terms and units of the *i*th `unit` child element, respectively. Square brackets encompass the units of numerical values.
2. For the purposes of this specification, the “irreducible units” of a model SHALL consist of 1) the units defined in a model that are not defined in terms of other units (i.e. the set of `units` elements in the CellML model which have no `unit` child elements), and 2) built-in irreducible units (those built-in units with ‘-’ in the ‘Unit Reduction...’ column of the Built-in units Table) referenced by variables or other units in the model.
3. The “unit reduction” is a conceptual property of `units` elements. It consists of a set of tuples where each tuple is composed of a) a unit name and b) a real-valued exponent. Tuples SHALL be determined as follows:
    1. If the `units` element has no `unit` child elements, then the set of tuples SHALL have a single member, which SHALL consist of the name of the `units` element and the exponent 1.0.
    2. If the `units` element has one or more `unit` child elements, then the set of tuples SHALL consist of the entire collection of tuples given by all `unit` child elements. Tuples for each `unit` child element SHALL be determined as follows:
        1. Where the units reference of the `unit` child element is to a single unit which is an irreducible unit, then the set of tuples SHALL have a single member, which SHALL consist of the name of the irreducible unit being referenced and the exponent 1.0.
        2. Where the units reference of the `unit` child element is to built-in units other than an irreducible unit, then the tuples SHALL be derived directly from the Built-in units table. Specifically, the set of tuples SHALL consist of the tuples given in the ‘Unit reduction tuple ... set’ column of the row for which the value in the ‘Name’ column matches the name of the units reference.
        3. Where the units reference of the `unit` child element is to a unit which is neither built-in, nor an irreducible unit, the set of tuples SHALL be defined recursively as the set of tuples for the `units` element so referenced.

4. The exponents of each tuple in the set for the current `unit` element, as derived by following rule 3.2.1, 3.2.2 or 3.2.3 above, SHALL be multiplied by the exponent term of the current, referencing, `unit` element.

3. Tuples which have the name element of ‘dimensionless’ SHALL be removed from the set of tuples. Note that this can result in the set of tuples being empty.
4. Where the set of tuples consists of tuples which have the same name element, those tuples SHALL be combined into a single tuple with that name element and an exponent being the sum of the combined tuples’ exponents. If the resulting tuple’s exponent term is zero, the tuple SHALL be removed from the set of tuples. Note that this can result in the set of tuples being empty.

Table: Prefix values

Name	Value
yotta	24
zetta	21
exa	18
peta	15
tera	12
giga	9
mega	6
kilo	3
hecto	2
deka	1
deci	−1
centi	−2
milli	−3
micro	−6
nano	−9
pico	−12
femto	−15
atto	−18
zepto	−21
yocto	−24

19.4. Component reference

1. A component reference SHALL be the name of a component, and SHALL be interpreted based on the context within the CellML model in which it occurs.

2. A component reference present in an information item which is a descendant of a `model` element SHALL be identical to either the `name` attribute on a `component` element or to the `name` attribute on an `import component` element.
3. A component reference which is identical to the `name` attribute on a `component` element SHALL be treated as a reference to that `component` element.
4. A component reference which is identical to the `name` attribute on an `import component` element SHALL be treated for the purposes of referencing as if the component reference appeared in the imported model, and referred to element with the `name` specified in the `component_ref` attribute of the `import component` element.
5. It is noted, for the avoidance of doubt, that CellML models MAY apply the previous rule recursively, to reference an `import component` element which in turn references another `import component` element.

19.5. Variable reference

1. When present in an information item which is a descendant of a `component` element, a variable reference SHALL be the name of a variable, and SHALL refer to the `variable` element in that component with a `name` attribute identical to the variable reference.
2. In all other cases, a variable reference SHALL consist of a component reference and a variable name. In this case, the variable reference SHALL be treated as if it was just the variable name present in the `component` element referenced by the component reference.

