Qualitative Models

Claudine Chaouiya  
chaouiya@igc.gulbenkian.pt  
IGC Rua da Quinta Grande 6  
P-2780-156 Oeiras  
Portugal

Sarah M Keating  
skeating@ebi.ac.uk  
European Bioinformatics Institute  
Cambridgeshire  
UK

Duncan Berenguier  
TAGC INSERM U928  
13288 Marseille  
France

Aurélien Naldi  
Center for Integrative Genomics  
CH-1015 Lausanne  
Switzerland

Denis Thieffry  
IBENS  
75005 Paris  
France

Martijn P. van Iersel  
European Bioinformatics Institute  
Cambridgeshire  
UK

Tomáš Helikar  
Department of Mathematics  
University of Nebraska Medical Center  
US

Version 1 Release 1

17th May 2013

The latest release, past releases, and other materials related to this specification are available at  
http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/Qualitative_Models_(qual)

This release of the specification is available at  
http://identifiers.org/combine.specifications/sbml.level-3.version-1.qual.version-1.release-1
1 Introduction

1.1 Motivation

Quantitative methods for modelling biological networks require an in-depth knowledge of the biochemical reactions and their stoichiometric and kinetic parameters. In many practical cases, this knowledge is missing. This has led to the development of several qualitative modelling methods using information such as gene expression data coming from functional genomic experiments.

The qualitative models contemplated in this package are essentially based on the definition of regulatory or influence graphs. The components of these models differ from the species and reactions used in current SBML models. For example, qualitative models typically associate discrete levels of activities with entity pools; the processes involving them cannot be described as reactions per se but rather as transitions between states. These systems can be viewed as reactive systems, whose dynamics are represented by means of state transition graphs (or other Kripke structures (see Clarke et al. 1999; Kripke 1963) representing, in the form of a graph, which nodes are the reachable states and in which the edges are the state transitions). In this context, logical regulatory networks (Boolean or multi-valued; Kauffman 1969; Thomas 1991) and standard Petri nets (Chaouiya, 2007) are the two formalisms mostly used in biology that give rise to such behaviours. Published models using these approaches cover, far from exhaustively, gene regulatory networks, signalling pathways (e.g. Albert and Othmer, 2003; Calzone et al., 2010; Fauré et al., 2006; Helikar et al., 2008; Mendoza and Xenarios, 2006; Naldi et al., 2010; Sánchez and Thieffry, 2003; Thieffry and Thomas, 1995), and metabolic pathways (see review by Chaouiya, 2007).

Finally, because their dynamics can be abstracted by Kripke structures, models expressed as systems of piece-wise linear differential equations (Batt et al., 2005) may be covered by this package, provided some extensions are included. Specific classes of high-level Petri nets may also be contemplated in the future (see ?? on page ??).

Despite differences from traditional SBML models, it is desirable to bring these classes of models under a common format scheme. The purpose of this Qualitative Models package for SBML Level 3 is to support encoding qualitative models in SBML.

1.2 Package dependencies

The QualitativeModels package has no dependencies on other SBML Level 3 packages. (If you find incompatibilities with other packages, please contact the Package Working Group. Contact information is shown on the front page of this document.)

1.3 Document conventions

Following the precedent set by the SBML Level 3 Core specification document (Hucka et al., 2010), we use UML 1.0 (Unified Modeling Language; Eriksson and Penker 1998; Oestereich 1999) class diagram notation to define the constructs provided by this package. We also use colour in the diagrams to carry additional information for the benefit of those viewing the document on media that can display color. The following are the colors we use and what they represent:

- **Black:** Items colored black in the UML diagrams are components taken unchanged from their definition in the SBML Level 3 Core specification document.

- **Green:** Items colored green are components that exist in SBML Level 3 Core, but are extended by this package. Class boxes are also drawn with dashed lines to further distinguish them.

- **Blue:** Items colored blue are new components introduced in this package specification. They have no equivalent in the SBML Level 3 Core specification.

We also use the following typographical conventions to distinguish the names of objects and data types from other entities; these conventions are identical to the conventions used in the SBML Level 3 Core specification document:
**AbstractClass**: Abstract classes are never instantiated directly, but rather serve as parents of other classes. Their names begin with a capital letter and they are printed in a slanted, bold, sans-serif typeface. In electronic document formats, the class names defined within this document are also hyperlinked to their definitions; clicking on these items will, given appropriate software, switch the view to the section in this document containing the definition of that class. (However, for classes that are unchanged from their definitions in SBML Level 3 Core, the class names are not hyperlinked because they are not defined within this document.)

**Class**: Names of ordinary (concrete) classes begin with a capital letter and are printed in an upright, bold, sans-serif typeface. In electronic document formats, the class names are also hyperlinked to their definitions in this specification document. (However, as in the previous case, class names are not hyperlinked if they are for classes that are unchanged from their definitions in the SBML Level 3 Core specification.)

**SomeThing, otherThing**: Attributes of classes, data type names, literal XML, and generally all tokens other than SBML UML class names, are printed in an upright typewriter typeface. Primitive types defined by SBML begin with a capital letter; SBML also makes use of primitive types defined by XML Schema 1.0 (Biron and Malhotra, 2000; Fallside, 2000; Thompson et al., 2000), but unfortunately, XML Schema does not follow any capitalization convention and primitive types drawn from the XML Schema language may or may not start with a capital letter.

For other matters involving the use of UML and XML, we follow the conventions used in the SBML Level 3 Core specification document.
2 Background and context

It is possible to represent some qualitative models using SBML Level 2 or indeed SBML Level 3 Core. However, after several attempts, experience showed that the possible confusion caused by the presence of irrelevant attributes and the need to reinterpret the semantics of some SBML elements could lead to ambiguity. At this point the decision was made to develop an SBML Level 3 package that captured the nature of qualitative models.

A first proposal was written in August 2008 by Duncan Berenguier and Nicolas Le Novère and discussed during a dedicated meeting on the 12th and 13th of August 2008. This meeting brought together a number of people who specialised in qualitative modelling. A summary of the meeting is available at http://www.ebi.ac.uk/compneur/xwiki/bin/view/SBML/L3F which also provides a link to the revised proposal document that was produced as a result of this meeting.

A secondary, but very valuable, outcome of this meeting was the formation of the Common Logical Modelling Toolbox (CoLoMoTo) community; a community that focuses on logical modelling but who are committed to making their models exchangeable and reusable as widely as possible. This small focussed community then took control of developing the SBML L3 Qualitative Models package.

The first CoLoMoTo meeting was held at Oeiras, Portugal in November 2010 (see http://compbio.igc.gulbenkian.pt/nmd/node/30, for the program and participants). A revised version of the proposal was discussed and a formal SBML L3 proposal document was written and circulated as a result of these and other discussions. This document is available at http://sbml.org/images/6/61/SBML-L3-qual-proposal_2.1.pdf.

The proposal was voted on and accepted by the SBML community (June 2011) and a dedicated discussion list set up (https://lists.sourceforge.net/lists/listinfo/sbml-qual). The package was presented at COMBINE 2011.

A second CoLoMoTo meeting took place in March 2012 (see http://co.mbine.org/colomoto/meetings/2012). During this meeting it was decided that there were parts of the proposal that had been introduced in anticipation of the future development of models. Whilst these are valuable aspects of the proposal there is no software supporting these features as yet. It was therefore decided to remove these features from a version 1 specification and reconsider them in the future for subsequent versions of the Qualitative Models package. A summary of these features is given in Section B on page 34 (appendix) of this document.
3 Package syntax and semantics

In this section, we define the syntax and semantics of the Qualitative Models package for SBML Level 3 Version 1. We expound on the various data types and constructs defined in this package, then in Section 4 on page 16, we provide complete examples of using the constructs in example SBML models.

