SBML Level 3 Package Specification

SBML Level 3 Package: Flux Balance Constraints ('fbc')

Brett G. Olivier
b.g.olivier@vu.nl
Systems Bioinformatics
VU University Amsterdam
Amsterdam, NH, The Netherlands

Frank T. Bergmann
fbergmann@caltech.edu
Computing and Mathematical Sciences
California Institute of Technology
Pasadena, CA, US

Version 1, Release 1

February 11, 2013

The latest release, past releases, and other materials related to this specification are available at

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

1 Introduction and motivation  
1.1 Proposal corresponding to this package specification ........................................ 3  
1.2 Tracking number .................................................................................................. 4  
1.3 Package dependencies .......................................................................................... 4  
1.4 Document conventions ......................................................................................... 4  

2 Background  
2.1 Problems with current SBML approaches ......................................................... 5  
2.2 Past work on this problem or similar topics ....................................................... 5  

3 Proposed syntax and semantics  
3.1 Namespace URI and other declarations necessary for using this package ........................................... 6  
3.2 Primitive data types ............................................................................................... 6  
3.2.1 Type FbcType .................................................................................................. 6  
3.2.2 Type FbcOperation ......................................................................................... 6  
3.3 The extended Model class .................................................................................... 7  
3.3.1 The FBC listOfFluxBounds .......................................................................... 7  
3.3.2 The FBC listOfObjectives ........................................................................... 7  
3.3.3 A note on units ............................................................................................... 8  
3.4 The extended Species class .................................................................................. 8  
3.5 The FBC FluxBound class .................................................................................... 9  
3.6 The FBC Objective class ...................................................................................... 11  
3.7 The FBC FluxObjective class ............................................................................. 12  

4 Illustrative examples of the FBC syntax  
4.1 Example one: the basic FBC syntax ...................................................................... 13  
4.1.1 Kinetic model description ............................................................................ 13  
4.1.2 Capacity constraints .................................................................................... 14  
4.1.3 Objective function ....................................................................................... 14  
4.1.4 Complete worked example .......................................................................... 15  

5 Best practices  
5.1 Examples contrasting the current SBML L2 encoding with L3 and FBC ................. 17  

6 Future development  
6.1 Model annotation ................................................................................................. 20  
6.1.1 Gene association annotation ....................................................................... 20  
6.1.2 A generic annotation mechanism ................................................................ 20  

A Validation of SBML documents  
A.1 Validation and consistency rules ....................................................................... 23  

Acknowledgments ....................................................................................................... 28  
References .................................................................................................................. 29
1 Introduction and motivation

Constraint based modeling is a widely accepted methodology used to analyze and study biological networks on both a small and whole organism (genome) scale. Typically these models are underdetermined and constraint based methods (e.g. linear, quadratic optimization) are used to optimize specific model properties. This is assumed to occur under a defined set of constraints (e.g. stoichiometric, metabolic) and bounds (e.g. thermodynamic, experimental and environmental) on the values that the solution fluxes can obtain.

Perhaps the most well known (and widely used) analysis method is Flux Balance Analysis (FBA; Orth et al., 2010) which is performed on Genome Scale Reconstructions (GSR’s; Oberhardt et al., 2009). Using FBA a target flux is optimized (e.g. maximizing a flux to biomass or minimizing ATP production) while other fluxes can be bounded to simulate a selected growth environment or specific metabolic state.

As constraint based models are generally underdetermined, i.e. few or none of the kinetic rate equations and related parameters are known, it is crucial that a model definition includes the ability to define optimization parameters such as objective functions, flux bounds and constraints. Currently this is not possible in the Systems Biology Markup Language (SBML) Level 2 or Level 3 core specification (Hucka et al., 2011, 2003).

The question of how to encode constraint based (also referred to as steady state or FBA) models in SBML is not new. However, advances in the methods used to construct genome scale constraint based models and the wider adoption of constraint based modeling in biotechnological/medical applications have led to a rapid increase in both the number of models being constructed and the tools used to analyze them.

Faced with such growth, both in number and diversity, the need for a standardized data format for the definition, exchange and annotation of constraint based models has become critical. As the core model components (e.g. species, reactions, stoichiometry) can already be efficiently described in SBML (with its associated active community, software and tool support) the Flux Balance Constraints package aims to extend SBML Level 3 core by adding the elements necessary to encode current and future constraint based models.

1.1 Proposal corresponding to this package specification

This specification for Flux Balance Constraints in SBML Level 3 Version 1 is based on the proposal, by this documents authors, located at the following URL:


The tracking number in the SBML issue tracking system (SBML Team, 2010) for Flux Balance Constraints package activities is 3154219. The version of the proposal used as the starting point for this specification is the version of March 2012. Previous versions of the current proposal are:

Version 3 (March 2012)


Version 2 (March 2011)

http://sbml.org/Community/Wiki/SBML_LEVEL_3_Proposals/Flux_Constraints_Proposal

Version 1 (February 2010)

http://precedings.nature.com/documents/4236/version/1

Details of earlier independent proposals are provided in Section 2.
1.2 Tracking number

As initially listed in the SBML issue tracking system under:


1.3 Package dependencies

The Flux Balance Constraints package adds additional classes to SBML Level 3 Version 1 Core and has no dependency on any other SBML Level 3 package.

1.4 Document conventions

Following the precedent set by the SBML Level 3 Core specification document, 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 color 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 classes that are never instantiated directly, but rather serve as parents of other object 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

2.1 Problems with current SBML approaches

While there is currently no official way of encoding constraint based models in SBML L2 there have been pragmatic approaches used by a variety of groups and applications. Arguably the most comprehensive and widely used format is that used by the COBRA toolbox (Becker et al., 2007) where the metabolic reaction network is well defined using SBML's Reaction and Species classes. However, other FBA specific model components such as flux bounds and the reactions that take part in the objective function are less well defined. In this case LocalParameter elements are used which (implicitly) rely on e.g. all tools knowing and using the same naming convention for the parameter ID's. Furthermore, reaction annotations are generally stored as tool specific HTML key-value pairs in a Notes element which has routinely led to different research groups and software using in-house and/or tool specific ways to describe the same information. While a step in the right direction this format is not suitable for implementation in SBML Level 3.

