
Simple Reaction / Pathway Models

Melanie Nelson

Table of Contents

Introduction	1
Basic Model Structure	1
Analysis of the XML	3
The model element and declaration of units	3
The environment component	3
The reactant/product components	4
The basic_reaction component	5
Connections	10
Download This Model	11

Introduction

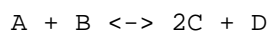
This document introduces the “best practice” for marking up models of biochemical reactions in CellML, using an extremely simple reaction model as an example. It includes fragments of CellML taken from a CellML document in which the model is defined in its entirety. This example shows the basics of pathway model structure in terms of CellML components and connections. It also includes the mathematics corresponding to the reaction and the associated conservation equations.

Much of the discussion in this document is similar to the analysis of the simple electrophysiological model [[../electrophysiological_models/basic_ep_models/basic_ep_model_doc.html](#)], and readers are referred to that document as appropriate.

You might find it helpful to refer to a full printout of the CellML document that makes up this example as you progress through this documentation. The various forms available online are presented in the section “Download This Model”.

Basic Model Structure

This model defines the single reversible reaction



The conventional rendering (pathway diagram) for the simple model is shown in Figure 1. In this rendering the reactants and products are shown as rounded rectangles on opposite ends of bidirectional arrows representing the reaction.

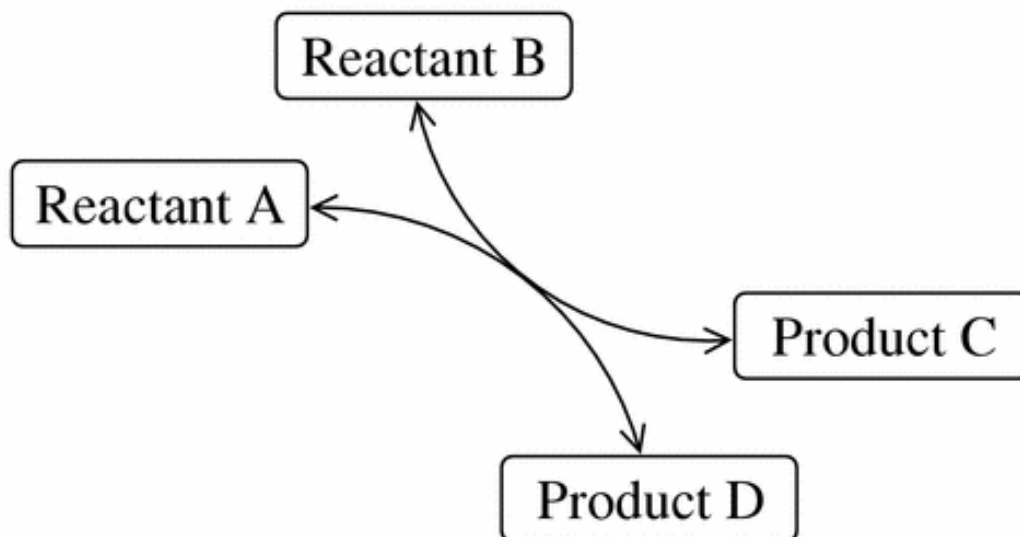


Figure 1. The conventional rendering (pathway diagram) for the simple reaction model.

In CellML, models are thought of as connected networks of discrete components. These components may correspond to physiologically separated regions or chemically distinct objects, or may be useful modelling abstractions. This simple reaction model has five components representing chemically distinct objects (two reactants, two products and the reaction itself) and one component defined purely for modelling convenience, which stores environment variables such as time. The CellML rendering of the simple reaction model is shown in Figure 2 (the different shapes in the diagram are explained in the notation guide [[./../introduction/notation.html](#)]). The component created to store environment variables is shown in Figure 2 to illustrate how these variables are handled in CellML. However, the **environment** component is often omitted from the CellML network rendering, as it will always exist, and it is usually connected to every other component.

