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 2, Release 1

March 25, 2015

This is a Release Candidate of the "fbc" package Version 2, Release 1 and not a normative document. Please send feedback to the Package Working Group mailing list at sbml-flux@lists.sourceforge.net. The official version 1 release 1 specification can be found here: http://identifiers.org/combine.specifications/sbml.level-3.version-1.fbc.version-1.release-1

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

This release of the specification is available at

Contents

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:

http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Flux_Balance_Constraints_Proposal_(2012)

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)

http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Flux_Balance_Constraints_Proposal_(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:
http://sourceforge.net/tracker/?func=detail&aid=3154219&group_id=71971&atid=894711.

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.
- *Grey*: Items colored ashgrey are components deprecated 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
V2 specific ways to describe the same information. A common example of such an annotation is the so called 'gene association' used to assess the effect of gene deletions on a reaction network. 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 known 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 modeling 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 Oberhardt et al. (2009).

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

http://sbml.org/images/4/4a/Olivier_sbml_forum_2009_09_04.pdf

Karthik Raman (2005) Flux annotations in SBML

http://sbml.org/images/d/d9/Raman-flux-annotations.pdf

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 18, 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:

"`http://www.sbml.org/sbml/level3/version1/fbc/version2`"

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 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/version2" 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 a new primitive: the enumeration `FbcType`, see Figure 1 for the interrelation between these entities.

The `SId` type is used as the data type for the identifiers of **GeneProduct** (Section 3.5) **FluxBound** (Section 3.6), **FluxObjective** (Section 3.8) and **Objective** (Section 3.7) 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 two values "`maximize`" or "`minimize`".