It is worth noting that while SBML Level 2 does have a construct know as Constraint its function is limited to measuring and reporting the model variables behavior in time. In contrast the FluxBound enforces the bounds on a steady-state flux and they can therefore be considered to be independent of one another. In addition, certain attributes that were widely used by the constraint based modelling community such as the Species attribute charge were removed from SBML. This has had the effect that a significant number of tools and models are still limited to using SBML Level 2 Version 1.

2.2 Past work on this problem or similar topics

The problem of describing and annotating FBA models in SBML has been raised at various times in the past few years. In this regard there are two known putative proposals one by Karthik Raman and the other by the Church Laboratory. As far as we are aware these proposals never developed beyond their initial presentation at SBML forums/hackathons. In 2009 the discussion was reopened at the SBML Forum held in Stanford and has subsequently developed into the current active package proposal and this document (see Section 1). In reverse chronological order these are:

Brett Olivier (2009)  SBML Level 3 FBA package discussion


Karthik Raman (2005)  Flux annotations in SBML


Church laboratory (pre 2005)  Metabolic flux model annotations

http://sbml.org/Community/Wiki/Old_known_SBML_annotations_list
3 Proposed syntax and semantics

In this section, we define the syntax and semantics of the Flux Balance Constraints 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 13, we provide complete examples of using the constructs in an example SBML model.

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 Flux Balance Constraints 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 Flux Balance Constraints package, the value of this attribute must be set to “false”.

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

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

3.2 Primitive data types

Section 3.1 of the SBML Level 3 Version 1 Core specification defines a number of primitive data types and also uses a number of XML Schema 1.0 data types (Biron and Malhotra, 2000). More specifically we make use of integer, double, string, SId and SIdRef. In addition we make use of two new primitives: the enumerations FbcType and FbcOperation, see Figure 1 for the interrelation between these entities.

The SId type is used as the data type for the identifiers of FluxBound (Section 3.5), FluxObjective (Section 3.7) and Objective (Section 3.6) classes. In the FBC package the ListOfObjectives has an attribute of type SIdRef that is used to refer to an `active' Objective.

3.2.1 Type FbcType

The Flux Balance Constraints package defines a new enumerated type FbcType which represents the optimization sense of the objective function. It can have one of the following values “maximize” or “minimize”.

3.2.2 Type FbcOperation

The Flux Balance Constraints package defines a new enumerated type FbcOperation which represents a boolean operator. It can take only one of the following values: “lessEqual”, “greaterEqual” or “equal.”
3.3 The extended Model class

The SBML Model class is extended with the addition of two children, i.e. a listOfFluxBounds and a listOfObjectives and a Model may contain at most one of these lists.

3.3.1 The FBC listOfFluxBounds

As shown in Figure 1 the ListOfFluxBounds is derived from SBase and inherits the attributes metaid and sboTerm, as well as the subcomponents for Annotation and Notes. ListOfFluxBounds must contain at least one FluxBound (defined in Section 3.5).

3.3.2 The FBC listOfObjectives

As shown in Figure 1 the ListOfObjectives is derived from SBase and inherits the attributes metaid and sboTerm, as well as the subcomponents for Annotation and Notes. Unlike most other SBML ListOf classes, ListOfObjectives introduces an additional required attribute activeObjective. The ListOfObjectives must contain at least one Objective (defined in Section 3.6).
**The activeObjective attribute**

This attribute is of type `SIdRef` and can only refer to the `id` of an existing `Objective`. This required attribute exists so that when multiple `Objective`s are included in a single model, the model will always be well described i.e. there is a single, primary objective function which defines a single optimum and its associated solution space.

### 3.3.3 A note on units

The main unit definitions that should be considered when using the Flux Balance Constraints package are the global model definitions of "extent" and "time" as all FBC flux related classes (i.e. `FluxBound` and `FluxObjective`) implicitly attain the same unit as the `Reaction` that they reference. More details on units can be found in their respective class definitions.

### 3.4 The extended Species class

The Flux Balance Constraints package extends the SBML Level 3 Version 1 Core `Species` class with the addition of two attributes `charge` and `chemicalFormula`.

![Species (extended)](image)

**Figure 2:** A UML representation of the extended SBML `Species` class used in the Flux Balance Constraints package. See Section 1.4 for conventions related to this figure.

#### The charge attribute

The optional attribute `charge` which contains a signed integer referring to the `Species` object's charge and is defined as it was in the SBML Level 2 Version 1 specification: "The optional field charge takes an integer indicating the charge on the species (in terms of electrons, not the SI unit coulombs)."

#### The chemicalFormula attribute

The optional attribute `chemicalFormula` containing a string that represents the `Species` objects elemental composition.

```xml
<species metaid="meta_M_atp_c" id="M_atp_c" name="ATP" compartment="Cytosol" boundaryCondition="false" initialConcentration="0" hasOnlySubstanceUnits="false" fbc:charge="-4" fbc:chemicalFormula="C10H12N5O13P3"/>
```

While there are many ways of referring to an elemental composition the purpose of the `chemicalFormula` attribute is to allow reaction balancing and validation which is particularly important in constraint based models.

The format of `chemicalFormula` must consist only of atomic names (as in the Periodic Table) or user defined compounds either of which take the form of a single capital letter followed by zero or more lowercase letters. Where there is more than a single atom present, this is indicated with an integer. With regards to order (and enhance interoperability) it is recommended to use the Hill system order [Hill (1900, 2012)](Hill1900).