19.6. Interpretation of initial values

1. The `initial_value` attribute of a `variable` element MUST either be a real number string, or a variable reference.
2. The conditions when initial values hold are (by design) not defined in a CellML model document. It is intended that those conditions be supplied by CellML processing software where appropriate.
3. Where the `initial_value` attribute has a real number value, it SHALL be interpreted as a statement that the variable on which the attribute appears is equal to that real number value, under the conditions when the initial value holds.
4. Where the `initial_value` attribute is a variable reference, it SHALL be interpreted as a statement that the variable on which the attribute appears is equal to the referenced variable under the conditions when the initial value holds.

19.7. Effect of units on variables

1. The value of the `units` attribute on every `variable` element MUST be a valid units reference. The target of this units reference is referred to as the variable units, and the corresponding unit reduction (see Interpretation of units) is referred to as the variable unit reduction.

19.8. Interpretation of mathematics

1. The following `component` elements SHALL, for the purposes of this specification, be “pertinent component elements”:

    1. All `component` elements in the top-level CellML infoset for the CellML model;
    2. All `component` elements referenced by `import component` elements (see The import component element information item) in the top-level CellML infoset; and
    3. All `component` elements which are descendants in the encapsulation digraph (see Interpretation of encapsulation) of a pertinent `component` element.
2. Every MathML element in the CellML model which appears as a direct child information item of a MathML `math` element information item, which in turn appears as a child information item of a pertinent `component` element, SHALL be treated, in terms of the semantics of the mathematical model, as a statement which holds true unconditionally.
3. Units referenced by a `units` attribute information item SHALL NOT affect the mathematical interpretation of the CellML model. However, CellML processing software MAY use this information to assist the user in the detection and correction of units errors in the CellML model.

19.9. Interpretation of encapsulation

1. For the purposes of this specification, there SHALL be a "conceptual encapsulation digraph" in which there is one node for every component in the CellML model.
2. Where a `component_ref` element appears as a child of another `component_ref` element, there SHALL be an arc in the encapsulation digraph, and that arc SHALL be from the node corresponding to the component referenced by the parent `component_ref` element, and to the node corresponding to the component referenced by the child `component_ref` element.
3. The encapsulation digraph MUST NOT contain any loops, and MUST NOT contain any directed cycles.
4. The encapsulated set for a component A SHALL be the set of all components B such that there exists an arc in the encapsulation digraph from the node corresponding to A to the node corresponding to B.
5. The encapsulation parent for a component A SHALL be the component corresponding to the node which is the parent node in the encapsulation digraph of the node corresponding to A.
6. The sibling set for a component A SHALL be the set of all components which have the same encapsulation parent as A, or in the case that A has no encapsulation parent, SHALL be the set of all components which do not have an encapsulation parent.
7. The hidden set for a component A SHALL be the set of all components B where component B is not in the encapsulated set for component A, and component B is not the encapsulation parent of component A, and component B is not in the sibling set for component A.
8. There MUST NOT be a `connection` element such that the component referenced by the `component_1` attribute is in the hidden set of the component referenced by the `component_2` attribute, nor vice versa.