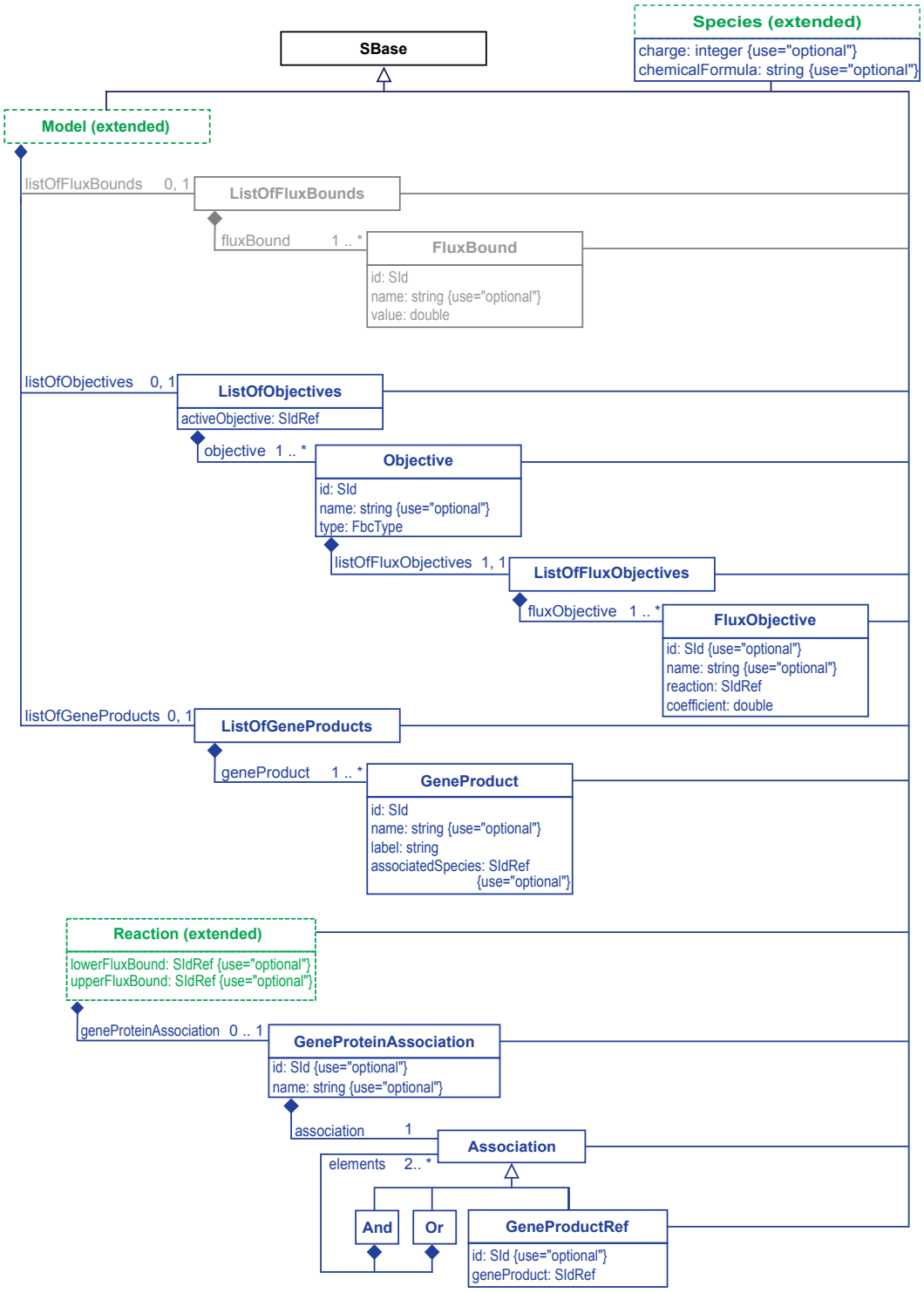

Figure 1: *A UML representation of the Flux Balance Constraints package. Derived from* ***SBase****, the FBC classes inherit support for constructs such as SBML* ***Notes*** *and* ***Annotation****'s. See Section 1.4 for conventions related to this figure. The individual classes are further discussed in the text.*

3.3 The extended Model class

The **SBML Model** class is extended by adding an optional `listOfFluxBounds`, `listOfObjectives` as well as a `listOfGeneProducts`. A **Model** may contain at most one of each 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 3.6).

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 3.7).

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 The FBC listOfGeneProducts

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

3.3.4 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`.

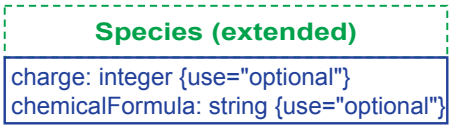

Figure 2: *A UML representation of the extended* ***SBML Species*** *class used in the Flux Balance Constraints package. See Section 1.4 for conventions related to this figure.*

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.

```
<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 inter-operability) it is recommended to use the Hill system order Hill (1900, 2012).

H2O4S	C2H5Br	BrH
C10H12N5O13P3	CH3I	

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

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 GeneProduct class

GeneProduct is a new FBC class derived from **SBML *SBase*** that inherits `metaid` and `sboTerm`, as well as the subcomponents for **Annotation** and **Notes**. The purpose of this class is to define a single gene product. It implements two required attributes `id` and `label` as well as two optional attributes `name` and `associatedSpecies`.

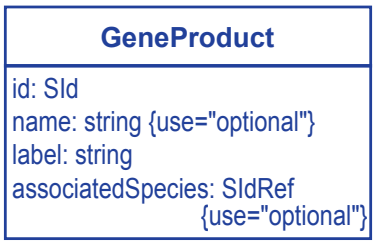

Figure 3: *A UML representation of the Flux Balance Constraints package **GeneProduct** class. See Section 1.4 for conventions related to this figure.*

The `id` *and* `name` *attributes*

A **GeneProduct** has a required attribute `id` of type `SId` and an optional attribute `name` of type `string`. The `id` attribute is used to uniquely identify a **GeneProduct** from a **GeneProductRef**.

The `label` *attribute*

The primary purpose of a **GeneProduct** is to reference a gene product uniquely. As these constructs do not follow the SId syntax, the required attribute `label` of type `string` is to be used to refer to them.

There is no set format for this identifier e.g. `Rv0649`, `3074.1` or `CRv4_Au5.s2.g9153.t1`. While ideally some form of restriction should be placed on the value of `label`, at this point it is left as a best practice suggestion that this attribute's value conform to the definition of an `SId`. As an example take an existing GPR annotation:

```
<p>GENE_ASSOCIATION: (Rv0649)</p>
```

this can now be formally (and unambiguously) encoded as:

```
<fbc:geneProduct metaid="meta_gene_1" fbc:id="gene1" fbc:label="Rv0649" fbc:associatedSpecies="s_Rv0649">
 <annotation>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#"
   xmlns:bqbiol="http://biomodels.net/biology-qualifiers/"
   xmlns:bqmodel="http://biomodels.net/model-qualifiers/">
   <rdf:Description rdf:about="#meta_gene_1">
    <bqbiol:is>
     <rdf:Bag>
      <rdf:li rdf:resource="http://identifiers.org/kegg.genes/mtu:Rv0649"/>
     </rdf:Bag>
    </bqbiol:is>
   </rdf:Description>
  </rdf:RDF>
 </annotation>
</fbc:geneProduct>