**Table 1:** Examples of chemical formulas written using the Hill System. As described in Section 3.4

<table>
<thead>
<tr>
<th>Chemical Formula</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2O4S</td>
<td>2H2O</td>
</tr>
<tr>
<td>C2H5Br</td>
<td>2H5Br</td>
</tr>
<tr>
<td>BrH</td>
<td>HBr</td>
</tr>
<tr>
<td>C10H12N5O13P3</td>
<td>C10H12N5O13P3</td>
</tr>
<tr>
<td>CH3I</td>
<td>CH3I</td>
</tr>
</tbody>
</table>
Using this notation the number of carbon atoms in a molecule is indicated first, followed by the number of hydrogen atoms and then the number of all other chemical elements in alphabetical order. When the formula contains no carbon; all elements, including hydrogen, are listed alphabetically.

3.5 The FBC FluxBound class

**FluxBound** is a new FBC class derived from **SBML SBase** that inherits **metaId** and **sboTerm**, as well as the sub-components for **Annotation** and **Notes**. The purpose of this class is to hold a single (in)equality that provides the maximum or minimum value that a reaction flux can obtain at steady state. It implements four attributes.

### The id and name attributes

A **FluxBound** has two optional attributes: **id** an attribute of type **SId** and **name** an attribute of type **string**.

### The reaction attribute

The required **reaction** attribute of type **SIdRef**. This attribute must refer to a **Reaction** element defined within the enclosing model.

### The operation attribute

The **operation** attribute contains a value of type **FbcOperation** that can take a limited set of boolean operators as defined in Section 3.2.2. The **operation** attribute represents a mathematical (in)equality of the form `<reaction><operator><value>` e.g. $R_5 \geq 0$, $R_5 \leq \infty$ and $R_7 = 1.0$. The mapping between traditional mathematical symbols and **FbcOperation** values is as follows:

- $<=$ → “lessEqual”
- $>$ = → “greaterEqual”
- $=$ = → “equal”

### The value attribute

The **value** attribute holds a **double** value representing the numerical value of the flux bound. This may include an explicitly defined $\pm \infty$ encoded as a value, e.g. “INF”.

### Encoding the FluxBound

As described in Section 3.5 the flux bound represents a mathematical (in)equality of the form `<reaction><operator><value>`. In SBML Level 3 Version 1 with FBC this is encoded as:

```xml
<fbc:listOfFluxBounds>
  <fbc:fluxBound fbc:id="R1b" fbc:reaction="R1" fbc:operation="greaterEqual" fbc:value="1.2"/>
  <fbc:fluxBound fbc:id="R2b" fbc:reaction="R2" fbc:operation="lessEqual" fbc:value="-1.2"/>
  <fbc:fluxBound fbc:id="R3b" fbc:reaction="R3" fbc:operation="greaterEqual" fbc:value="-INF"/>
  <fbc:fluxBound fbc:id="R4b" fbc:reaction="R4" fbc:operation="lessEqual" fbc:value="INF"/>
  <fbc:fluxBound fbc:id="R5b" fbc:reaction="R5" fbc:operation="equal" fbc:value="1"/>
</fbc:listOfFluxBounds>
```

For an example of the how the **FluxBound** relates to the description of the underlying mathematical model please see Section 4.1.2.

### Units

The **value** defined by the **FluxBound** has the units of the **reaction** that it refers to i.e. the globally defined unit of “extent per time.”
**Consistency of flux bounds**

It is possible, and in some cases necessary, to declare more than one `FluxBound` relating to a particular reaction. To allow as much flexibility to the user as possible, there is no restriction on the number of flux bounds and the operation specified by each that may be declared for a given reaction. However, in the case where multiple flux bounds are declared for one reaction the combined set must not produce inconsistent values for the upper and lower bounds of the constraint.

For example:

```xml
<fluxBound id="b1" reaction="R" operation="greaterEqual" value="3"/>
<fluxBound id="b2" reaction="R" operation="equal" value="2"/>
```

is obviously inconsistent since it defines $R \geq 3$ and $R = 2$ - which cannot be correct. However

```xml
<fluxBound id="b1" reaction="R" operation="greaterEqual" value="3"/>
<fluxBound id="b2" reaction="R" operation="equal" value="3"/>
```

would be fine since, whilst the information is essentially repeated, $R \geq 3$ and $R = 3$ produce the consistent result that $R = 3$.

**Reactions with undefined flux bounds**

In the spirit of SBML Level 3 Version 1 Core the Flux Balance Constraints package does not define any default values for any element. However, in the case of a reaction with no defined flux bounds it is possible to infer this information from the reaction reversibility. In this case: irreversible reactions should be considered to be positive, $0 \leq J \leq \infty$ and reversible ones free/unbound, $-\infty \leq J \leq \infty$.

Similarly, there is also the potential for a bound to “seemingly” conflict with the reaction that it bounds’ reversibility, e.g. a reaction is irreversible but has bounds $-\infty \leq J \leq \infty$. In the context of this package, flux bounds should be considered authoritative. This follows from the fact that a `FluxBound` can enforce an irreversible reaction, by restricting the flux ($0 \leq J \leq \infty$), as well as a reversible reaction ($-\infty \leq J \leq \infty$). It is left to the software implementation to deal with any obvious inconsistencies.
3.6 The FBC Objective class

The FBC **Objective** class is derived from SBML **SBBase** and inherits **metaid** and **sboTerm**, as well as the subcomponents for **Annotation** and **Notes**. An integral component in a complete description of a steady-state model is the so-called 'objective function' which generally consist of a linear combination of model variables (fluxes) and a sense (direction). In the FBC package this concept is succinctly captured in the **Objective** class.

**The id and name attributes**

An **Objective** has a required attribute **id** of type **SId** and an optional attribute **name** of type **string**.