3.1 Namespace URI and other declarations necessary for using this package

Every SBML Level 3 package is identified uniquely by an XML namespace URI. For an SBML document to be able to use a given SBML Level 3 package, it must declare the use of that package by referencing its URI. The following is the namespace URI for this version of the Qualitative Models package for SBML Level 3 Version 1:


In addition, SBML documents using a given package must indicate whether understanding the package is required for complete mathematical interpretation of a model, or whether the package is optional. This is done using the attribute required on the <sbml> element in the SBML document. For the Qualitative Models package, if the model contains any mathematical constructs from the package, i.e. Transitions, the value of this attribute must be set to “true”.

The following fragment illustrates the beginning of a typical SBML model using SBML Level 3 Version 1 and this version of the Qualitative Models package:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
xmlns:qual="http://www.sbml.org/sbml/level3/version1/qual/version1" qual:required="true">
```

3.2 Primitive data types

Section 3.1 of the SBML Level 3 specification defines a number of primitive data types and also uses a number of XML Schema 1.0 data types (Biron and Malhotra, 2000). We assume and use some of them in the rest of this specification, specifically boolean, ID, SID, SIDRef, and string. The Qualitative Model package defines other primitive types; they are described below.

3.2.1 Type sign

The sign is an enumeration of values used to indicate direction of an Input within the system. The possible values are positive, negative, dual and unknown.

3.2.2 Type transitionInputEffect

The transitionInputEffect is an enumeration of values used to indicate the effect of an Input Transition within the system. The possible values are none and consumption.

3.2.3 Type transitionOutputEffect

The transitionOutputEffect is an enumeration of values used to indicate the effect of an Output Transition within the system. The possible values are production and assignmentLevel.
3.3 Qualitative modelling

Before describing the classes and their attributes that have been used by this Qualitative Models Specification it is worth clarifying the intended meaning of some of the terms used.

3.3.1 Levels

The entities being modelled have a level associated with them that indicates the current state of the entity. A level is an integer and takes values that range from “0” up to and including a maximum.

In future versions of the Qualitative Modelling specification, it is intended to introduce a means of specifying symbols to represent any value that might be appropriate in the model (see Section B on page 34).

3.3.2 Transitions

Qualitative Models consider transitions that alter the levels of entities involved in the model, depending on the level of some other entities. This may involve the level of an entity being increased or decreased by a fixed amount; the level remaining unchanged; or the level being reassigned to an alternate value. Transitions occur when a set of conditions is met. These conditions may involve the levels falling above or below a given threshold.

A simple example of this is the case where there are two entities A and B and the model states that when the level of A exceeds “1” (the threshold), the level of B is increased by “1”.

3.3.3 Function terms

The resulting value of an entity affected by a transition may have several possibilities that are governed by a number of conditions. Each transition can have a list of conditional functions functionTerms, each associated with a result that allow the user to specify sets of piecewise conditions. For example a model may wish to encode the following

\[ B = \begin{cases} 
B + 1 & \text{if } A < 1 \\
B & \text{if } 1 \leq A < 3 \\
B + 2 & \text{otherwise}
\end{cases} \]

In this case the Transition would have a FunctionTerm for each of the first two conditions and a DefaultTerm for the otherwise component.

3.3.4 Interpretation of time

Transitions occur when a set of conditions are met. This specification assumes that these conditions are not dependent on time and can occur at any arbitrary time point. Thus the use of any math that explicitly involves time (e.g. the csymbol time or delay) is not recommended. It is anticipated that future versions will consider time issues see Section B on page 34.

3.3.5 Hybrid models

It is noted in Section B on page 34 that this specification does not facilitate the use of SBML constructs outside the scope of this package within a particular model. This is an aspect of modelling that will be addresses in future versions.

3.4 The extended Model class

The extension of SBML Level 3 Core’s Model class is relatively straightforward: the Qualitative Models Package adds two lists, one for holding qualitativeSpecies (listOfQualitativeSpecies, of class ListOfQualitativeSpecies), and the other for holding transitions (listOfTransitions, of class ListOfTransitions). Figure 1 on the following page provides the UML diagram.
The **Model** element may contain at most one **ListOfQualitativeSpecies**, which must contain at least one **QualitativeSpecies**. It may also contain at most one **ListOfTransitions** which must contain at least one **Transition**. The **QualitativeSpecies** class and the **Transition** class are defined in Section 3.5 and Section 3.6 on page 10 respectively.

![Diagram](image)

**Figure 1**: The definitions of the extended **Model** class. In other respects, **Model** remains defined as in the SBML Level 3 Core specification.

### 3.5 The QualitativeSpecies class

Similarly to the **Species** in SBML, the components of qualitative models refer to pools of entities that are considered indistinguishable and are each located in a specific **Compartment**. However, here components are characterised by their qualitative influences rather than by taking part in reactions. Therefore, we define the **QualitativeSpecies** element to represent such pools of entities.

In a Petri net, *qualitative species* refer to the places of the model, while in a logical model, they refer to the variables of this model (i.e. nodes of the influence graph).

A **QualitativeSpecies** describes a pool of indistinguishable entities in a **Compartment**. It is associated with a **level** (an integer representing e.g. an activity state, or a functional level of concentration, etc.) The **QualitativeSpecies** class is defined in Figure 2 on the next page.

**The id attribute**

The **id** attribute takes a required value of type **SId**. The **id** is used as an identifier for the particular **QualitativeSpecies**. It can be used as a `<ci>` element within math elements included by elements defined within the namespace of the Qualitative Models specification i.e. the **math** element of a **FunctionTerm**, in which case it is interpreted as the **level** of this **QualitativeSpecies**. Note that for SBML Level 3 Version 1 identifiers from a given package cannot be referenced by elements outside that package.

**The name attribute**

A **QualitativeSpecies** also has an optional **name** attribute of type **string**. The **name** attribute should be used in the same manner as on SBML Level 3 Core objects; see Section 3.3.2 of the SBML Level 3 Version 1 Core specification for more information.
Section 3.5 The QualitativeSpecies class

The **compartment** attribute

The required attribute `compartment`, of type `SIdRef`, is used to identify the compartment in which the qualitativeSpecies is located. The attribute's value must be the identifier of an existing `Compartment` object in the model. This attribute is comparable with the `compartment` attribute on the `Species` element.

The **constant** attribute

The required attribute `constant`, of type `boolean`, is used to indicate that the level of the qualitativeSpecies is fixed or can be varied. This attribute is comparable with the `constant` attribute on the `Species` element.

Typically, in a regulatory or influence graph a `QualitativeSpecies` may receive no interaction and if so, would appear only as an `Input` in the model and have the value of the `constant` attribute set to "true". In other influence graphs or in Petri net models a `QualitativeSpecies` may occur as an `Input` whose level is changed by the `Transition` and would have `constant` set to "false". The nature of changes to a `QualitativeSpecies` resulting from a `Transition` is also recorded using the `transitionEffect` attribute on the `Input` (see section Section 3.6.1) and may be set to "none" to indicate there is no change. This duplication of information provides a means of validating the modeller's intent and also allows entities on the borders of a system to be easily identified.

The **initialLevel** attribute

The `initialLevel` is a non-negative integer that defines the initial level of the `QualitativeSpecies` in its `Compartment`. This attribute is optional but cannot exceed the value of the `maxLevel` attribute, if both are set.

The **maxLevel** attribute

The `maxLevel` is a non-negative integer that sets the maximal level of the `QualitativeSpecies`. This attribute is optional but when set, the level of the `QualitativeSpecies` must not exceed this value at any point in a simulation.

In logical models, the `maxLevel` must be coherent with the `resultLevel` values in the function terms defined for the corresponding transition, i.e. the model must not contain a `FunctionTerm` that attempts to set a `level` that exceeds this value.

In Petri nets, this attribute is meant to define place capacities. Hence, a transition is not enabled if the value resulting from its firing would exceed the `maxLevel` of one of its output places. The attribute is not required and even if explicitly stated, the restriction imposed by place capacities in a Petri net model must be encapsulated within the `math` element of the `FunctionTerm` elements.

This attribute can also be used to indicate the range of possible levels for a `QualitativeSpecies` whose `constant` attribute is true. This may seem a little contradictory, since if the `constant` attribute is true then the level associated with the `QualitativeSpecies` cannot vary. However, it provides additional information regarding the possible levels particularly in the case where no `initialLevel` has been set.

![Figure 2: The definitions of the QualitativeSpecies class.](image)
3.6 The Transition class

A **Transition** element contains at most one **ListOfInputs** and one **ListOfOutputs** and exactly one **ListOfFunctionTerms**. These objects classes are defined in **Figure 3**.

A **Transition** defines the changes in *level* associated with the **QualitativeSpecies** that occur when a **Transition** is enabled.
In logical models a **Transition** is used to specify the logical rule associated with a **QualitativeSpecies** (that appears as an **Output** of this **Transition**). For example, the rule \( A > 1 : B = 2 \) would be encapsulated as a **Transition** with **QualitativeSpecies** "A" as an **Input** and "B" as an **Output**; the \( A > 1 \) rule being encode by the **math** element of a **FunctionTerm** with the **resultLevel** attribute having a value "2".

In Petri net models a **Transition** is interpreted, using the common Petri net semantics, as events that might occur within the system causing tokens to be moved. The example in Section 4.2 on page 19 illustrates a simple Petri net model with two input places, two output places and one transition.

**The id attribute**

A **Transition** element has an optional **id** attribute of type **SId**. In constrast to most SBML classes the **id** attribute on a **Transition** has no mathematical interpretation.

**The name attribute**

There is an optional **name** attribute of type **string** that should be used in the same manner as on SBML Level 3 Core objects; see Section 3.3.2 of the SBML Level 3 Version 1 Core specification for more information.

### 3.6.1 The Input class

The **ListOfInputs** contains at least one element of type **Input**. Each **Input** refers to a **QualitativeSpecies** that participates in the corresponding **Transition**. In Petri nets, these are the input places of the transition. In logical models, they are the regulators of the species whose behaviour is defined by the transition.

**The id attribute**

An **Input** element has an optional **id** attribute of type **SId**. The identifier of an **Input** can be used as a <ci> element within math elements included by elements defined within the namespace of the Qualitative Models specification i.e. the **math** element of a **FunctionTerm**, in which case it it interpreted as the **thresholdLevel** of this **Input**. Note that for SBML Level 3 Version 1 identifiers from a given package cannot be referenced by elements outside that package.

**The name attribute**

There is an optional **name** attribute of type **string** that should be used in the same manner as on SBML Level 3 Core objects; see Section 3.3.2 of the SBML Level 3 Version 1 Core specification for more information.

**The qualitativeSpecies attribute**

The required attribute **qualitativeSpecies**, of type **SIdRef**, is used to identify the **QualitativeSpecies** that is the **input** of this **Transition**. The attribute's value must be the identifier of an existing **QualitativeSpecies** object in the model. This attribute is comparable with the **species** attribute on the **SpeciesReference** element.

**The thresholdLevel attribute**

The **thresholdLevel** is a non-negative **integer** that can be used to set the threshold level of the particular input. This attribute relates to the contribution of this input required for the transition to take place. In logical regulatory models, it refers to the threshold level that must be reached or exceeded in order for the regulation to take place, while in a Petri net, it refers to the number of tokens required to enable the transition (weight of the arc connecting the input place to the transition). Whether the **level** of a **QualitativeSpecies** should reach or exceed the **thresholdLevel** in order for the **Transition** to occur will be encoded in the **math** elements of the FunctionTerms listed for the given **Transition**.

The **thresholdLevel** is used by the FunctionTerms associated with the containing **Transition** to determine the applicable **resultLevel** that should be applied. The **id** of the **Input** represents this value and can be used in the **math** element of a **FunctionTerm**. When defined, this attribute should be coherent with the content of the **FunctionTerm**, i.e. if a **number** is used in the **FunctionTerm** to compare the current **level** of a species, this number...
must correspond to the `thresholdLevel` of the corresponding `Input`. Since a `number` can be used within the `FunctionTerm` to represent the `thresholdLevel` of an `Input` it is not compulsory to use this attribute to specify the value. A missing `thresholdLevel` attribute merely implies that the threshold is incorporated into the `FunctionTerm` using a number.

**The `transitionEffect` attribute**

Each `Input` has a required attribute `transitionEffect` of type `transitionInputEffect` which describes how the `QualitativeSpecies` referenced by the `Input` is affected by the `Transition`. Table 1 shows the possible values with the interpretation of each value.

<table>
<thead>
<tr>
<th>TransitionInputEffect</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>The level associated with the <code>qualitativeSpecies</code> is not modified.</td>
</tr>
<tr>
<td>consumption</td>
<td>The level of the <code>qualitativeSpecies</code> is decreased by the <code>resultLevel</code> of the applicable <code>FunctionTerm</code> possibly modified by the <code>thresholdLevel</code> of the <code>Input</code>.</td>
</tr>
</tbody>
</table>

*Table 1: Interpretation of the `transitionEffect` attribute on an `Input`. Note: as discussed in Section 3.6.3 on page 14 the 'applicable `FunctionTerm`' refers to whichever `FunctionTerm` in the `ListOfFunctionTerms` evaluates to "true" or the `DefaultTerm` if all of the `FunctionTerm` objects evaluate to "false".*

The following example illustrates the interpretation of the `transitionEffect` attribute.