<species id="s_Rv0649" compartment="Cytosol" hasOnlySubstanceUnits="false"
                       boundaryCondition="true" constant="true"/>
```

The `associatedSpecies` *attribute*

A **GeneProduct** may optionally refer to a **Species** as used for example in the Manchester encoding. In that case the attribute `associatedSpecies` of type `SIdRef` is to be used to point to an existing **Species** in the model.

3.6 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 three attributes.

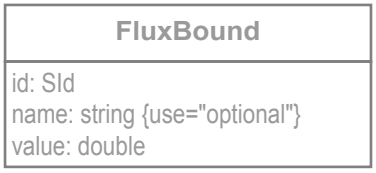

Figure 4: *A UML representation of the Flux Balance Constraints package* ***FluxBound*** *class. See Section 1.4 for conventions related to this figure.*

The `id` *and* `name` *attributes*

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

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`".

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."

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, $0 <= J <= \infty$ and reversible ones free/unbound, $-\infty <= J <= \infty$.

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 $-\infty <= J <= \infty$. 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 ($0 <= J <= \infty$), as well as a reversible reaction ($-\infty <= J <= \infty$). It is left to the software implementation to deal with any obvious inconsistencies.

3.7 The FBC Objective class

The FBC **Objective** class is derived from **SBML *SBase*** 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:

$$\begin{aligned} maximize &\mapsto \text{"maximize"} \\ minimize &\mapsto \text{"minimize"} \end{aligned}$$

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 3.8). This implies that if an **Objective** is defined there should be at least one **FluxObjective** contained in a **ListOfFluxObjectives**.

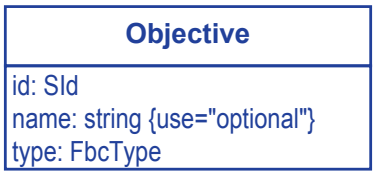

Figure 5: *A UML representation of the Flux Balance Constraints package* ***Objective*** *class. See Section 1.4 for conventions related to this figure.*

Encoding the Objective

The Flux Balance Constraints package allows for the definition of multiple model objectives with one being designated as active (see Section 3.7) as illustrated in this example:

```
<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 3.8). For an example of how the **Objective** relates to the description of the underlying mathematical model please see Section 4.1.3.

3.8 The FBC FluxObjective class

The FBC **FluxObjective** class is derived from **SBML** ***SBase*** and inherits `metaid` and `sboTerm`, as well as the subcomponents for **Annotation** and **Notes**.

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

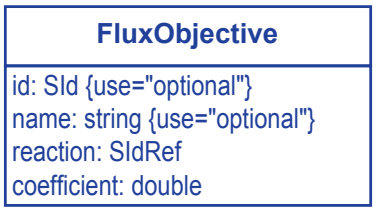

Figure 6: *A UML representation of the Flux Balance Constraints package* ***FluxObjective*** *class. See Section 1.4 for conventions related to this figure.*

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

```
<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>
```

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.

3.9 The extended Reaction class

The Flux Balance Constraints package extends the SBML Level 3 Version 1 Core **Reaction** class with the addition of a new optional element **GeneProteinAssociation** as well as two optional attributes `lowerFluxBound` and `upperFluxBound`.

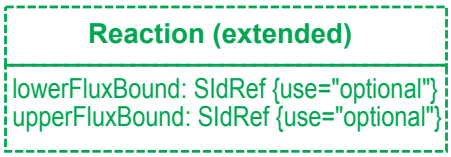

Figure 7: *A UML representation of the extended* ***SBML Reaction*** *class used in the Flux Balance Constraints package. See Section 1.4 for conventions related to this figure.*

The attributes `lowerFluxBound` *and* `upperFluxBound`

The optional attributes `lowerFluxBound` and `upperFluxBound` of type `SIdRef` are used to specify the lower and upper flux bounds for the **Reaction**. (In the case that equal bounds are to be used on the reaction, both attributes should point to the same SBML element).

The attributes have to refer to an existing **Parameter** in the model. This makes it possible to calculate the value of a flux bound based on **InitialAssignment** objects in the case of a constant **Parameter** (i.e. SBML parameters that have its `constant` attribute set to). Should the parameter not be constant, then its value can be additionally updated by all SBML Level 3 Version 1 Core constructs (i.e.: **EventAssignment**, **AssignmentRule**, and **AlgebraicRule**).

To ease adoption and ensure that not all tools have to immediately implement support for changing flux bounds the attributes may also refer to the an existing **FluxBound** in the model.

Encoding the flux bounds

To generate a list of (in)equalities for each reaction, out of the references in `upperFluxBound` and `lowerFluxBounds`, one first resolves the reference to the underlying **Parameter** or **FluxBound** . If they point to the same element, the equality will be of the form:

<reaction> = <value>

otherwise two inequalities are to be derived:

<reaction> >= <lowerFluxBound value>
<reaction> <= <upperFluxBound value>

In SBML Level 3 Version 1 with FBC this is encoded as:

```
<listOfParameters>
 <parameter constant="true" id="R1b" value="1.2"/>
 <parameter constant="true" id="R2b" value="-1.2"/>
 <parameter constant="true" id="negInf" value="-INF"/>
 <parameter constant="true" id="posInf" value="INF"/>
 <parameter constant="true" id="R5b" value="1"/>
</listOfParameters>

<listOfReactions>
 <reaction id="R1" fbc:lowerFluxBound="negInf" fbc:upperFluxBound="R1b" ... />
 <reaction id="R2" fbc:lowerFluxBound="R2b" fbc:upperFluxBound="posInf" ... />
 <reaction id="R3" fbc:lowerFluxBound="negInf" fbc:upperFluxBound="posInf" ... />
 <reaction id="R4" fbc:lowerFluxBound="posInf" fbc:upperFluxBound="posInf" ... />
 <reaction id="R5" fbc:lowerFluxBound="R5b" fbc:upperFluxBound="R5b" ... />
</listOfReactions>
```

additionally the **InitialAssignment** construct can be used to change the value of the **Parameter** elements. If in the example above `constant="false"` then the elements can be set additionally by **EventAssignment**, **AssignmentRule**, and **AlgebraicRule**.