**The type attribute**

The required **type** attribute contains an **FbcType** type which represents the sense of the optimality constraint and can take one of two values:

\[
\text{maximize} \rightarrow \text{"maximize"}
\]
\[
\text{minimize} \rightarrow \text{"minimize"}
\]

**The listOfFluxObjectives element**

The element **listOfFluxObjectives** which contains a **ListOfFluxObjectives** is derived from and functions like a typical SBML **ListOf** class with the restriction that it must contain one or more elements of type **FluxObjective** (see Section 3.7). This implies that if an **Objective** is defined there should be at least one **FluxObjective** contained in a **ListOfFluxObjectives**.

**Objective**

- **id**: **SId**
- **name**: **string** {use="optional"}
- **type**: **FbcType**

*Figure 4:* A UML representation of the Flux Balance Constraints package **Objective** class. See Section 1.4 for conventions related to this figure.

**Encoding the Objective**

The Flux Balance Constraints package allows for the definition of multiple model objectives with one being designated as active (see Section 3.6) as illustrated in this example:

```xml
<fbc:listOfObjectives fbc:activeObjective="obj1">
  <fbc:objective fbc:id="obj1" fbc:type="maximize">
    <fbc:listOfFluxObjectives>
      <fbc:fluxObjective fbc:reaction="R101" fbc:coefficient="1"/>
    </fbc:listOfFluxObjectives>
  </fbc:objective>
  <fbc:objective fbc:id="obj2" fbc:type="minimize">
    <fbc:listOfFluxObjectives>
      <fbc:fluxObjective fbc:reaction="R102" fbc:coefficient="-2.5"/>
      <fbc:fluxObjective fbc:reaction="R103" fbc:coefficient="1"/>
    </fbc:listOfFluxObjectives>
  </fbc:objective>
</fbc:listOfObjectives>
```

Note how both **Objective** instances differ in **type** and each contains different set of **FluxObjectives** (see Section 3.7). For an example of how the **Objective** relates to the description of the underlying mathematical model please see Section 4.1.3.
3.7 The FBC FluxObjective class

The FBC FluxObjective class is derived from SBML SBase and inherits metaid and sboTerm, as well as the sub-components for Annotation and Notes.

The FluxObjective class is a relatively simple container for a model variable weighted by a signed linear coefficient.

The id and name attributes

A FluxObjective has two optional attributes: id an attribute of type SId and name an attribute of type string.

The reaction and coefficient attributes

The required reaction is of type SIdRef and is restricted to refer only to a Reaction while the coefficient attribute holds a double referring to the coefficient that this FluxObjective takes in the enclosing Objective. For example the objective Maximize: 1 R1 + 2 R2 would be encoded as

```xml
<fbc:listOfObjectives fbc:activeObjective="obj1">
  <fbc:objective fbc:id="obj1" fbc:type="maximize">
    <fbc:listOfFluxObjectives>
      <fbc:fluxObjective fbc:reaction="R1" fbc:coefficient="1"/>
      <fbc:fluxObjective fbc:reaction="R2" fbc:coefficient="2"/>
    </fbc:listOfFluxObjectives>
  </fbc:objective>
</fbc:listOfObjectives>
```

**Figure 5:** A UML representation of the Flux Balance Constraints package FluxObjective class. See Section 1.4 for conventions related to this figure.

Units

As described above the FluxObjective defined here as $n \cdot J$ where the coefficient ($n$) is dimensionless and the value ($J$) takes the units of the reaction flux i.e. “extent per time”. Therefore, the FluxObjective ($n \cdot J$) has the unit “extent per time” where the units of reaction “extent” and “time” are defined globally.
This section contains a worked example showing the encoding of a model suitable for Flux Balance Analysis using the Flux Balance Constraints package.

### 4.1 Example one: the basic FBC syntax

#### 4.1.1 Kinetic model description

![Figure 6: FBC syntax example: a simple four reaction pathway. The reactions are R1, R2, X1, X2 with fixed species IN, OUT, ATP, NADH and variable species A, B.]

As shown in Figure 6 this example is a simple four reaction pathway that transforms metabolite IN to OUT. The model was created and analyzed using the SBW Flux Balance FBC implementation Bergmann (2012); Bergmann and Sauro (2006). In SBML each reaction is represented as a chemical process transforming reactants to products, e.g. reaction R1 is encoded in XML as (see also the complete example provided at the end of this section):

```xml
<reaction id="R1" reversible="false" fast="false">
  <listOfReactants>
    <speciesReference species="IN" stoichiometry="1" constant="true"/>
  </listOfReactants>
  <listOfProducts>
    <speciesReference species="A" stoichiometry="1" constant="true"/>
  </listOfProducts>
</reaction>
```

Using the reagent identity and stoichiometry it is possible to compactly describe this network in terms of its reaction stoichiometry as shown in Table 2 where each reaction is represented as a column.

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>X1</th>
<th>X2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*Table 2: Example one: stoichiometric matrix, N*

While the stoichiometry contains the structural properties of the reaction network the full description of a biological model can be described as a set of ordinary differential equations (ODE’s). Of course other formalisms do exist,
but here we will concentrate exclusively on kinetic models where the change in concentration of each variable component in the system ($\frac{ds}{dt}$) is a non-linear function of the rates of the reactions which either create or consume it (the product of the stoichiometric matrix, $N$ and the vector of reaction rates, $v$).

$$\frac{ds}{dt} = Nv$$  \hspace{1cm} (1)

The formulation of the kinetic model, as shown in Equation 1 is typical of the kind that can already be described using SBML Level 3 Version 1 Core where the vector $v$ would contain rate equations as a function of parameters and variable species. In a steady-state, constraint based model these rates are considered unknowns and the system of equations can be rewritten as a set of linear constraints (see Equation 2):

$$NJ = 0$$  \hspace{1cm} (2)

Note that the rate vector $v$ is now represented as the steady-state flux vector $J$. However, in order to perform a typical steady-state analysis such as flux balance analysis (FBA) we need to include more information into the model description. SBML Level 3 Version 1 Core does not have an unambiguous way of encoding either a capacity constraint or an objective target and for this we need to use the additional constructs provided by the Flux Balance Constraints package. In the following sections the same model data is shown encoded as XML and as a Linear Program (LP) in the common format used by IBM CPLEX.