```xml
<listOfInputs>
  <input qualitativeSpecies="A" transitionEffect="none" thresholdLevel="2" />
  <input qualitativeSpecies="B" transitionEffect="consumption" />
  <input qualitativeSpecies="C" transitionEffect="consumption" thresholdLevel="2" />
</listOfInputs>
```

In the case of `qualitativeSpecies"A"` the level is unaltered by the `Transition` and hence the `transitionEffect` attribute is set to "none".

The level of `qualitativeSpecies"B"` is reduced; hence the `transitionEffect` is "consumption". The level is reduced by the value of the `resultLevel` from whichever `FunctionTerm` is applicable (see Section 3.6.5).

Similarly, the level of "C" is also reduced, but on this occasion by 2 (the `thresholdLevel`) times the `resultLevel` of whichever `FunctionTerm` is applicable.

It should be noted that in logical models the `transitionEffect` is always set to "none", while in Petri nets, it can be set to "none" (indicating a read arc) or to "consumption". The Petri net example in Section 4 provides a further example of the use of the `transitionEffect` and `thresholdLevel` attributes.

**The `sign` attribute**

The `sign` of type `sign` can be used as an indication as to whether the contribution of this input is positive, negative, both (dual) or unknown. This enables a model to distinguish between stimulation and inhibition and can facilitate interpretation of the model without the mathematics. The sign is particularly used for visualization purposes and has no impact on the mathematical interpretation. This attribute is optional.

### 3.6.2 The `Output` class

The `ListOfOutputs` contains at least one element of type `Output`.

Each `Output` refers to a `QualitativeSpecies` that participates in (is affected by) the corresponding `Transition`. In Petri net models these are the output places of the transition.
In a logical model, a **QualitativeSpecies** should be referenced in at most one **ListOfOutputs**, (that of the **Transition** defining the evolution of this species). This restriction is discussed in more detail in Section 5. When a **Transition** has several outputs, it is because the referenced species share the same regulators and the same logical rules.

**The id attribute**

An **Output** element has an optional id attribute of type SId. The identifier of an **Output** can be used as a <ci> element within math elements included by elements defined within the namespace of the Qualitative Models specification i.e. the math element of a **FunctionTerm**, in which case it is interpreted as the outputLevel of this **Output**. Note that for SBML Level 3 Version 1 identifiers from a given package cannot be referenced by elements outside that package.

**The name attribute**

There is an optional name attribute of type **string** that should be used in the same manner as on SBML Level 3 Core objects; see Section 3.3.2 of the SBML Level 3 Version 1 Core specification for more information.

**The qualitativeSpecies attribute**

The required attribute **qualitativeSpecies**, of type SIdRef, is used to identify the **QualitativeSpecies** that is the output of this **Transition**. The attribute's value must be the identifier of an existing **QualitativeSpecies** object in the model. This attribute is comparable with the species attribute on the **SpeciesReference** element.

**The outputLevel attribute**

The **outputLevel** is a non-negative integer used along with the transitionEffect to specify the effect of the **Transition** on the corresponding **QualitativeSpecies**. It does not specify the result of a **Transition**; this is done by using the resultLevel attribute on a **FunctionTerm**. However, in Petri nets, it relates to the weight of the arc connecting the transition to the output place and may be multiplied by the resultLevel in a "production" situation. In logical models there is no interpretation of the outputLevel attribute as the outcome of a **Transition** is always an assignment to the resultLevel defined by the **FunctionTerm**.

The outputLevel attribute is optional since if the transitionEffect is set to "assignmentLevel" (as in logical models), it has no meaning. However, where the transitionEffect of the **Output** is set to "production" (as in Petri net models) the resulting level of the **QualitativeSpecies** is the resultLevel from the appropriate **FunctionTerm** multiplied by the outputLevel. Since there are no default values in SBML Level 3, when the transitionEffect is set to "production" the outputLevel attribute must have a value.

**The transitionEffect attribute**

Each **Output** has a required attribute transitionEffect of type **transitionOutputEffect** which describes how the **QualitativeSpecies** referenced by the **Output** is affected by the **Transition**. Table 2 shows the possible values with the interpretation of each value.

<table>
<thead>
<tr>
<th>TransitionOutputEffect</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>production</td>
<td>The level of the qualitativeSpecies is increased by the resultLevel of the applicable FunctionTerm possibly modified by the outputLevel of the Output.</td>
</tr>
<tr>
<td>assignmentLevel</td>
<td>The level of the qualitativeSpecies is set to the resultLevel of the selected term.</td>
</tr>
</tbody>
</table>

**Table 2:** Interpretation of the transitionEffect attribute on an **Output**. Note: as discussed in Section 3.6.3 on the next page the 'applicable FunctionTerm' refers to whichever FunctionTerm in the ListOfFunctionTerms evalutaes to "true" or the DefaultTerm if all of the FunctionTerm objects evalutaes to "false".
The following example illustrates the interpretation of the transitionEffect attribute. In the case of qualitativeSpecies “A” the level is assigned the resultLevel from the whichever FunctionTerm is applicable, whereas the level of qualitativeSpecies “B” is increased by resultLevel (effectively resultLevel times 1 (outputLevel)). Similarly, the level of “C” is increased by 2 (outputLevel) times resultLevel (see also Petri net example in Section 4).