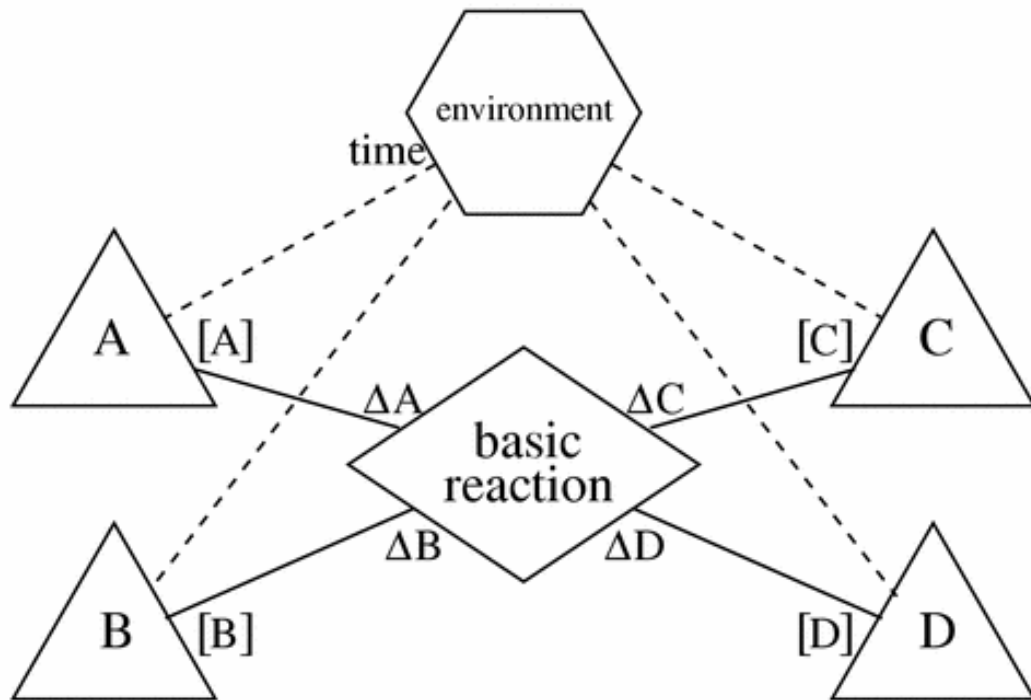


Figure 2. A network diagram for the simple reaction example. The model has a component for each of the reactants and products, a component for the reaction, and an “abstract” component that is used as a convenient container for the variable `time`. The variables in the model are shown next to the components in which their value may be modified and alongside the connections along which they are passed to other components (where their value may not be modified). The dotted lines marking the connections from the `environment` component to the other components have no special significance — they are dotted because they would generally be omitted from the rendering.

Analysis of the XML

The model element and declaration of units

The root `<model>` element of the CellML document for this simple reaction model is identical to the corresponding element in the simple electrophysiological model [../electrophysiological_models/basic_ep_models/basic_ep_model_doc.html], and the set of `<units>` elements (with which a set of named units is created for use in the model) is extremely similar to the set defined in the simple electrophysiological model (readers should refer to the analysis of that model for discussion of these parts of the model).

The environment component

The model consists of six components: one for each of the two reactants and two products, one for the reaction itself, and one that acts as a convenient container for variables that don't belong anywhere else. The `environment` component declares variables that are used by all or most of the other components in the model. In the case of this simple example model, the only global variable is the independent variable `time`. In a more complex model, an environment component might be used to define model-wide constants or concentrations of cofactors that appear in several unrelated reactions. Each `environment` variable is declared with a `public_interface` value of `"out"` and can be mapped to a corresponding variable with a `public_interface` value of `"in"` in the components where it is used. The

definition of the **environment** component for the example model is given in Figure 3.

```
<component name="environment">  
  <variable name="time" public_interface="out" units="time" />  
</component>
```

Figure 3. The definition of the **environment** component. This component is used to declare the independent variable **time**, which is used by the components representing the chemical species in the reaction.

Note that the name **environment** has absolutely no special significance, but is simply a human-readable identifier enabling the modeller to determine the purpose of this component. (We expect that software would make use of ontology information associated with the component to work out that this component would not be rendered in the same way as a reactant component, for example).

The reactant/product components

The next four components after the **environment** component correspond to the containers for the reactants and products and assume the name of the chemical species they contain: **A**, **B**, **C**, and **D**. The definition of the component for reactant **A** is shown in Figure 4. The definitions of the other three components are essentially identical.

```
<component name="A">  
  <variable name="A" public_interface="out" units="concentration_units" />  
  <variable name="delta_A" public_interface="in" units="flux_units" />  
  <variable name="time" public_interface="in" units="second" />  
  
  <math xmlns="http://www.w3.org/1998/Math/MathML">  
    <apply><eq />  
      <apply><diff />  
        <bvar><ci> time </ci></bvar>  
        <ci> A </ci>  
      </apply>  
      <ci> delta_A </ci>  
    </apply>  
  </math>  
</component>
```

Figure 4. The definition of the reactant **A** component. The component defines three variables for intern-

al use, as well defining the equation that governs the concentration of the species **A** with MathML.