In case no evaluation of **InitialAssignment**, **EventAssignment**, **AssignmentRule**, and **AlgebraicRule** is desired, it is possible to continue using the **FluxBound** construct using the same scheme:

```
<fbc:listOfFluxBounds>
 <fbc:fluxBound fbc:id="R1b" fbc:value="1.2"/>
 <fbc:fluxBound fbc:id="R2b" fbc:value="-1.2"/>
 <fbc:fluxBound fbc:id="negInf" fbc:value="-INF"/>
 <fbc:fluxBound fbc:id="posInf" fbc:value="INF"/>
 <fbc:fluxBound fbc:id="R5b" fbc:value="1"/>
</fbc:listOfFluxBounds>

<listOfReactions>
 <reaction id="R1" fbc:lowerFluxBound="negInf" fbc:upperFluxBound="R1b" ... />
 <reaction id="R2" fbc:lowerFluxBound="R2b" fbc:upperFluxBound="posInf" ... />
 <reaction id="R3" fbc:lowerFluxBound="negInf" fbc:upperFluxBound="posInf" ... />
 <reaction id="R4" fbc:lowerFluxBound="posInf" fbc:upperFluxBound="posInf" ... />
 <reaction id="R5" fbc:lowerFluxBound="R5b" fbc:upperFluxBound="R5b" ... />
</listOfReactions>
```

3.10 The FBC GeneProteinAssociation class

V2 The Flux Balance Constraints package defines a **GeneProteinAssociation** class that derives from ***SBase*** and inherits the attributes `metaid` and `sboTerm` as well as the subcomponents for **Annotation** and **Notes**. As shown in Figure 1 the **GeneProteinAssociation** class extends **Reaction** with one or more genes (or gene products). Where more than one gene is present in an association they are then expressed as a logical expression where genes are related to one another using logical 'and' and 'or' operators.

The `id` *attribute*

The **GeneProteinAssociation** class defines an optional attribute: `id` of type `SId`

The `name` *attribute*

The **GeneProteinAssociation** class defines an optional attribute: `name` of type `string`

The `association` *element*

Each **GeneProteinAssociation** contains a single ***Association***, however, as described in Section 3.11 an ***Association*** is an abstract class that implies that an `association` will always contain an instance of one of its sub-classes: **And**, **Or** or **GeneProductRef**.

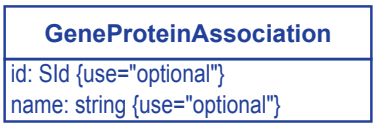

Figure 8: *A UML representation of the Flux Balance Constraints package* ***GeneProteinAssociation*** *class. See Section 1.4 for conventions related to this figure.*

Encoding the GeneProteinAssociation

As described in Section 3.10 the **GeneProteinAssociation** is simply a container that contains one of three types of ***Association*** either holding a single **GeneProductRef** or two or more genes in an **And** or **Or** relationship. For example the following, typical gene–protein association expression from the BiGG database *E. coli* reconstruction (iJR904) Reed et al. (2003); Schellenberger et al. (2010)

```
((B3670 and B3671) or (B0077 and B0078) or (B3768 and B3769 and B3767))
```

is now encoded in the Flux Balance Constraints package as:

```
<fbc:listOfGeneProducts>
  <fbc:geneProduct fbc:id="g_b3670" label="b3670" />
  <fbc:geneProduct fbc:id="g_b3671" label="b3671" />
  <fbc:geneProduct fbc:id="g_b0077" label="b0077" />
  <fbc:geneProduct fbc:id="g_b0078" label="b0078" />
  <fbc:geneProduct fbc:id="g_b3768" label="b3768" />
  <fbc:geneProduct fbc:id="g_b3769" label="b3769" />
  <fbc:geneProduct fbc:id="g_b3767" label="b3767" />
</fbc:listOfGeneProducts>

<reaction id = "R_ACHBS" ... >
<fbc:geneProteinAssociation fbc:id="ga_29">
 <fbc:or>
  <fbc:and>
   <fbc:geneProductRef fbc:geneProduct="g_b3670"/>
   <fbc:geneProductRef fbc:geneProduct="g_b3671"/>
  </fbc:and>
  <fbc:and>
   <fbc:geneProductRef fbc:geneProduct="g_b0077"/>
   <fbc:geneProductRef fbc:geneProduct="g_b0078"/>
  </fbc:and>
  <fbc:and>
   <fbc:geneGeneProductref fbc:geneProduct="g_b3768"/>
   <fbc:geneGeneProductref fbc:geneProduct="g_b3769"/>
   <fbc:geneGeneProductref fbc:geneProduct="g_b3767"/>
  </fbc:and>
 </fbc:or>
</fbc:geneProteinAssociation>
</reaction>
```

3.11 The FBC Association class

V2 The Flux Balance Constraints package defines an abstract ***Association*** class that is derived from ***SBase*** and inherits the attributes `metaid` and `sboTerm`, as well as the subcomponents for **Annotation** and **Notes**. It represents either a single gene, or a collection of genes in a logical expression and is only ever instantiated as one of its subclasses: **GeneProductRef** (Section 3.12), **And** (Section 3.13) and **Or** (Section 3.14).