```xml
<listOfOutputs>
  <output qualitativeSpecies="A" transitionEffect="assignmentLevel"/>
  <output qualitativeSpecies="B" transitionEffect="production" outputLevel="1"/>
  <output qualitativeSpecies="C" transitionEffect="production" outputLevel="2"/>
</listOfOutputs>
```

In logical models the transitionEffect is set to “assignmentLevel” whilst in standard Petri nets it is set to “production”. It is envisioned that to encode High Level Petri nets it will be necessary to allow the use of “assignmentLevel” as an Output transitionEffect; however considering the implications of this is left to future versions of the specification (see Section B on page 34).

### 3.6.3 The ListOfFunctionTerms class

The ListOfFunctionTerms may contain any number of FunctionTerm elements, and must contain exactly one DefaultTerm. Each FunctionTerm encodes the conditions under which this term is selected. The DefaultTerm describes the result of the Transition applied by default (i.e. when no term evaluates to “true”).

### 3.6.4 The DefaultTerm class

The DefaultTerm defines the default result of a Transition. This term is used if there are no other FunctionTerm elements or if none of the Math elements of the FunctionTerm elements evaluates to “true”.

**The resultLevel attribute**

The default result is described by a resultLevel. This attribute is required.

The resultLevel is a non-negative integer describing a level. The resultLevel is used; possibly together with the thresholdLevel or outputLevel to determine the level of a QualitativeSpecies resulting from the Transition.

### 3.6.5 The FunctionTerm class

Each FunctionTerm is also associated with a result and in addition to a Boolean function inside a Math element that can be used to set the conditions under which this term is selected.

**The resultLevel attribute**

The result of the term is described by the required attribute resultLevel.

The resultLevel is a non-negative integer describing a level. The resultLevel is used; possibly together with the thresholdLevel or outputLevel to determine the level of a QualitativeSpecies resulting from the Transition.

**The Math element**

Each FunctionTerm holds a boolean function encoded in a Math element, using the subset of MathML 2.0 as defined in SBML L3v1 Section 3.4.6. Since the concept of time is beyond the scope of this specification it is recommended that the csymbols “time” and “delay” that explicitly involve time are not used.

This element encodes the conditions under which the FunctionTerm is selected. When the Math element contains the identifier of a QualitativeSpecies, Input or Output, this identifier represents the level, thresholdLevel or outputLevel of the corresponding element. It should be noted that for the purposes of this specification these all have integer values. Tools working with Boolean models with allowed levels restricted to “0” and “1” may choose to interpret the identifiers as boolean. However this specification requires that any math element unambiguously
returns a boolean function. Thus, assuming A is an identifier representing a level, the math expression if (A) is not valid and must be explicitly written as if (A == 1) (or similar). Tools may need to consider this when exporting models.

3.6.6 Mathematical interpretation of Transitions and FunctionTerms

In the Qualitative Models package, transitions are the central mechanism for describing processes that change the levels of the qualitative species of the model. Here, we clarify their interpretation in the framework of logical modelling.

The function terms of a Transition define the transition function for one QualitativeSpecies, i.e. its state transitions depend on the levels of the species that appear as input of that transition (its "regulators"). The function terms together with the default term thus define a state transition table indicating what level the qualitative species will move to (target level), based on the current level of its regulators. In the case of multi-valued levels (as opposed to Boolean), this evolution proceeds step-wise towards the target level, i.e. each component of two successor states of the system differ at most by 1. The QualitativeSpecies affected by the Transition is referenced by the Output element. In the situation where there is more than one Output listed, the referenced species share the same regulators and the same logical rules.

The model must be fully defined. Whatever the state of the system, one single value must apply (that of the DefaultTerm or the resultLevel of a FunctionTerm). More than one FunctionTerm can share the same resultLevel, which is the equivalent to a single term holding the disjunction (OR) of all these terms. There must be no conflicting terms: whenever multiple function terms apply (are true), their resultLevel must be the same.

It should be noted that the level associated with a QualitativeSpecies has values from 0 up to the maxLevel (where declared). The mathematics of the model (i.e. the FunctionTerm and DefaultTerm element together with the transitionEffect) should not allow the level to either become negative or exceed the maximum.

Importantly, given a model, one has then to choose an updating policy that defines how enabled transitions are processed (synchronously, asynchronously, etc.). However, this information is not part of the model per se.

3.7 Namespace scoping rules for identifiers

The values of any id attribute of type SId within the qual namespace are considered to have the same scope as any id attribute with type SId in the core SBML namespace. Thus the values of the attributes id and qual:id must be unique across the set of all id and qual:id attribute values of all objects in a model. In addition to those classes of objects specified in the SBML Level 3 Version 1 Core specification; Model, FunctionDefinition, Compartment, Species, Reaction, SpeciesReference, ModifierSpeciesReference, Event, and Parameter objects, this includes the following objects from this Qualitative Modelling package: QualitativeSpecies, Transition, Input and Output.
This proposal mainly covers logical models but it can also handle standard Petri nets. We provide one toy example of each and the logical model of the decision between lysis and lysogenization in temperate bacteriophage as defined by Thieffry and Thomas (1995).

### 4.1 Simple Logical Regulatory Graph

The following example shows a simple LRG with 3 regulators A, B and C, where A can take three values \(A = \{0, 1, 2\}\), and B,C are Boolean. Moreover, A positively regulates B, which positively regulates C, which positively regulates A. In turn A activates itself at level 1 but inhibits itself at a higher level (2) as illustrated by Figure 4.

![Figure 4: A simple Logical Regulatory Network.](image)

The logical functions are the following:

\[
A_{i+1} := \begin{cases} 
2 & \text{if } (1 \leq A_i < 2) \text{ or } ((C_i \geq 1) \text{ and } (A_i \geq 1)) \\
1 & \text{if } (A_i < 1) \text{ and } (C_i \geq 1) \\
0 & \text{otherwise}
\end{cases}
\]

\[
B_{i+1} := \begin{cases} 
1 & \text{if } A_i \geq 1 \\
0 & \text{otherwise}
\end{cases}
\]

\[
C_{i+1} := \begin{cases} 
1 & \text{if } B_i \geq 1 \\
0 & \text{otherwise}
\end{cases}
\]

The state transition tables are thus:

<table>
<thead>
<tr>
<th>(A_i)</th>
<th>(C_i)</th>
<th>(A_{i+1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(A_i)</th>
<th>(B_{i+1})</th>
<th>(B_i)</th>
<th>(C_{i+1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