### 4.1.2 Capacity constraints

A capacity constraint: in this example the maximum limit (upper bound) of the flux through reaction $R1$ is set to be one (with an arbitrary unit of flux). In LP format this can be written as:

```
Bounds
R1 <= 1.0
```

the same information encoded as XML:

```xml
<fbc:listOfFluxBounds>
    <fbc:fluxBound fbc:reaction="R1" fbc:operation="lessEqual" fbc:value="1"/>
</fbc:listOfFluxBounds>
```

### 4.1.3 Objective function

This described a target which can be maximized or minimized: in this example the flux through reaction $R2$ will be maximized.

```
Maximize
objective1_objf: + 1.0 R2
```

the same information encoded as XML:

```xml
<fbc:listOfObjectives fbc:activeObjective="objective1">
    <fbc:objective fbc:id="objective1" fbc:type="maximize">
        <fbc:listOfFluxObjectives>
            <fbc:fluxObjective fbc:reaction="R2" fbc:coefficient="1"/>
        </fbc:listOfFluxObjectives>
    </fbc:objective>
</fbc:listOfObjectives>
```
4.1.4 Complete worked example

To conclude we show how the complete model described in Figure 6 encoded as both an LP and as XML. Formulated as an LP the problem can be written as:

\[
\text{Maximize} \\
\text{objective1_objf: } + 1.0 \text{ R2} \\
\text{Subject To} \\
A: + R1 - X1 - X2 = 0.0 \\
B: - R2 + X1 + X2 = 0.0 \\
\text{Bounds} \\
R1 <= 1.0 \\
0.0 <= R2 <= +\infty \\
0.0 <= X1 <= +\infty \\
0.0 <= X2 <= +\infty \\
\text{END}
\]

Solving this we find that maximization of flux through \( R2 \) gives an optimal solution \( R2 = 1 \), shown in Equation 3, with one possible solution for \( J \).

\[
\begin{pmatrix}
1 & 0 & -1 & -1 \\
0 & -1 & 1 & 1 \\
\end{pmatrix}
\begin{pmatrix}
1.0 \\
1.0 \\
0.0 \\
1.0 \\
\end{pmatrix} = 0
\]

Finally we provide the complete model, described above, encoded using the Flux Balance Constraints package:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core"
xmlns:fbc="http://www.sbml.org/sbml/level3/version1/fbc/version1"
level="3" version="1" fbc:required="false">
  <model id="fbcSpecExample1" timeUnits="time">
    <listOfUnitDefinitions>
      <unitDefinition id="volume">
        <listOfUnits>
          <unit kind="litre" exponent="1" scale="0" multiplier="1"/>
        </listOfUnits>
      </unitDefinition>
      <unitDefinition id="substance">
        <listOfUnits>
          <unit kind="mole" exponent="1" scale="0" multiplier="1"/>
        </listOfUnits>
      </unitDefinition>
      <unitDefinition id="time">
        <listOfUnits>
          <unit kind="second" exponent="1" scale="0" multiplier="1"/>
        </listOfUnits>
      </unitDefinition>
    </listOfUnitDefinitions>
    <listOfCompartments>
      <compartment id="compartment" spatialDimensions="3" size="1" units="volume" constant="true"/>
    </listOfCompartments>
    <listOfSpecies>
      <species id="IN" compartment="compartment" initialConcentration="0" substanceUnits="substance"
hasOnlySubstanceUnits="false" boundaryCondition="true" constant="false"/>
      <species id="OUT" compartment="compartment" initialConcentration="0" substanceUnits="substance"
hasOnlySubstanceUnits="false" boundaryCondition="true" constant="false"/>
      <species id="A" compartment="compartment" initialConcentration="0" substanceUnits="substance"
```
Section 4.1 Example one: the basic FBC syntax