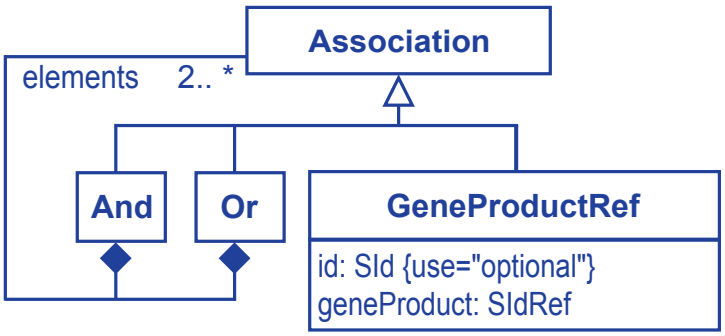

Figure 9: *A UML representation of the Flux Balance Constraints package* ***Association*** *and derived classes. See Section 1.4 for conventions related to this figure.*

3.12 The FBC GeneProductRef class

V2 The Flux Balance Constraints package defines a **GeneProductRef** class that references a gene (or gene product) from the **ListOfGeneProducts** declared on the **Model**. It is derived from an ***Association*** and thereby inherits the ***SBase*** attributes `metaid` and `sboTerm`, as well as the subcomponents for **Annotation** and **Notes** as described in Figure 9.

It is highly recommended that as a best practice and for future interoperability, a gene should be annotated using the inherited MIRIAM compliant SBML **Annotation** mechanism. Doing so will help reduce the dependence and ambiguity of using an overloaded, semantically meaningful `geneProduct`.

The `id` *attribute*

The **GeneProteinAssociation** class defines an optional attribute `id` of type `SId`.

The `geneProduct` *attribute*

The required `geneProduct` attribute of type `FbcSIdRef` references a **GeneProduct** element declared in the **ListOfGeneProducts**.

3.13 The FBC And class

V2 The Flux Balance Constraints package defines an **And** class that represents a gene (or gene product) and is derived from and ***Association*** and thereby inherits the ***SBase*** attributes `metaid` and `sboTerm`, as well as the subcomponents for **Annotation** and **Notes** as described in Figure 9. This class represents a set of two or more associations that are related in an order independent *'and'* relationship.

The `elements` *element*

Each **And** must contain two or more instances (not necessarily of the same type) of any ***Association*** subclass (**And**, **Or**, **GeneProductRef**).

```
<reaction id = "R_ACACCT" ... >
 <fbc:geneProteinAssociation fbc:id="ga_18">
  <fbc:and>
   <fbc:geneProductRef fbc:geneProduct="g_b3670"/>
   <fbc:geneProductRef fbc:geneProduct="g_b3671"/>
  </fbc:and>
 </fbc:geneProteinAssociation>
</reaction>
```

3.14 The FBC Or class

V2 The Flux Balance Constraints package defines an **Or** class that represents a gene (or gene product) and is derived from and ***Association*** and thereby inherits the ***SBase*** attributes `metaid` and `sboTerm`, as well as the subcomponents for **Annotation** and **Notes** as described in Figure 9. This class represents a set of two or more associations that are related in an order independent *'or'* relationship.

The `elements` *element*

Each **Or** must contain two or more instances (not necessarily of the same type) of any ***Association*** subclass (**And**, **Or**, **GeneProductRef**).

```
<reaction id = "R_ABTA" ... >
 <fbc:geneProteinAssociation fbc:id="ga_16">
  <fbc:or>
   <fbc:geneProductRef fbc:geneProduct="g_b2662"/>
   <fbc:geneProductRef fbc:geneProduct="g_b1302"/>
  </fbc:or>
 </fbc:geneProteinAssociation>
</reaction>
```

4 Illustrative examples of the FBC syntax

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

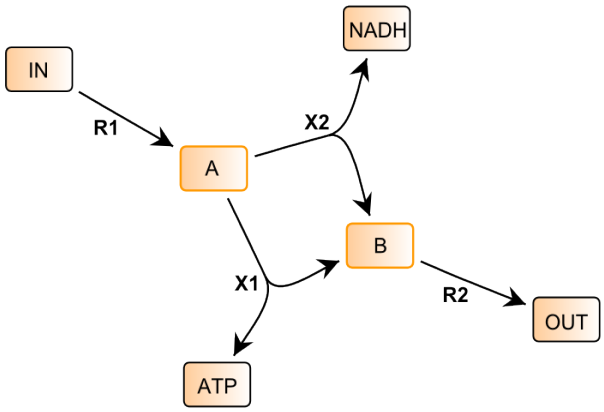

Figure 10: *FBC syntax example: a simple four reaction pathway. The reactions are R1, R2, X1, X2 with fixed species IN, OUT, ATP, NADH and variable species A, B.*