```xml
<?xml version="1.0" encoding="UTF8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
  xmlns:qual="http://www.sbml.org/sbml/level3/version1/qual/version1" qual:required="true">
  <model id="example">
    <listOfCompartments>
      <compartment id="cytosol" name="cytosol" constant="true"/>
      <compartment id="nucleus" name="nucleus" constant="true"/>
    </listOfCompartments>
  </model>
</sbml>
<qual:listOfQualitativeSpecies>
  <qual:qualitativeSpecies qual:compartment="cytosol" qual:constant="false" qual:id="A" qual:maxLevel="2"/>
  <qual:qualitativeSpecies qual:compartment="cytosol" qual:constant="false" qual:id="B" qual:maxLevel="1"/>
  <qual:qualitativeSpecies qual:compartment="nucleus" qual:constant="false" qual:id="C" qual:maxLevel="1"/>
</qual:listOfQualitativeSpecies>

<qual:listOfTransitions>
  <qual:transition qual:id="tr_B">
    <qual:listOfInputs>
      <qual:input qual:id="theta_B_A" qual:qualitativeSpecies="A" qual:thresholdLevel="1" qual:transitionEffect="none" qual:sign="positive"/>
    </qual:listOfInputs>
    <qual:listOfOutputs>
      <qual:output qual:transitionEffect="assignmentLevel" qual:qualitativeSpecies="B"/>
    </qual:listOfOutputs>
    <qual:listOfFunctionTerms>
      <qual:functionTerm qual:resultLevel="1">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <apply>
            <geq/>
            <ci>A</ci>
            <ci>theta_B_A</ci>
          </apply>
        </math>
      </qual:functionTerm>
      <qual:defaultTerm qual:resultLevel="0"/>
    </qual:listOfFunctionTerms>
  </qual:transition>

  <qual:transition qual:id="tr_A">
    <qual:listOfInputs>
      <qual:input qual:id="theta_A_A1" qual:qualitativeSpecies="A" qual:thresholdLevel="1" qual:transitionEffect="none" qual:sign="positive"/>
      <qual:input qual:id="theta_A_A2" qual:qualitativeSpecies="A" qual:thresholdLevel="2" qual:transitionEffect="none" qual:sign="negative"/>
      <qual:input qual:id="theta_A_C" qual:qualitativeSpecies="C" qual:thresholdLevel="1" qual:transitionEffect="none" qual:sign="positive"/>
    </qual:listOfInputs>
    <qual:listOfOutputs>
      <qual:output qual:qualitativeSpecies="A" qual:transitionEffect="assignmentLevel"/>
    </qual:listOfOutputs>
  </qual:transition>
</qual:listOfTransitions>
<qual:listOfFunctionTerms>
  <qual:functionTerm qual:resultLevel="2">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <!-- (A >= 1 and A < 2) or (C >= 1 and A >= 1) -->
      <apply>
        <or/>
        <apply>
          <and/>
          <apply>
            <geq/>
            <ci>A</ci>
            <ci>theta_A_A1</ci>
          </apply>
          <apply>
            <lt/>
            <ci>A</ci>
            <ci>theta_A_A2</ci>
          </apply>
        </apply>
        <apply>
          <and/>
          <apply>
            <geq/>
            <ci>C</ci>
            <ci>theta_A_C</ci>
          </apply>
        </apply>
      </apply>
    </math>
  </qual:functionTerm>
  <qual:functionTerm qual:resultLevel="1">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <!-- (A < 1) and C >= 1 -->
      <apply>
        <and/>
        <apply>
          <lt/>
          <ci>A</ci>
          <ci>theta_A_A1</ci>
        </apply>
        <apply>
          <geq/>
          <ci>C</ci>
          <ci>theta_A_C</ci>
        </apply>
      </apply>
    </math>
  </qual:functionTerm>
  <qual:defaultTerm qual:resultLevel="0"/>
</qual:listOfFunctionTerms>
4.2 Simple Petri net

The following example shows a simple, standard Petri net, with 4 places A, B, C and D and one transition $t_1$ as depicted in Figure 5.

Figure 5: A Petri net model.
Listing 2: Petri net example
4.3 Logical model of the immunity control in bacteriophage lambda

This last example is the multi-valued, logical model as defined by Thieffry and Thomas (1995) for the core network controlling the decision between lysis and lysogeny in temperate bacteriophage.

![Interaction graph of the four-variable multi-valued model of phage lambda switch (Thieffry and Thomas, 1995).](image)

```xml
<?xml version="1.0" encoding="UTF8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
     xmlns:qual="http://www.sbml.org/sbml/level3/version1/qual/version1" qual:required="true">
  <model id="phage_lambda">
    <listOfCompartments>
      <compartment id="comp1" constant="true"/>
    </listOfCompartments>
    <qual:listOfQualitativeSpecies>
      <qual:qualitativeSpecies qual:id="s_Cl" qual:compartment="comp1" qual:maxLevel="2" qual:constant="false"/>
      <qual:qualitativeSpecies qual:id="s_Cro" qual:compartment="comp1" qual:maxLevel="3" qual:constant="false"/>
      <qual:qualitativeSpecies qual:id="s_Cll" qual:compartment="comp1" qual:maxLevel="1" qual:constant="false"/>
      <qual:qualitativeSpecies qual:id="s_N" qual:compartment="comp1" qual:maxLevel="1" qual:constant="false"/>
    </qual:listOfQualitativeSpecies>
    <qual:listOfTransitions>
      <qual:transition qual:id="tr_CI">
        <qual:listOfInputs>
          <qual:input qual:qualitativeSpecies="s_Cro" qual:sign="negative" qual:transitionEffect="none"/>
          <qual:input qual:qualitativeSpecies="s_Cll" qual:sign="positive" qual:transitionEffect="none"/>
        </qual:listOfInputs>
        <qual:listOfOutputs>
          <qual:output qual:qualitativeSpecies="s_CI" qual:transitionEffect="assignmentLevel"/>
        </qual:listOfOutputs>
      </qual:transition>
    </qual:listOfTransitions>
  </model>
</sbml>
```
Section 4.3  Logical model of the immunity control in bacteriophage lambda

<qual:listOfFunctionTerms>
  <qual:defaultTerm qual:resultLevel="0"/>
  <qual:functionTerm qual:resultLevel="2">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <or/>
        <eq/>
        <apply>
          <eq/>
          <ci>s_Cro</ci>
          <cn type="integer">0</cn>
        </apply>
        <apply>
          <and/>
          <apply>
            <geq/>
            <ci>s_Cro</ci>
            <cn type="integer">1</cn>
          </apply>
          <apply>
            <eq/>
            <ci>s_CII</ci>
            <cn type="integer">1</cn>
          </apply>
        </apply>
      </apply>
    </math>
  </qual:functionTerm>
</qual:listOfFunctionTerms>

<qual:listOfInputs>
  <qual:input qual:qualitativeSpecies="s_Cl" qual:sign="negative" qual:transitionEffect="none"/>
  <qual:input qual:qualitativeSpecies="s_Cro" qual:sign="negative" qual:transitionEffect="none"/>
</qual:listOfInputs>

<qual:listOfOutputs>
  <qual:output qual:qualitativeSpecies="s_Cro" qual:transitionEffect="assignmentLevel"/>
</qual:listOfOutputs>