Each reaction species component defines three variables for use internally. The first is the concentration of the chemical species that the component represents. This variable has the same name as the parent component (each component in the model and each variable in a component must be named uniquely, but there is no restriction on a variable having the same name as a component). This variable is declared with a **public_interface** attribute value of "out", so its value can be modified in the current component, and is available to other components that are connected to this one.

The second variable in the **A** component is **delta_A**, which represents the change in the concentration of species **A** as a result of the reaction. It has a **public_interface** attribute value of "in". Its value is imported from the **basic_reaction** component, in which it is declared with a **public_interface** value of "out".

Finally, the independent variable **time** is declared with a **public_interface** value of "in", allowing it to be imported from the **environment** component.

The component also defines the equation that governs the conservation of the species **A** using MathML [<http://www.w3.org/TR/MathML2/>] elements. The equations are contained in a block with a root element **<math>**. The **<math>** element includes a default namespace declaration (the **xmlns** attribute), which overrides the default CellML namespace declaration on the document's **<model>** element to place the **<math>** element and all of its children in the MathML namespace. The equation states that the rate of change of **A** with respect to time is equal to **delta_A**.

In a more complicated model, the conservation equation might set the rate of change of a species equal to the sum of a number of delta variables, where each delta variable represents the change in concentration of a species due to a single reaction.

The basic_reaction component

The last **<component>** element defined in the model is the one corresponding to the reaction itself. It has the name of **basic_reaction**. The component definition is reproduced in Figure 5. The purpose of the reaction component is to define the mathematics of the reaction itself. The **basic_reaction** component calculates *delta* variables, which represent the changes in the concentrations of the participating species due to the reaction process. These delta variables are exposed in such a way that they can be used in conservation equations stored in the species components.

```
<component name="basic_reaction">  
  
  <variable name="A" public_interface="in" units="concentration_units" />  
  
  <variable name="B" public_interface="in" units="concentration_units" />  
  
  <variable name="C" public_interface="in" units="concentration_units" />  
  
  <variable name="D" public_interface="in" units="concentration_units" />  
  
  
  <variable name="delta_A" public_interface="out" units="flux_units" />  
  
  <variable name="delta_B" public_interface="out" units="flux_units" />  
  
  <variable name="delta_C" public_interface="out" units="flux_units" />  
  
  <variable name="delta_D" public_interface="out" units="flux_units" />
```

```
<variable name="k_forward" units="2nd_order_rate_constant" />
<variable name="k_reverse" units="3rd_order_rate_constant" />
<variable name="r" units="flux_units" />

<reaction reversible="yes">
  <variable_ref variable="A">
    <role
      role="reactant" direction="forward"
      delta_variable="delta_A" stoichiometry="1" />
  </variable_ref>
  <variable_ref variable="B">
    <role
      role="reactant" direction="forward"
      delta_variable="delta_B" stoichiometry="1" />
  </variable_ref>
  <variable_ref variable="C">
    <role
      role="product" direction="forward"
      delta_variable="delta_C" stoichiometry="2" />
  </variable_ref>
  <variable_ref variable="D">
    <role
      role="product" direction="forward"
      delta_variable="delta_D" stoichiometry="1" />
  </variable_ref>
  <variable_ref variable="r">
    <role role="rate">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        ...
      </math>
    </role>
  </variable_ref>
</reaction>
</component>
```

Figure 5. The `<component>` element of the simple reaction model, which contains the definition of the actual reaction. The mathematics associated with the reaction rate calculation have been omitted.

The `basic_reaction` component defines a number of variables. The first set of variables are those whose value is imported from another component for use internally: the concentrations of the two reactants (**A** and **B**) and the two products (**C** and **D**). All of these variables are declared with a `public_interface` attribute value of "in". The second set of variables declared in the reaction component are the change in concentration variables, `delta_A`, `delta_B`, `delta_C`, and `delta_D`. The values of these variables are calculated within the reaction component and exposed to the other components in the model, so these variables are declared with a `public_interface` attribute value of "out". The third set of variables are purely for use internally, and so have no `public_interface` attribute. The first two of these, `k_forward` and `k_reverse`, are the rate constants used in the calculation of the reaction rate, which is the third variable `r`. The separate calculation of the reaction rate allows the modeller to make use of the implied mathematics properties of the `<role>` elements to concisely specify the values of the delta variables.

One of the more specialized elements in the CellML vocabulary is the `<reaction>` element, which follows the variable declarations inside the `basic_reaction` component. Reaction elements are used in CellML to provide an easy way of specifying which species (as represented by variables) take part in a reaction and indicating their role in the reaction. Although this information could sometimes be obtained from analysis of a mathematical definition of the reaction behaviour, use of the `<reaction>` element makes it possible for processing software to easily obtain enough information to produce reaction equations and pathway diagrams for a model.