As shown in Figure 10 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):

```
<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.

	R1	R2	X1	X2
A	1	0	-1	-1
B	0	-1	1	1

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**).

$$\frac{ds}{dt} = \mathbf{N}\mathbf{v} \tag{1}$$

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):

$$\mathbf{N}\mathbf{J} = 0 \tag{2}$$

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:

```
<listOfParameters>
 <parameter id="R1b" constant="true" value="1" />
</listOfParameters>

...

<reaction id="R1" reversible="false" fast="false"
  fbc:upperFluxBound="R1b">
 <listOfReactants>
  <speciesReference species="IN" stoichiometry="1" constant="true"/>
 </listOfReactants>
 <listOfProducts>
  <speciesReference species="A" stoichiometry="1" constant="true"/>
 </listOfProducts>
</reaction>
```

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:

```
<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 10 encoded as both an LP and as XML. Formulated as an LP the problem can be written as:

```
\\ Example one LP format

Maximize
objective1_objf:  + 1.0 R2

Subject To
 A: + R1 - X1 - X2 = 0.0
 B: - R2 + X1 + X2 = 0.0

Bounds
R1 <= 1.0
0.0 <= R2 <= +inf
0.0 <= X1 <= +inf
0.0 <= X2 <= +inf

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 $\mathbf{J}$.

$$\begin{pmatrix} 1 & 0 & -1 & -1 \\ 0 & -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1.0 \\ \mathbf{1.0} \\ 0.0 \\ 1.0 \end{pmatrix} = \mathbf{0} \tag{3}$$

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

```
<?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/version2"
  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"
  hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
 <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>
<listOfParameters>
 <parameter id="R1u" constant="true" value="0"/>
 <parameter id="R1l" constant="true" value="1"/>
 <parameter id="R2u" constant="true" value="0"/>
 <parameter id="R2l" constant="true" value="INF"/>
 <parameter id="X1u" constant="true" value="0"/>
 <parameter id="X1l" constant="true" value="INF"/>
 <parameter id="X2u" constant="true" value="0"/>
 <parameter id="X2l" constant="true" value="INF"/>
</listOfParameters>
<listOfReactions>
 <reaction id="R1" reversible="false" fast="false"
  fbc:lowerFluxBound="R1l" fbc:upperFluxBound="R1u" >
  <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"
  fbc:lowerFluxBound="R2l" fbc:upperFluxBound="R2u" >
  <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"
  fbc:lowerFluxBound="X1l" fbc:upperFluxBound="X1u" >
  <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"
  fbc:lowerFluxBound="X2l" fbc:upperFluxBound="X2u" >
  <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: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>
</sbml>
```

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

```
<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

```
<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 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.

```
<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

```
<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.

```
<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"/>
```

Reaction definition and flux bounds

SBML Level 2 Reaction

```
<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#</html:p>
  <html:p>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>
  <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="0" />
   <parameter id="FLUX_VALUE" value="0" units="mmol_per_gDW_per_hr"/>
  </listOfParameters>
 </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, that issue is currently discussed on the package working group.

```
<reaction metaid="meta_R_GTHS" id="R_GTHS" name="glutathione␣synthetase" reversible="false">
 <annotation>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#"
    xmlns:bqbiol="http://biomodels.net/biology-qualifiers/"
    xmlns:bqmodel="http://biomodels.net/model-qualifiers/">
   <rdf:Description rdf:about="#meta_R_GTHS">
    <bqbiol:is>
     <rdf:Bag>
      <rdf:li rdf:resource="http://identifiers.org/ec-code/6.3.2.3"/>
     </rdf:Bag>
    </bqbiol:is>
   </rdf:Description>
  </rdf:RDF>
 </annotation>
 <fbc:listOfGeneAssociations>
   <fbc:geneAssociation fbc:id="ga1">
     <fbc:gene fbc:name="b2947" />
   </fbc:geneAssociation>
 </fbc:listOfGeneAssociations>
 <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>
</reaction>
```

```
<fbc:listOfFluxBounds>
 <fbc:fluxBound fbc:id="R_GTHS_lower_bnd" fbc:value="0"/>
 <fbc:fluxBound fbc:id="R_GTHS_upper_bnd" fbc:value="999999"/>
</fbc:listOfFluxBounds>
```

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 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: "`http://www.sbml.org/sbml/level3/version1/fbc/version2`". (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, 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/version2`". (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**, **Re-**

action, **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 2, 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 2, 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 2, 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 2, Section 3.3 on page 8.)

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 2, Section 3.3 on page 8.)

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 2, Section 3.3 on page 8.)