```xml
<model>
  <listOfSpecies>
    <species id="B" compartment="compartment" initialConcentration="0" substanceUnits="substance"
      hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
    <species id="ATP" compartment="compartment" initialConcentration="0" substanceUnits="substance"
      hasOnlySubstanceUnits="false" boundaryCondition="true" constant="false"/>
    <species id="NADH" compartment="compartment" initialConcentration="0" substanceUnits="substance"
      hasOnlySubstanceUnits="false" boundaryCondition="true" constant="false"/>
  </listOfSpecies>
  <listOfReactions>
    <reaction id="R1" reversible="false" fast="false">
      <listOfReactants>
        <speciesReference species="IN" stoichiometry="1" constant="true"/>
      </listOfReactants>
      <listOfProducts>
        <speciesReference species="A" stoichiometry="1" constant="true"/>
      </listOfProducts>
    </reaction>
    <reaction id="R2" reversible="false" fast="false">
      <listOfReactants>
        <speciesReference species="B" stoichiometry="1" constant="true"/>
      </listOfReactants>
      <listOfProducts>
        <speciesReference species="OUT" stoichiometry="1" constant="true"/>
      </listOfProducts>
    </reaction>
    <reaction id="X1" reversible="false" fast="false">
      <listOfReactants>
        <speciesReference species="A" stoichiometry="1" constant="true"/>
      </listOfReactants>
      <listOfProducts>
        <speciesReference species="ATP" stoichiometry="1" constant="true"/>
        <speciesReference species="B" stoichiometry="1" constant="true"/>
      </listOfProducts>
    </reaction>
    <reaction id="X2" reversible="false" fast="false">
      <listOfReactants>
        <speciesReference species="A" stoichiometry="1" constant="true"/>
      </listOfReactants>
      <listOfProducts>
        <speciesReference species="B" stoichiometry="1" constant="true"/>
        <speciesReference species="NADH" stoichiometry="1" constant="true"/>
      </listOfProducts>
    </reaction>
  </listOfReactions>
  <fbc:listOfFluxBounds>
    <fbc:fluxBound fbc:reaction="R1" fbc:operation="greaterEqual" fbc:value="0"/>
    <fbc:fluxBound fbc:reaction="R1" fbc:operation="lessEqual" fbc:value="1"/>
    <fbc:fluxBound fbc:reaction="R2" fbc:operation="greaterEqual" fbc:value="0"/>
    <fbc:fluxBound fbc:reaction="R2" fbc:operation="lessEqual" fbc:value="INF"/>
    <fbc:fluxBound fbc:reaction="X1" fbc:operation="greaterEqual" fbc:value="0"/>
    <fbc:fluxBound fbc:reaction="X1" fbc:operation="lessEqual" fbc:value="INF"/>
    <fbc:fluxBound fbc:reaction="X2" fbc:operation="greaterEqual" fbc:value="0"/>
    <fbc:fluxBound fbc:reaction="X2" fbc:operation="lessEqual" fbc:value="INF"/>
  </fbc:listOfFluxBounds>
  <fbc:listOfObjectives fbc:activeObjective="objective1">
    <fbc:objective fbc:id="objective1" fbc:type="maximize">
      <fbc:listOfFluxObjectives>
        <fbc:fluxObjective fbc:reaction="R2" fbc:coefficient="1"/>
      </fbc:listOfFluxObjectives>
    </fbc:objective>
  </fbc:listOfObjectives>
</model>
```
5 Best practices

In this section, we recommend a number of practices for using and interpreting various constructs in the Flux Balance Constraints package. These recommendations are non-normative, but we advocate them strongly; ignoring them will not render a model invalid, but may reduce inter-operability between software and models.

5.1 Examples contrasting the current SBML L2 encoding with L3 and FBC

These examples contrast some elements of an existing model, iJR904 from the BiGG Database encoded in the COBRA format Becker et al. (2007); Reed et al. (2003); Schellenberger et al. (2010) that have been translated into SBML Level 3 Version 1 using the CBMPy implementation of the FBC package Olivier (2012); Olivier et al. (2005) and libSBML experimental ver. 5.6.0 Bornstein et al. (2008).

**Objective function definition**

**SBML Level 2 objective function**

```xml
<reaction id="R_BiomassEcoli" name="BiomassEcoli" reversible="false">
  <kineticLaw>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <ci>FLUX_VALUE</ci>
    </math>
    <listOfParameters>
      <parameter id="LOWER_BOUND" value="0" units="mmol_per_gDW_per_hr"/>
      <parameter id="UPPER_BOUND" value="999999" units="mmol_per_gDW_per_hr"/>
      <parameter id="OBJECTIVE_COEFFICIENT" value="1"/>
      <parameter id="FLUX_VALUE" value="0" units="mmol_per_gDW_per_hr"/>
    </listOfParameters>
  </kineticLaw>
</reaction>
```

**The SBML Level 3 objective function**

```xml
<fbc:listOfObjectives fbc:activeObjective="obj1">
  <fbc:objective fbc:id="obj1" fbc:type="maximize">
    <fbc:listOfFluxObjectives>
      <fbc:fluxObjective fbc:reaction="R_BiomassEcoli" fbc:coefficient="1"/>
    </fbc:listOfFluxObjectives>
  </fbc:objective>
</fbc:listOfObjectives>
```

**Species definition**

It is particularly useful to contrast the differences in the Species definition as it is used in genome scale models.

**SBML Level 2 Species annotation version 1**

To begin with we let's examine the SBML Level 2 Version 1 species definition used by the BiGG database and COBRA Becker et al. (2007); Schellenberger et al. (2010). Note how the name attribute is overloaded with the chemical formula.

```xml
<species id="M_atp_c" name="ATP_C10H12N5O13P3" compartment="Cytosol" charge="-4"/>
```
SBML Level 2 Species annotation version 2

A newer variation of the above, probably necessitated by the discontinuation of the charge attribute in SBML and libSBML

```xml
<species id="M_atp_c" name="ATP" compartment="c">
  <notes>
    <body xmlns="http://www.w3.org/1999/xhtml">
      <p>FORMULA: C10H12N5O13P3</p>
      <p>CHARGE: -4</p>
    </body>
  </notes>
</species>
```

The SBML Level 3 Species attributes

Hopefully, with the adoption of SBML FBC these species properties can be unified into a common format.

```xml
<species metaid="meta_M_atp_c" id="M_atp_c" name="ATP" compartment="Cytosol"
  boundaryCondition="false" initialConcentration="0" hasOnly Substance Units="false"
  fbc:charge="-4" fbc:chemicalFormula="C10H12N5O13P3"/>
```

Reaction definition and flux bounds

SBML Level 2 Reaction