<qual:listOfFunctionTerms>
  <qual:defaultTerm qual:resultLevel="0"/>
  <qual:functionTerm qual:resultLevel="2">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <and/>
        <leq/>
        <apply>
          <ci>s_CI</ci>
          <cn type="integer">1</cn>
        </apply>
      </apply>
    </math>
  </qual:functionTerm>
</qual:listOfFunctionTerms>
Section 4.3 Logical model of the immunity control in bacteriophage lambda

\[ s_{\text{Cro}} = 3 \]

\[ \text{and/} \]
\[ \text{leq/} \]
\[ s_{\text{CI}} \leq 1 \]
\[ s_{\text{Cro}} = 2 \]

\[ s_{\text{Cll}} \]

\[ \text{and/} \]
\[ \text{leq/} \]
\[ s_{\text{CI}} \leq 1 \]

\[ s_{\text{Cro}} = 2 \]

\[ s_{\text{N}} \]

\[ s_{\text{Cll}} \]

\[ \text{and/} \]
\[ \text{leq/} \]
\[ s_{\text{CI}} \leq 1 \]
Listing 3: Phage lambda switch example
5 Best practices

The aim of this specification is to provide a common basis for modelling several different types of qualitative models. To facilitate this goal, the elements defined have attributes that are optional in some types of models but not in others; or indeed have meaning in some contexts but not in others. Here we outline the two cases that can be modelled using the syntax of this specification.

5.1 Logical Regulatory Networks

In these models a QualitativeSpecies is listed as an Input to a Transition if it acts as an activator/inhibitor or regulator in that Transition and as an Output if the evolution of that QualitativeSpecies is governed by the Transition i.e. it is a regulated species. A QualitativeSpecies that is an Output must have a constant attribute set to “false”.

The maxLevel attribute on the QualitativeSpecies can be set to indicate the possible values that the species can take. For example, a boolean would have a maxLevel of “1” indicating that it could have only values of 0 and 1. The level of a QualitativeSpecies can never exceed this value and a model must not contain a FunctionTerm that assigns a level exceeding the maxLevel.

In these classes of models the transitionEffect of the Input species should be set to “none” as the Input species are not altered by the transition.

The thresholdLevel, when specified, indicates the level at which the species participates in the transition. Any reference to the Input id attribute in a <ci> element within a functionTerm of the transition refers to the value of this thresholdLevel. This provides a means of encoding the statement: A transition occurs when the level of A exceeds the threshold level of B; as A > thresholdB rather than A > 1; which may provide a modeller with additional information about the number.

The sign attribute indicates the type of effect on the Output of the Transition (the regulated species): “positive” (activation), “negative” (inhibition), “dual” (positive or negative depending e.g. on co-factors) or “unknown”. It is optional and mainly used for graphical purposes.

At the time of writing it has been decided a QualitativeSpecies should only be referenced by an Output with a transitionEffect of “assignmentLevel” in at most one Transition. For logical regulatory networks, this applies to any Output. However, it is anticipated that in the future the specification may be expanded and used in high level Petri Net modeling where this restriction would be prohibitive. Thus this is a recommendation rather than a requirement.

Discussions are still ongoing about the possible incoherency of the FunctionTerm elements and the need to avoid cumbersome descriptions. As of the writing of this specification, the following guidelines are recommended to ensure coherent definitions:

- The FunctionTerm elements of all the transitions targeting the same output should be “coherent”: the conditions of two FunctionTerm elements, leading to different effects on the level of the output, should not be fulfilled at the same time (i.e. they should be exclusive).

- If several FunctionTerm elements lead to the same effect on the level of the same output, then the importing tool should consider the disjunction (OR) on the conditions of the terms.

5.2 Petri Nets

In Petri Net models the QualitativeSpecies represent places within the model. Since the initial conditions are part of the model the initialLevel attribute for each QualitativeSpecies must be set.

In order to represent an unbound place the maxLevel attribute of the QualitativeSpecies is left unset.
The transitionEffect of an Input is set to "consumption", unless this input is connected to the transition by a test arc (meaning the transition has no effect on its marking).

The thresholdLevel of an Input indicates the weight of the arc from this place to the transition and is required. It is used to specify the enabling conditions of the transition (and to indicate the number of tokens consumed by the firing of this transition).

In this class of models the sign attribute on an Input should not be defined.

The transitionEffect of an Output is set to "production".

The outputLevel of an Output indicates the weight of the arc from the transition to this place, it should be defined and is interpreted as the number of tokens produced by the firing of this transition.

There are no default values in SBML so the thresholdLevel on an Input and the outputLevel on an Output must always be set to indicate the weight of the arcs.

The place capacity can be specified using the maxLevel attribute on a QualitativeSpecies. However the restriction on the Transition imposed by this capacity must be explicitly encoded into the math element of any FunctionTerm.
A Validation of SBML documents

A.1 Validation and consistency rules

This section summarizes all the conditions that must (or in some cases, at least should) be true of an SBML Level 3 Version 1 model that uses the Qualitative Models package. We use the same conventions as are used in the SBML Level 3 Version 1 Core specification document. In particular, there are different degrees of rule strictness. Formally, the differences are expressed in the statement of a rule: either a rule states that a condition must be true, or a rule states that it should be true. Rules of the former kind are strict SBML validation rules—a model encoded in SBML must conform to all of them in order to be considered valid. Rules of the latter kind are consistency rules. To help highlight these differences, we use the following three symbols next to the rule numbers:

✓ A checked box indicates a requirement for SBML conformance. If a model does not follow this rule, it does not conform to the Qualitative Models specification. (Mnemonic intention behind the choice of symbol: “This must be checked.”)

▲ A triangle indicates a recommendation for model consistency. If a model does not follow this rule, it is not considered strictly invalid as far as the Qualitative Models specification is concerned; however, it indicates that the model contains a physical or conceptual inconsistency. (Mnemonic intention behind the choice of symbol: “This is a cause for warning.”)

★ A star indicates a strong recommendation for good modeling practice. This rule is not strictly a matter of SBML encoding, but the recommendation comes from logical reasoning. As in the previous case, if a model does not follow this rule, it is not strictly considered an invalid SBML encoding. (Mnemonic intention behind the choice of symbol: “You’re a star if you heed this.”)

The validation rules listed in the following subsections are all stated or implied in the rest of this specification document. They are enumerated here for convenience. Unless explicitly stated, all validation rules concern objects and attributes specifically defined in the Qualitative Models package.

For convenience and brevity, we use the shorthand “qual:x” to stand for an attribute or element name x in the namespace for the Qualitative Models package, using the namespace prefix qual. In reality, the prefix string may be different from the literal “qual” used here (and indeed, it can be any valid XML namespace prefix that the modeler or software chooses). We use “qual:x” because it is shorter than to write a full explanation everywhere we refer to an attribute or element in the Qualitative Models package namespace.

General rules about this package

qual-10101 ✓ To conform to the Qualitative Models package specification for SBML Level 3 Version 1, an SBML document must declare the use of the following XML Namespace: “http://www.sbml.org/sbml/level3/version1/qual/version1”. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.1 on page 6.)

qual-10102 ✓ Wherever they appear in an SBML document, elements and attributes from the Qualitative Models package must be declared either implicitly or explicitly to be in the XML namespace “http://www.sbml.org/sbml/level3/version1/qual/version1”. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.1 on page 6.)

General rules for MathML content

qual-10201 ▲ The MathML math element in a FunctionTerm object should evaluate to a value of type boolean. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.5 on page 14.)
### Section A.1 Validation and consistency rules

**qual-10202** ▲ The MathML `math` element in a `FunctionTerm` object should not use the `csymbol` elements “time” and “delay” as these explicitly introduce time into the model. As yet time is not considered within the Qualitative Models package specification. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.5 on page 14.)