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 2, Section 3.3 on page 8.)

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 2, Section 3.3 on page 8.)

fbc-20206 ☑ A **ListOfObjectives** object may have the optional attributes `metaid` and `sboTerm` defined by SBML Level 3 Core. Additionally the **ListOfObjectives** 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 **ListOfObjectives** object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.3 on page 8.)

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 2, Section 3.3.2 on page 8).

fbc-20208 ☑ The value of attribute `fbc:activeObjective` on the **ListOfObjectives** object must be the identifier of an existing **Objective**. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.3.2 on page 8.)

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 2, 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 2, 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:id`, and `fbc:value`, and may have the optional attribute `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 2, Section 3.6 on page 10.)

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 2, Section 3.6 on page 10.)

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 2, Section 3.6 on page 10.)

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 2, Section 3.7 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 2, Section 3.7 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 2, Section 3.7 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 2, Section 3.7 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 2, Section 3.7 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 2, Section 3.7 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 2, Section 3.7 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 2, Section 3.8 on page 12.)

fbc-20604 ☑ 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 2, Section 3.8 on page 12.)

fbc-20605 ☑ 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 2, Section 3.8 on page 12.)

fbc-20606 ☑ 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 3.8 on page 12.)

fbc-20607 ☑ 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 2, Section 3.8 on page 12.)

Rules for extended Reaction object

fbc-20701 ☑ There may be at most one instance of a **GeneProteinAssociation** within a **Reaction** object using Flux Balance Constraints. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.9 on page 13.)

fbc-20702 ☑ A **SBML Reaction** object may have the optional attributes `fbc:lowerFluxBound` and `fbc:upperFluxBound`. No other attributes from the Flux Balance Constraints namespaces are permitted on a **Reaction**. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 1, Section 3.9 on page 13)

fbc-20703 ☑ The attribute `fbc:lowerFluxBound` of a **Reaction** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.9 on page 13.)

fbc-20704 ☑ The attribute `fbc:upperFluxBound` of a **Reaction** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.9 on page 13.)

fbc-20705 ☑ The attribute `fbc:lowerFluxBound` of a **Reaction** must point to an existing **Parameter** (or **FluxBound**) in the model. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.9 on page 13.)

fbc-20706 ☑ The attribute `fbc:upperFluxBound` of a **Reaction** must point to an existing **Parameter** (or **FluxBound**) in the model. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.9 on page 13.)

Rules for GeneProteinAssociation object

fbc-20801 ☑ A **GeneProteinAssociation** 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 **GeneProteinAssociation**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20802 ☑ A **GeneProteinAssociation** 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 **GeneProteinAssociation**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20803 ☑ A **GeneProteinAssociation** object may have the optional attributes `fbc:id` `fbc:name`. No other attributes from the SBML Level 3 Flux Balance Constraints namespace are permitted on a **GeneProteinAssociation** object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.10 on page 14.)

fbc-20804 ☑ The attribute `fbc:id` on a **GeneProteinAssociation** must be of the data type `SId`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.10 on page 14.)

fbc-20805 ☑ A **GeneProteinAssociation** object must have one and only one of the concrete ***Association*** objects: **GeneProductRef**, **And** or **Or**. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.10 on page 14.)

fbc-20806 ☑ The attribute `fbc:name` on a **GeneProteinAssociation** must be of the data type `string`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.10 on page 14.)

Rules for GeneProductRef object

fbc-20901 ☑ A **GeneProductRef** 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 **GeneProductRef**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20902 ☑ A **GeneProductRef** 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 **GeneProductRef**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20903 ☑ A **GeneProductRef** object must have the required attribute `fbc:geneProduct` and may have the optional attribute `fbc:id`. No other attributes from the SBML Level 3 Flux Balance Constraints namespace are permitted on a **GeneProductRef** object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.12 on page 16.)

fbc-20904 ☑ The attribute `fbc:geneProduct` on a **GeneProductRef** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.12 on page 16.)

fbc-20905 ☑ The attribute `fbc:id` on a **GeneProductRef** must be of the data type `SId`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.12 on page 16.)

fbc-20908 ☑ The attribute `fbc:geneProduct` on a **GeneProductRef** if set, must refer to `id` of a **GeneProduct** in the **Model**. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.12 on page 16.)

Rules for And object

fbc-20101 ☑ An **And** 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 an **And**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20102 ☑ An **And** 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 an **And**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20103 ☑ An **And** object must have two or more concrete ***Association*** objects: **GeneProductRef**, **And**, or **Or**. No other elements from the SBML Level 3 Flux Balance Constraints namespace are permitted on an **And** object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.13 on page 17.)

Rules for Or object

fbc-20111 ☑ An **Or** 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 an **Or**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20112 ☑ An **Or** 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 an **Or**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20113 ☑ An **Or** object must have two or more concrete ***Association*** objects: **GeneProductRef**, **And**, or **Or**. No other elements from the SBML Level 3 Flux Balance Constraints namespace are permitted on an **Or** object. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 2, Section 3.14 on page 17.)