19.10. Interpretation of map_variables

1. For the purposes of this specification, a variable equivalence (conceptual) network SHALL be a graph with one node for every `variable` element in the CellML model.
2. For each `map_variables` element present in the CellML model, we define variables *A* and *B* for use in the rules in this section as follows.
    1. Variable *A* SHALL be the variable referenced by the encompassing `connection` element's `component_1` and this `map_variables` element's `variable_1` attributes.
    2. Variable *B* SHALL be the variable referenced by the encompassing `connection` element's `component_2` and this `map_variables` element's `variable_2` attributes.
3. For every `map_variables` element present in the CellML model, there SHALL be an arc in the variable equivalence network.
    1. One endpoint of the arc in the variable equivalence network SHALL be the node corresponding to variable *A*.
    2. One endpoint of the arc in the variable equivalence network SHALL be the node corresponding to variable *B*.
4. CellML models MUST NOT contain any pair of `map_variables` elements which define the same arc in the variable equivalence network (according to the previous paragraph).
5. The variable equivalence network MUST NOT contain any cycles (that is, it must be a tree).
6. For each `map_variables` element present in the CellML model, the variable unit reduction (see Effect of units on variables) of variable *A* MUST have an identical set of tuples to the variable unit reduction of variable *B*. Two sets of tuples SHALL be considered identical if all of the tuples from each set are present in the other, or if both sets are empty. Two tuples are considered identical if and only if both the name and exponent value of each tuple are equivalent.
7. Processing software concerned with providing numerical solutions to the mathematical model MAY provide automatic numerical conversion between variables which have the same variable unit reduction but different multipliers (see Interpretation of units) by employment of conversion factors.
8. For a given variable, the available interfaces SHALL be determined by the `interface` attribute information item on the corresponding `variable` element as follows.
    1. A value of `public` specifies that the variable has a public interface.
    2. A value of `private` specifies that the variable has a private interface.
    3. A value of `public_and_private` specifies that the variable has both a public and a private interface.
    4. A value of `none` specifies that the variable has no interface.
    5. If the `interface` attribute information item is absent, then the variable has no interface.

9. The applicable interfaces for variables *A* and *B* SHALL be defined as follows.
    1. When the parent `component` element of variable *A* is in the sibling set of the parent `component` element of variable *B*, the applicable interface for both variables *A* and *B* SHALL be the public interface.
    2. When the parent `component` element of variable *A* is in the encapsulated set of the parent `component` element of variable *B*, the applicable interface for variable *A* SHALL be the public interface, and the applicable interface for variable *B* SHALL be the private interface.
    3. When the parent `component` element of variable *B* is in the encapsulated set of the parent `component` element of variable *A*, the applicable interface for variable *A* SHALL be the private interface, and the applicable interface for variable *B* SHALL be the public interface.
10. CellML models MUST NOT contain a `map_variables` element where the applicable interface of variable *A* or *B* is not defined or is not an available interface.
11. For the purposes of this specification, the `variable` elements in a CellML model SHALL be treated as belonging to one of one or more disjoint "connected variable sets". Each set of "connected variables" is the set of all `variable` elements for which the corresponding nodes in the variable equivalence network form a weakly connected subgraph. Each set of connected variables represents one variable in the underlying mathematical model.

19.11. Interpretation of variable resets

1. Each `reset` element describes a change to be applied to the variable referenced by the `variable` attribute when specified conditions are met during the simulation of the model.
2. All `reset` elements SHALL be considered sequentially for the connected variable set (see Interpretation of map_variables) to which the referenced variable belongs. The sequence SHALL be determined by the value of the `reset` element's `order` attribute, lowest (least positive) having priority.
3. The change, and conditions for the change, to a variable for a given `reset` element SHALL be defined by the evaluation of that element's `when` child elements.
    a. A `when` element SHALL be deemed to be true when the evaluation of the MathML expression encoded in first child element of the `when` element changes from the boolean expression false to true.
    b. If a `when` element is deemed to be true, then false, then true again during the same 'instant' or interval of integration, it SHALL nevertheless be held to be false.
    c. The second child element of a `when` element SHALL define the MathML expression to be evaluated and assigned to the parent `reset` element's referenced variable when the `when` element is deemed to be true.
    d. Consideration of the set of `when` child elements of a given `reset` element SHALL stop at the first occurrence of a `when` deemed to be true.

e. The order of consideration of the set of `when` child elements SHALL be based on the value of the `when` element's `order` attribute, lowest having priority.

20. References

RFC 2119: Key words for use in RFCs to Indicate Requirement Levels (March 1997)

Extensible Markup Language (XML) 1.1 (Second Edition) (16 August 2006)

XML Information Set (Second Edition) (4 February 2004)

Namespaces in XML 1.1 (Second Edition) (16 August 2006)

Mathematical Markup Language (MathML) Version 2.0 (Second Edition) (21 October 2003)

XML Linking Language (XLink) Version 1.0 (27 June 2001)