### General rules about identifiers

**qual-10301** ✓ (Extends validation rule #10301 in the SBML Level 3 Version 1 Core specification.) Within a `Model` the values of the attributes `id` and `qual:id` on every instance of the following classes of objects must be unique across the set of all `id` and `qual:id` attribute values of all such objects in a model: the `Model` itself, plus all contained `FunctionDefinition`, `Compartment`, `Species`, `Reaction`, `SpeciesReference`, `ModifierSpeciesReference`, `Event`, and `Parameter` objects, plus the `QualitativeSpecies`, `Transition`, `Input` and `Output` objects defined by the Qualitative Models package. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.7 on page 15.)

### Rules for the extended SBML class

**qual-20101** ✓ In all SBML documents using the Qualitative Models package, the `SBML` object must include a value for the attribute `qual:required` attribute. (References: SBML Level 3 Version 1 Core, Section 4.1.2.)

**qual-20102** ✓ The value of attribute `qual:required` on the `SBML` object must be of the data type `boolean`. (References: SBML Level 3 Version 1 Core, Section 4.1.2.)

**qual-20103** ✓ The value of attribute `qual:required` on the `SBML` object must be set to “true” if the model contains any `Transition` objects. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.1 on page 6.)

### Rules for extended Model object

**qual-20201** ✓ There may be at most one instance of each of the following kinds of objects within a `Model` object using Qualitative Models: `ListOfTransitions` and `ListOfQualitativeSpecies`. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.4 on page 7.)

**qual-20202** ✓ The various `ListOf__` subobjects with an `Model` object are optional, but if present, these container object must not be empty. Specifically, if any of the following classes of objects are present on the `Model`, it must not be empty: `ListOfQualitativeSpecies` and `ListOfTransitions`. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.4 on page 7.)

**qual-20203** ✓ Apart from the general notes and annotation subobjects permitted on all SBML objects, a `ListOfTransitions` container object may only contain `Transition` objects. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.4 on page 7.)

**qual-20204** ✓ Apart from the general notes and annotation subobjects permitted on all SBML objects, a `ListOfQualitativeSpecies` container object may only contain `QualitativeSpecies` objects. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.4 on page 7.)

**qual-20205** ✓ A `ListOfQualitativeSpecies` object may have the optional `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Qualitative Models namespace are permitted on a `ListOfQualitativeSpecies` object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.4 on page 7.)
A ListOfTransitions object may have the optional attributes metaid and sboTerm defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Qualitative Models namespace are permitted on a ListOfTransitions object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.4 on page 7.)

**Rules for QualitativeSpecies object**

A QualitativeSpecies object may have the optional SBML Level 3 Core attributes metaid and sboTerm. No other attributes from the SBML Level 3 Core namespaces are permitted on a QualitativeSpecies. (References: SBML Level 3 Version 1 Core, Section 3.2.)

A QualitativeSpecies object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a QualitativeSpecies. (References: SBML Level 3 Version 1 Core, Section 3.2.)

A QualitativeSpecies object must have the required attributes qual:id, qual:compartment and qual:constant, and may have the optional attributes qual:name, qual:initialLevel and qual:maxLevel. No other attributes from the SBML Level 3 Qualitative Models namespace are permitted on a QualitativeSpecies object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.5 on page 8.)

The attribute qual:constant in QualitativeSpecies must be of the data type boolean. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.5 on page 8.)

The attribute qual:name in QualitativeSpecies must be of the data type string. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.5 on page 8.)

The attribute qual:initialLevel in QualitativeSpecies must be of the data type integer. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.5 on page 8.)

The attribute qual:maxLevel in QualitativeSpecies must be of the data type integer. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.5 on page 8.)

The value of the attribute qual:compartment in a QualitativeSpecies object must be the identifier of an existing Compartment object defined in the enclosing Model object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.5 on page 8.)

The value of the attribute qual:initialLevel in a QualitativeSpecies object cannot be greater than the value of the qual:maxLevel attribute for the given QualitativeSpecies object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.5 on page 8.)

A QualitativeSpecies with attribute qual:constant set to true can only be referred to by an Input. It cannot be the subject of an Output in a Transition. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.5 on page 8.)

A QualitativeSpecies that is referenced by an Output with the qual:transitionEffect attribute set to “assignmentLevel” cannot be referenced by any other Output with the same transitionEffect throughout the set of transitions for the containing model. (References: SBML Level 3 Package Specification for QualitativeModels, Version 1 Section 3.6.2 on page 12)

The attribute qual:initialLevel in QualitativeSpecies must not be negative. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.5 on page 8.)
Section A Validation and consistency rules

**Rules for Transition object**

qual-20401 ✓ The attribute `qual:maxLevel` in `QualitativeSpecies` must not be negative. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.5 on page 8.)

qual-20402 ✓ A `Transition` object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a `Transition`. (References: SBML Level 3 Version 1 Core, Section 3.2.)

qual-20403 ✓ A `Transition` object may have the optional attributes `qual:id` and `qual:name`. No other attributes from the SBML Level 3 Qualitative Models namespace are permitted on a `Transition` object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6 on page 10.)

qual-20404 ✓ The attribute `qual:name` in `Transition` must be of the data type `string`. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6 on page 10.)

qual-20405 ✓ A `Transition` must have one and only one instance of the `ListOfFunctionTerms` objects and may have at most one instance of the `ListOfInputs` and `ListOfOutputs` objects from the Qualitative Models namespace. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6 on page 10.)

qual-20406 ✓ The `ListOfInputs` and `ListOfOutputs` subobjects on a `Transition` object are optional, but if present, these container object must not be empty. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6 on page 10.)

qual-20407 ✓ Apart from the general notes and annotation subobjects permitted on all SBML objects, a `ListOfInputs` container object may only contain `Input` objects. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6 on page 10.)

qual-20408 ✓ Apart from the general notes and annotation subobjects permitted on all SBML objects, a `ListOfOutputs` container object may only contain `Output` objects. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6 on page 10.)

qual-20409 ✓ Apart from the general notes and annotation subobjects permitted on all SBML objects, a `ListOfFunctionTerms` container object must contain one and only one `DefaultTerm` object and then may only contain `FunctionTerm` objects. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6 on page 10.)

qual-20410 ✓ A `ListOfInputs` object may have the optional `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Qualitative Models namespace are permitted on a `ListOfInputs` object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6 on page 10.)

qual-20411 ✓ A `ListOfOutputs` object may have the optional attributes `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Qualitative Models namespace are permitted on a `ListOfOutputs` object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6 on page 10.)

qual-20412 ✓ A `ListOfFunctionTerms` object may have the optional attributes `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Qualitative Models namespace are permitted on a `ListOfFunctionTerms` object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6 on page 10.)
Section A.1 Validation and consistency rules

qual-20413 ✓ No element of the ListOfFunctionTerms object may cause the level of a QualitativeSpecies to exceed the value qual:maxLevel attribute. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.6 on page 15.)

qual-20414 ✓ No element of the ListOfFunctionTerms object may cause the level of a QualitativeSpecies to become negative. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.6 on page 15.)

Rules for Input object

qual-20501 ✓ A Input object may have the optional SBML Level 3 Core attributes metaid and sboTerm. No other attributes from the SBML Level 3 Core namespaces are permitted on a Input. (References: SBML Level 3 Version 1 Core, Section 3.2.)

qual-20502 ✓ A Input object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a Input. (References: SBML Level 3 Version 1 Core, Section 3.2.)

qual-20503 ✓ A Input object must have the required attributes qual:qualitativeSpecies as well as qual:transitionEffect, and may have the optional attributes qual:id, qual:name, qual:sign and qual:thresholdLevel. No other attributes from the SBML Level 3 Qualitative Models namespace are permitted on a Input object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.1 on page 11.)

qual-20504 ✓ The attribute qual:name in Input must be of the data type string. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.1 on page 11.)

qual-20505 ✓ The value of the attribute qual:sign of a Input object must conform to the syntax of the SBML data type sign and may only take on the allowed values of sign defined in SBML; that is, the value must be one of the following: “positive”, “negative”, “dual” or “unknown”. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.1 on page 11.)

qual-20506 ✓ The value of the attribute qual:transitionEffect of a Input object must conform to the syntax of the SBML data type transitionInputEffect and may only take on the allowed values of transitionInputEffect defined in SBML; that is, the value must be one of the following: “none” or “consumption”. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.1 on page 11.)

qual-20507 ✓ The attribute qual:thresholdLevel in Input must be of the data type integer. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.1 on page 11.)

qual-20508 ✓ The value of the attribute qual:qualitativeSpecies in an Input object must be the identifier of an existing QualitativeSpecies object defined in the enclosing Model object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.1 on page 11.)

qual-20509 ✓ An Input that refers to a QualitativeSpecies that has a qual:constant attribute set to “true” cannot have the attribute qual:transitionEffect set to “consumption”. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.1 on page 11.)

qual-20510 ✓ The attribute qual:thresholdLevel in Input must not be negative. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.1 on page 11.)