```xml
<reaction id="R_GTHS" name="glutathione synthetase" reversible="false">
  <notes>
    <html:p>Abbreviation: R_GTHS</html:p>
    <html:p>EC Number: 6.3.2.3</html:p>
    <html:p>SUBSYSTEM: Cofactor and Prosthetic Group Biosynthesis</html:p>
    <html:p>Equation: [c] : atp + glucys + gly --&gt; adp + gthrd + h + pi</html:p>
    <html:p>Confidence Level: 0</html:p>
    <html:p>LOCUS:b2947#ABBREVIATION:gshB#ECNUMBERS:6.3.2.3#</html:p>
    <html:p>NAME:glutathione synthase#ABBREVIATION:GshB#GENE ASSOCIATION: (b2947)</html:p>
  </notes>
  <listOfReactants>
    <speciesReference species="M_atp_c" stoichiometry="1"/>
    <speciesReference species="M_glucys_c" stoichiometry="1"/>
    <speciesReference species="M_gly_c" stoichiometry="1"/>
  </listOfReactants>
  <listOfProducts>
    <speciesReference species="M_adp_c" stoichiometry="1"/>
    <speciesReference species="M_gthrd_c" stoichiometry="1"/>
    <speciesReference species="M_h_c" stoichiometry="1"/>
    <speciesReference species="M_pi_c" stoichiometry="1"/>
  </listOfProducts>
  <kineticLaw>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <ci>FLUX_VALUE</ci>
    </math>
  </kineticLaw>
</reaction>
```
The SBML Level 3 Reaction and FluxBound

As an example and where (unambiguously) possible the SBML Level 2 annotation has been converted into MIRIAM compliant RDF; in this case the EC number. In addition this example highlights an open issue, namely, how to deal with the information currently encoded in the Notes element. Currently this information is not completely encodable using the current SBML annotation mechanism, however, for an example of an alternative mechanism, implemented as a tool specific annotation please see Section 6.
6 Future development

In this section we highlight some open issues not addressed in this version of the FBC specification.

6.1 Model annotation

While the Flux Balance Constraints package addresses the basic constraint based model structure the effective annotation of individual model components is critical to its efficient (re)use. Here two examples of custom annotations are provided as an example of how this could be achieved.

6.1.1 Gene association annotation

An example of an annotation currently encoded in the Notes element that is not encodable using the SBML Annotation mechanism is the so called “Gene Association”.

In order to capture this information we propose the following mechanism which can be used to represent a logical expression containing “genes” in this case the final (transcribed/translated) protein product of a gene which forms a sub-component of a protein.

```xml
<fbc:listOfGeneAssociations>
  <fbc:geneAssociation fbc:id="ga1" fbc:reaction="R_ACGAptsp">  
    <fbc:or>
      <fbc:and>
        <fbc:gene>b2415</fbc:gene>
        <fbc:gene>b2417</fbc:gene>
        <fbc:gene>b2416</fbc:gene>
      </fbc:and>
      <fbc:and>
        <fbc:gene>b2415</fbc:gene>
        <fbc:gene>b0679</fbc:gene>
        <fbc:gene>b2416</fbc:gene>
      </fbc:and>
    </fbc:or>
  </fbc:geneAssociation>
</fbc:listOfGeneAssociations>
```

6.1.2 A generic annotation mechanism

An annotation issue previously introduced in the example shown in Section 4 where a SBML Level 2 Reaction is encoded using the Flux Balance Constraints package.

**SBML Level 2 Reaction**