The `<reaction>` element may contain a `reversible` attribute to indicate that the reaction defined may proceed in both directions; in this example, the value is "yes". In the simple reaction model, the `<reaction>` element contains five `<variable_ref>` elements. The first four each reference a species variable and the fifth references the variable that is used to calculate the reaction rate. A variable is referenced using the `variable` attribute, the value of which must match the name of a variable declared in the current component. Each `<variable_ref>` element may contain one or more `<role>` elements. The `role` attribute of each `<role>` element declares the way in which the referenced variable participates in the reaction. In this example, species **A** and **B** have a role of "reactant", and species **C** and **D** have a role of "product". Species may also assume roles of "catalyst", "activator", "inhibitor", and "modifier". Note that a single species (represented by a single variable) may participate within the same reaction in numerous roles, e.g., a reactant may also be an inhibitor. This is specified by creating more than one `<role>` element within the same `<variable_ref>` element.

The value of the `role` attribute specifies the participation of the species when the reaction is proceeding in the direction specified by the value of the `direction` attribute. The `direction` attribute may take values of "forward", "backward", and "both". Its value defaults to "forward" if omitted (note that the `direction` attribute could have been omitted on the role elements in this example). The fact that the reaction is reversible implies that a species that is a reactant in the forward direction is a product in the backward direction and that a species that is a product in the forward direction is a reactant in the backward direction. No such assumptions are made about species that act in other roles.

The `delta_variable` attribute on the `<role>` elements is used to associate a delta variable with the principle variable referenced in the parent `<variable_ref>` element. The delta variable represents the change in the concentration of the principle variable due to its involvement in the current reaction. A `delta_variable` attribute may only appear on `<role>` elements with a `role` attribute value of "reactant" or "product", as these are the only forms of reaction participation where the concentration of the participating species is changed. In the simple example, the variable `delta_A` is declared to be the delta variable associated with variable **A**.

The `stoichiometry` attribute on the `<role>` element defines the stoichiometry of the current variable relative to the other reaction participants. In this example the `stoichiometry` attributes on each of the `<role>` elements allow us to form the chemical expression for the reaction. Different values of

stoichiometry may be defined on different **<role>** elements within the same variable reference, allowing for instance, a species to participate as a reactant with one stoichiometry and as an inhibitor with another.

When both **delta_variable** and **stoichiometry** attributes are defined on the same **<role>** element, it implicitly defines an equation relating the delta variable to the reaction rate.

- For reactants: $\text{delta_variable} = (\text{stoichiometry})(\text{rate})$
- For products: $\text{delta_variable} = -(\text{stoichiometry})(\text{rate})$

The reaction rate variable is indicated by a **<variable_ref>** element containing a **<role>** element with a **role** attribute value of **"rate"** — in this example that variable is **r**. The implied equation equates the delta variable with the product of the stoichiometry and rate, producing the following equation:

$$\text{delta}_A = 1.0 \cdot r \tag{1}$$

This is equivalent to the block of MathML shown in Figure 6.

```
<math xmlns="http://www.w3.org/1998/Math/MathML" xmlns:cellml="bob">
  <apply><eq />
    <ci>delta_A </ci>
    <apply><times />
      <cn cellml:units="dimensionless">1.0 </cn>
      <ci>r </ci>
    </apply>
  </apply>
</math>
```

Figure 6. The role element with **delta_variable** and **stoichiometry** attribute values of **"delta_A"** and **"1"** respectively implicitly defines the equation above. The rate variable **r** is the variable referenced with a role of **"rate"**.

If math is to be implicitly defined using the scheme just described, then the **<reaction>** element must contain a variable reference which assigns a role of **"rate"** to one of the participants. This variable may participate in the reaction in no other way, and a **<role>** element with a **role** attribute value of **"rate"** must not contain **direction**, **delta_variable**, or **stoichiometry** attributes. It is recommended that the equation or system of equations used to calculate the value of the rate variable be placed inside the **<role>** element for software or readers to find the relevant equation(s). These mathematics, which were omitted from Figure 5, is shown in Figure 7.

```
<component name="basic_reaction" xmlns:cellml="bob">
```

...

`<reaction reversible="yes">`

...

`<variable_ref variable="r">``<role role="rate">``<math xmlns="http://www.w3.org/1998/Math/MathML">``<apply><eq />``<ci> r </ci>``<apply><plus />``<apply><minus />``<apply><times />``<ci> k_forward </ci>``<ci> A </ci>``<ci> B </ci>``</apply>``</apply>``<apply><times />``<ci> k_reverse </ci>``<apply><power />``<ci> C </ci>``<cn cellml:units="dimensionless"> 2.0 </cn>``</apply>``<ci> D </ci>``</apply>``</apply>``</apply>``</math>`

```

    </role>

    </variable_ref>

  </reaction>

</component>

```

Figure 7. The value of the reaction rate is defined inside a `<role>` element with a `role` attribute value of `"rate"`, inside a `<variable_ref>` element that references the variable `r`.