Rules for Output object

qual-20601 ✓ A Output object may have the optional SBML Level 3 Core attributes metaid and sboTerm. No other attributes from the SBML Level 3 Core namespaces are permitted on a Output. (References: SBML Level 3 Version 1 Core, Section 3.2.)
A **Output** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **Output**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

A **Output** object must have the required attributes **qual:qualitativeSpecies** as well as **qual:transitionEffect**, and may have the optional attributes **qual:id**, **qual:name** and **qual:outputLevel**. No other elements from the SBML Level 3 Qualitative Models namespace are permitted on a **Output** object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.2 on page 12.)

The attribute **qual:name** in **Output** must be of the data type **string**. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.2 on page 12.)

The value of the attribute **qual:transitionEffect** of a **Output** object must conform to the syntax of the SBML data type **transitionOutputEffect** and may only take on the allowed values of **transitionOutputEffect** defined in SBML; that is, the value must be one of the following: "production" or "assignmentLevel". (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.2 on page 12.)

The attribute **qual:outputLevel** in **Output** must be of the data type **integer**. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.2 on page 12.)

The value of the attribute **qual:qualitativeSpecies** in an **Output** object must be the identifier of an existing **QualitativeSpecies** object defined in the enclosing **Model** object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.2 on page 12.)

The **QualitativeSpecies** referred to by the attribute **qual:qualitativeSpecies** in an **Output** object must have the value of its **qual:constant** attribute set to **false**. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.2 on page 12.)

When the value of the attribute **qual:transitionEffect** of a **Output** object is set to the value "production" the attribute **qual:outputLevel** for that particular **Output** object must have a value set. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.2 on page 12.)

The attribute **qual:outputLevel** in **Output** must not be negative. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.2 on page 12.)

---

**Rules for DefaultTerm object**

A **DefaultTerm** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespaces are permitted on a **DefaultTerm**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

A **DefaultTerm** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **DefaultTerm**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

A **DefaultTerm** object must have the required attributes **qual:resultLevel**. No other attributes from the SBML Level 3 Qualitative Models namespace are permitted on a **DefaultTerm** object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.4 on page 14.)

The attribute **qual:resultLevel** in **DefaultTerm** must be of the data type **integer**. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.4 on page 14.)
Section A.1 Validation and consistency rules

**Rules for FunctionTerm object**

- **qual-20801 ✓** A FunctionTerm object may have the optional SBML Level 3 Core attributes metaid and sboTerm. No other attributes from the SBML Level 3 Core namespaces are permitted on a FunctionTerm. (References: SBML Level 3 Version 1 Core, Section 3.2.)

- **qual-20802 ✓** A FunctionTerm object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a FunctionTerm. (References: SBML Level 3 Version 1 Core, Section 3.2.)

- **qual-20803 ✓** A FunctionTerm object must have the required attributes qual:resultLevel. No other attributes from the SBML Level 3 Qualitative Models namespace are permitted on a FunctionTerm object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.5 on page 14.)

- **qual-20804 ✓** A FunctionTerm object may contain exactly one MathML qual:math element. No other elements from the SBML Level 3 Qualitative Models namespace are permitted on a FunctionTerm object. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.5 on page 14.)

- **qual-20805 ✓** The attribute qual:resultLevel in FunctionTerm must be of the data type integer. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.5 on page 14.)

- **qual-20806 ✓** The attribute qual:resultLevel in FunctionTerm must not be negative. (References: SBML Level 3 Package Specification for Qualitative Models, Version 1, Section 3.6.5 on page 14.)
B Future directions

To account for qualitative models where parameters are not (all) instantiated, as well as models for which timing constraints are specified, an extension of the current specification was contemplated. Here, we briefly recapitulate the elements and attributes that have been discarded in the current specification, with the intent to fuel discussions on future extensions. We also point out other extensions that may be considered in the near future, since they were evoked while discussing the current specification.

Finally, we briefly comment on the possible use of this qual package to represent other types of Petri net models.

B.1 Symbols

Definition of SymbolicValue

The **QualitativeSpecies** element may contain at most one **ListOfSymbolicValues** that contains zero or more **SymbolicValue**s. An empty list is allowed, and useful for e.g. adding annotations. The **SymbolicValue** element defines a non instantiated parameter. Such symbols may represent the different solutions of piecewise linear differential equations, along with different thresholds.

*The id and name attributes*

These attributes are used according to the SBML L3.1 Section 3.3. The attribute id is mandatory and name is optional.

*The rank attribute*

The rank is an integer that defines the position of the symbol in the **ListOfSymbolicValues**. This attribute is optional.
**The thresholdLevel and thresholdSymbol attributes:**
The thresholdLevel is an **integer** and thresholdSymbol is a **SIdRef**. They are optional and exclusive.

**The resultLevel and resultSymbol attributes:**
The result of the term is described by a resultLevel or a resultSymbol. Both are optional, but one of them must be defined.

**assignmentSymbol**: The symbol associated to the qualitativeSpecies is set to the resultSymbol of the selected term.

### B.2 Temporisation

**The temporisationType attribute:**
The temporisationType is an **enumeration** the “temporisation” of the **Transition**, that is the updating policy associated with the **Transition**. It can be set to **timer**, **priority**, **sustain**, **proportion** or **rate**. This attribute is optional.

**The temporisationValue attribute and the TemporisationMath element:**
The attribute temporisationValue and the element **TemporisationMath** allow the specification of the “temporisation” of the **Transition** under the corresponding **FunctionTerm**. Both are optional. Depending on the value of the temporisationType, either one or both could be used.

The temporisationValue is a **double**. The element **TemporisationMath** holds a MathML function returning a **double**.

### B.3 Classes of models and random models

Several comments indicate that a future extension could support the representation of classes of models (**i.e.** models that are not fully parametrised, meaning that e.g. the logical rule of a component is incomplete), or random models, e.g. where several logical rules are associated to a component, the choice of a rule in the course of the dynamical evolution being arbitrary. This might be done by revising the requirement of certain elements as well as the current semantics of function terms.

### B.4 Interaction with SBML Core concepts

At the time at which this Qualitative Modelling specification was developed, the policy and process for interacting with SBML Level 3 Core constructs was undefined. Thus, this particular specification does not facilitate the use of Core constructs. It is anticipated that in the future the specification will be extended to allow the use of these constructs; in particular Parameters and Events.

### B.5 Petri net models

The current specification covers the needs of standard Petri Nets but provides a base for expansion to encompass more sophisticated modelling in this area. Both High Level Petri Nets (also referred to as Coloured PetriNets) and Timed PN should be achievable in future versions.

We note that most PN models currently refer to metabolic or other reaction networks, and PN tools dedicated to this modelling framework often provides support for the SBML core format (see, e.g., Rohr et al., 2010).
Figure 8: Possible future extensions of the Transition class.
Acknowledgments

The development of the qual SBML package has been made possible through the organization of three meetings. These were mainly supported by the EMBL-EBI, Cambridge, UK and by the IGC, Portugal. It has been boosted by the path2models projects sponsored by EMBL-EBI. Finally the authors would like to thank the members of the sbml-qual Package Working Group, and in particular F. Büchel, A. Dräger, A. von Kamp, S. Klamt, N. Le Novère, F. Mittag, N. Rodriguez, J. Saez-Rodriguez and I. Xenarios who have also collaborated in this proposal.