```xml
<reaction id="R_GTHS" name="glutathione_synthetase" reversible="false">
  <notes>
    <html:p>Abbreviation: R_GTHS</html:p>
    <html:p>EC Number: 6.3.2.3</html:p>
    <html:p>SUBSYSTEM: Cofactor and Prosthetic Group Biosynthesis</html:p>
    <html:p>Equation: [c] : atp + glucys + gly --&gt; adp + gthrd + h + pi</html:p>
    <html:p>Confidence Level: 0</html:p>
    <html:p>LOCUS:b2947#ABBREVIATION:gshB#ECNUMBERS:6.3.2.3#</html:p>
    <html:p>NAME:glutathione synthetase#ABBREVIATION:GshB#</html:p>
  </notes>
  <listOfReactants>
    <speciesReference species="M_atp_c" stoichiometry="1"/>
    <speciesReference species="M_glucys_c" stoichiometry="1"/>
    <speciesReference species="M_gly_c" stoichiometry="1"/>
  </listOfReactants>
</reaction>
```
SBML Level 3 Reaction

Note how that in order to maintain all the annotations encoded in the SBML Level 2 Reaction notes a custom annotation is introduced i.e. the KeyValueData class\(^1\).

In general this example highlights the need for a community supported annotation mechanism for genome scale, constraint based models.

\(^1\)More information about this annotation is available at http://pysces.sourceforge.net/KeyValueData
<speciesReference species="M_gthrd_c" stoichiometry="1"/>
<speciesReference species="M_h_c" stoichiometry="1"/>
<speciesReference species="M_pi_c" stoichiometry="1"/>
</listOfProducts>
</reaction>
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 Flux Balance Constraints 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 Flux Balance Constraints 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 Flux Balance Constraints 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 Flux Balance Constraints package.

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

### General rules about this package

- **fbc-10101** ✓ To conform to the Flux Balance Constraints package specification for SBML Level 3 Version 1, an SBML document must declare the use of the following XML Namespace:

  \[\text{http://www.sbml.org/sbml/level3/version1/fbc/version1}\]  

  (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.1 on page 6.)

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

### General rules about identifiers

- **fbc-10301** ✓ (Extends validation rule #10301 in the SBML Level 3 Version 1 Core specification.) Within a Model the values of the attributes id and fbc:id on every instance of the following classes of objects must be unique across the set of all id and fbc: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 `FluxBound`, `Objective` and `FluxObjective` objects defined by the Flux Balance Constraints package. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.2 on page 6.)

**fbc-10302 ✓** The value of a `fbc:id` attribute must always conform to the syntax of the SBML data type `SId`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.2 on page 6.)

### Rules for the extended SBML class

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

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

**fbc-20103 ✓** The value of attribute `fbc:required` on the SBML object must be set to “false”. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.1 on page 6.)

### Rules for extended Model object

**fbc-20201 ✓** There may be at most one instance of each of the following kinds of objects within a Model object using Flux Balance Constraints: `ListOfFluxBounds` and `ListOfObjectives`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.3 on page 7.)

**fbc-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: `ListOfFluxBounds` and `ListOfObjectives`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.3 on page 7.)

**fbc-20203 ✓** Apart from the general notes and annotation subobjects permitted on all SBML objects, a `ListOfFluxBounds` container object may only contain `FluxBound` objects. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.3 on page 7.)

**fbc-20204 ✓** Apart from the general notes and annotation subobjects permitted on all SBML objects, a `ListOfObjectives` container object may only contain `Objective` objects. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.3 on page 7.)

**fbc-20205 ✓** A `ListOfFluxBounds` 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 Flux Balance Constraints namespace are permitted on a `ListOfFluxBounds` object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.3 on page 7.)

**fbc-20206 ✓** A `ListOfObjectives` object may have the optional attributes `metaid` and `sboTerm` defined by SBML Level 3 Core. Additionally the `ListObjectives` must contain the attribute `activeObjective`. No other attributes from the SBML Level 3 Core namespace or the Flux Balance Constraints namespace are permitted on a `ListObjectives` object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.3 on page 7.)

**fbc-20207 ✓** The value of attribute `fbc:activeObjective` on the `ListOfObjectives` object must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.3.2 on page 8).
Section A.1 Validation and consistency rules

**Rules for extended Species object**

- **fbc-20301** ✓ A SBML Species object may have the optional attributes `fbc:charge` and `fbc:chemicalFormula`. No other attributes from the Flux Balance Constraints namespaces are permitted on a Species. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.4 on page 8.)

- **fbc-20302** ✓ The value of attribute `fbc:charge` on the SBML Species object must be of the data type `integer`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.4 on page 8.)

- **fbc-20303** ✓ The value of attribute `fbc:chemicalFormula` on the SBML Species object must be set to a string consisting only of atomic names or user defined compounds and their occurrence. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.4 on page 8.)

**Rules for FluxBound object**

- **fbc-20401** ✓ A FluxBound object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a FluxBound. (References: SBML Level 3 Version 1 Core, Section 3.2.)

- **fbc-20402** ✓ A FluxBound object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a FluxBound. (References: SBML Level 3 Version 1 Core, Section 3.2.)

- **fbc-20403** ✓ A FluxBound object must have the required attributes `fbc:reaction`, `fbc:operation` and `fbc:value`, and may have the optional attributes `fbc:id` and `fbc:name`. No other attributes from the SBML Level 3 Flux Balance Constraints namespace are permitted on a FluxBound object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.5 on page 9.)

- **fbc-20404** ✓ The attribute `fbc:reaction` of a FluxBound must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.5 on page 9.)

- **fbc-20405** ✓ The attribute `fbc:name` of a FluxBound must be of the data type `string`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.5 on page 9.)

- **fbc-20406** ✓ The attribute `fbc:operation` of a FluxBound must be of the data type `FbcOperation` and thus its value must be one of “lessEqual”, “greaterEqual” or “equal”. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.5 on page 9.)

- **fbc-20407** ✓ The attribute `fbc:value` of a FluxBound must be of the data type `double`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.5 on page 9.)

- **fbc-20408** ✓ The value of the attribute `fbc:reaction` of a FluxBound object must be the identifier of an existing Reaction object defined in the enclosing Model object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.5 on page 9.)

- **fbc-20409** ✓ The combined set of all FluxBound’s with identical values for `fbc:reaction` must be consistent. That is while it is possible to define a lower and an upper bound for a reaction, it is not possible to define multiple lower or upper bounds. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.5 on page 9.)
Rules for Objective object

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

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

fbc-20503 ✓ A Objective object must have the required attributes fbc:id and fbc:type and may have the optional attribute fbc:name. No other attributes from the SBML Level 3 Flux Balance Constraints namespace are permitted on a Objective object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.6 on page 11.)

fbc-20504 ✓ The attribute fbc:name on a Objective must be of the data type string. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.6 on page 11.)

fbc-20505 ✓ The attribute fbc:type on a Objective must be of the data type FbcType and thus its value must be one of “minimize” or “maximize”. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.6 on page 11.)

fbc-20506 ✓ A Objective object must have one and only one instance of the ListOfFluxObjectives object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.6 on page 11.)

fbc-20507 ✓ The ListOfFluxObjectives subobject within a Objective object must not be empty. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.6 on page 11.)

fbc-20508 ✓ Apart from the general notes and annotation subobjects permitted on all SBML objects, a ListOfFluxObjectives container object may only contain FluxObjective objects. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.6 on page 11.)

fbc-20509 ✓ A ListOfFluxObjectives 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 Flux Balance Constraints namespace are permitted on a ListOfFluxObjectives object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.6 on page 11.)

Rules for FluxObjective object

fbc-20601 ✓ A FluxObjective object may have the optional SBML Level 3 Core attributes metaid and sboTerm. No other attributes from the SBML Level 3 Core namespace are permitted on a FluxObjective. (References: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20602 ✓ A FluxObjective object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a FluxObjective. (References: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20603 ✓ A FluxObjective object must have the required attributes fbc:reaction and fbc:coefficient, and may have the optional attributes fbc:id and fbc:name. No other attributes from the SBML Level 3 Flux Balance Constraints namespace are permitted on a FluxObjective object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.7 on page 12.)
The attribute `fbc:name` on a `FluxObjective` must be of the data type `string`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.7 on page 12.)

The value of the attribute `fbc:reaction` of a `FluxObjective` object must conform to the syntax of the SBML data type `SIdRef`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.7 on page 12.)

The value of the attribute `fbc:reaction` of a `FluxObjective` object must be the identifier of an existing `Reaction` object defined in the enclosing `Model` object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.7 on page 12.)

The value of the attribute `fbc:coefficient` of a `FluxObjective` object must conform to the syntax of the SBML data type `double`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.7 on page 12.)
We would like to thank all the people who contributed in various ways to the development of both the original proposal and this specification.

For financial/travel/technical and moral support we thank especially (in alphabetical order): Michael Hucka (Cal-Tech, USA), Ursula Kummer (Heidelberg University, Germany), Herbert Sauro (University of Washington, USA) and Bas Teusink (VU University Amsterdam, The Netherlands).

A special word of thanks goes to Sarah Keating for her invaluable work with the libSBML and SBML Toolbox implementations and critical reading of this document.

We also would like to thank (in alphabetical order) Ben Heavner, Daniel Hyduke, Nicolas Le Novère, Keiran Smallbone, Lucian Smith, Neil Swainston, members of the FBC Package Working Group and all others who contributed to discussions on various occasions.
References