B Summary of changes between versions

B.1 Version 1 to version 2

V2 Awaiting confirmation.

Acknowledgments

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 (CalTech, 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 M. Keating for her invaluable work with the libSBML and SBML Toolbox implementations and critical reading of this document.

V2 We also would like to thank (in alphabetical order) Andreas Dräger, Ali Ebrahim, Nicolas Rodriguez, Ben Heavner,
V2 Daniel Hyduke, Nicolas Le Novère, Kieran Smallbone, Lucian Smith, Neil Swainston, Alex Thomas, members of the FBC Package Working Group and all others who contributed to discussions on various occasions.

V2 If you feel you should be mentioned in this section please contact the authors and ask to be included.

References

Becker, S. A., Feist, A. M., Mo, M. L., Hannum, G., Palsson, B. O., and Herrgard, M. J. (2007). Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox. *Nat. Protocols*, 2(3):727–738. 1750-2799 10.1038/nprot.2007.99 10.1038/nprot.2007.99.

Bergmann, F. T. (2012). Sbw flux balance. Available via the World Wide Web at https://github.com/fbergmann/FluxBalance.

Bergmann, F. T. and Sauro, H. M. (2006). SBW - a modular framework for systems biology. In *Proceedings of the 38th conference on Winter simulation*, WSC '06, pages 1637–1645. Winter Simulation Conference.

Biron, P. V. and Malhotra, A. (2000). XML Schema part 2: Datatypes (W3C candidate recommendation 24 October 2000). Available via the World Wide Web at http://www.w3.org/TR/xmlschema-2/.

Bornstein, B. J., Keating, S. M., Jouraku, A., and Hucka, M. (2008). LibSBML: an API Library for SBML. *Bioinformatics*, 24(6):880–881.

Eriksson, H.-E. and Penker, M. (1998). *UML Toolkit*. John Wiley & Sons, New York.

Fallside, D. C. (2000). XML Schema part 0: Primer (W3C candidate recommendation 24 October 2000). Available via the World Wide Web at http://www.w3.org/TR/xmlschema-0/.

Hill, E. A. (1900). On a system of indexing chemical literature; adopted by the classification division of the U.S. patent office. *Journal of the American Chemical Society*, 22(8):478–494.

Hill, E. A. (2012). Wikipedia: The Hill System. Available via the World Wide Web at http://en.wikipedia.org/wiki/Hill_system.

Hucka, M., Bergmann, F. T., Hoops, S., Keating, S. M., Sahle, S., and Wilkinson, D. J. (2011). Available via the World Wide Web at http://sbml.org/Documents/Specifications.

Hucka, M., Finney, A., Sauro, H., Bolouri, H., Doyle, J., Kitano, H., Arkin, A., Bornstein, B., Bray, D., Cornish-Bowden, A., Cuellar, A., Dronov, S., Gilles, E., Ginkel, M., Gor, V., Goryanin, I., Hedley, W., Hodgman, T., Hofmeyr, J., Hunter, P., Juty, N., Kasberger, J., Kremling, A., Kummer, U., Novère, N. L., Loew, L., Lucio, D., Mendes, P., Minch, E., Mjolsness, E., Nakayama, Y., Nelson, M., Nielsen, P., Sakurada, T., Schaff, J., Shapiro, B., shimizu, T., Spence, H., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., and Wang, J. (2003). The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–31.

Oberhardt, M. A., Palsson, B. Ø., and Papin, J. A. (2009). Applications of genome-scale metabolic reconstructions. *Molecular Systems Biology*, 5:320.

Oestereich, B. (1999). *Developing Software with UML: Object-Oriented Analysis and Design in Practice*. Addison-Wesley.

Olivier, B. G. (2012). CBMPy: Constraint Based Modelling in Python. Available via the World Wide Web at http://pysces.sf.net/cbm.

Olivier, B. G., Rohwer, J. M., and Hofmeyr, J.-H. S. (2005). Modelling cellular systems with PySCeS. *Bioinformatics*, 21(4):560–561.

Orth, J. D., Thiele, I., and Palsson, B. Ø. (2010). What is flux balance analysis? *Nature Biotechnology*, 28(3):245–248.

Reed, J. L., Vo, T. D., Schilling, C. H., and Palsson, B. O. (2003). An expanded genome-scale model of Escherichia coli K-12 (iJR904 GSM/GPR). *Genome Biol*, 4(9):R54.

SBML Team (2010). The SBML Issue Tracker. Available via the World Wide Web at http://sbml.org/issue-tracker.

Schellenberger, J., Park, J., Conrad, T., and Palsson, B. (2010). BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions. *BMC Bioinformatics*, 11(1):213.

Thompson, H. S., Beech, D., Maloney, M., and Mendelsohn, N. (2000). XML Schema part 1: Structures (W3C candidate recommendation 24 October 2000). Available online via the World Wide Web at the address http://www.w3.org/TR/xmlschema-1/.