The possible values of the `role` attribute are based primarily on the information needed for producing a pathway diagram representation of the model — further information must be obtained from the mathematics. If the mathematics are omitted, it is possible to define qualitative models using nothing but `<reaction>` elements. If a modeller erroneously defines a model in which the information in a `<reaction>` element contradicts the information in the mathematics, the CellML specification instructs processing software to give precedence to the information in the mathematics when *running the model*. However, it is left to the application to decide how to handle the *rendering* of the model. Therefore, modellers should be very careful when writing or editing CellML documents by hand. It is anticipated that CellML authoring software will contain constraints that prevent modellers from incorporating inconsistent information in `<math>` and `<reaction>` elements.

To learn more about the reaction element, see Section 7 [[../.../public/specification/20010518/reactions.html](http://public.specification/20010518/reactions.html)] of the CellML specification.

Connections

The simple reaction model contains a total of eight connections, which define the mappings between the variables of different components. The definition of the connection between the `A` species component and the `basic_reaction` component is shown in Figure 8. Only one connection may exist between any two components in the model. Therefore, all variable mappings between these two components are defined in a single connection.

Each `<connection>` element must contain a single `<map_components>` element, which references (in any order) two components contained in the current `<model>` element. The variables that are being mapped between the two components are listed successively in separate `<map_variables>` elements. The value of the `variable_1` attribute must equal the value of the `name` attribute on a `<variable>` element contained in the `<component>` referenced by the `component_1` attribute on the `<map_component>` element, and the value of the `variable_2` attribute must correspond to a variable in `component_2`. The mappings must conform to the rules laid out in Section 3.2.4 [[../.../public/specification/20010518/model_structure.html#sec_structure_connection_element](http://public.specification/20010518/model_structure.html#sec_structure_connection_element)] of the CellML specification. In this example the value of the variable `A` is being passed from component `A` to the `basic_reaction` component, and the value of the variable `delta_A` is being passed in the opposite direction.

```

<connection>

  <map_components component_1="A" component_2="basic_reaction">

    <map_variables variable_1="A" variable_2="A" />

    <map_variables variable_1="delta_A" variable_2="delta_A" />

  </map_components>

</connection>

```

Figure 8. The definition of the connection between the **A** species component and the **basic_reaction** component, with the associated variable mappings.

Since variable names only need to be unique across a component, and not across the entire model, we can give the input concentration variables in the reaction component the same names as the corresponding output variables in the chemical species components, and the output change in concentration variables in the reaction component the same names as the corresponding input variables in the chemical species components. Note, however, that simply giving the variables the same name does **not** specify their connection. We must do that explicitly in the **<connection>** elements.

Download This Model

The CellML description of the model that this documentation discusses is available in a number of formats. If you have your browser set up to view text files served with the “text/xml” MIME type, then you can have a look at the XML file directly here [../../models/basic_reaction_model.xml]. If not, you can save the target of that link to disk by shift-clicking on the preceding link. A “pretty-printed” browsable HTML version of the XML file is available here [../../models/basic_reaction_model.html] — note that you cannot download and save this version for later viewing since it makes use of stylesheets for formatting. If you wish to save or print out the “pretty-printed” version of the XML, a PDF version is also available here [../../models/basic_reaction_model.pdf]. Finally a gzipped tarball (the Unix equivalent of a winzip file) with this documentation, the raw XML and the pretty-printed PDF version of the XML is available here [../../downloads/cellml_basic_reaction_model.tar.gz].

Here are those links again:

- [basic_reaction_model.xml \[../../models/basic_reaction_model.xml\]](#) — the raw XML.
- [basic_reaction_model.html \[../../models/basic_reaction_model.html\]](#) — an HTML version for browsing online.
- [basic_reaction_model.pdf \[../../models/basic_reaction_model.pdf\]](#) — a PDF version suitable for printing.
- [cellml_basic_reaction_model.tar.gz \[../../downloads/cellml_basic_reaction_model.tar.gz\]](#) — a gzipped tarball with the XML and this documentation.
- [basic_reaction_model_maths.pdf \[../../maths_pdf/basic_reaction_model_maths.pdf\]](#) — a PDF of the equations described in the model generated directly from the CellML description using the MathML Renderer [../../public/tools/index.html].