SBML Level 3 Package Specification

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

Brett G. Olivier PhD
0000-0002-5293-5321
Systems Biology, A-LIFE
Vrije Universiteit Amsterdam
Amsterdam, NH

Frank T. Bergmann PhD
0000-0001-5553-4702
BioQUANT
Heidelberg University
Heidelberg

Sarah Keating PhD
0000-0002-3356-3542
Advanced Research Computing Centre
University College London
London, UK

Matthias König PhD
0000-0003-1725-179X
Institute for Theoretical Biology
Humboldt Universität zu Berlin
Berlin, DE

Version 3, Release 1

January 2, 2024

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
http://identifiers.org/combine.specifications/sbml.level-3.version-1.fbc.version-3.release-1

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. Due to their large size these models are generally underdetermined and constraint-based optimization methods (such as linear or mixed integer convex optimization) are used to analyze them. Optimization is assumed to occur within 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) which is performed on Genome Scale Metabolic 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 and few or none of the kinetic rate equations, flux capacity constraints 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 (Olivier and Bergmann) archived at the URL:

https://web.archive.org/web/20151006163154/http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Flux_Balance_Constraints_Proposal_(2012)

Issues with this and other SBML packages can be filed at https://github.com/sbmlteam/sbml-specifications/issues. Issues pertaining to this package in particular are labeled with the 'L3 Package: fbc' tag. 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:

Proposal version 3 (March 2012)

https://web.archive.org/web/20151006163154/http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Flux_Balance_Constraints_Proposal_(2012)

Proposal version 2 (March 2011)

https://web.archive.org/web/20120824234050/http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Flux_Constraints_Proposal

Proposal version 1 (February 2010)

https://doi.org/10.1038/npre.2010.4236.1

Details of related, earlier, independent proposals are provided in Section 2.

1.2 Package dependencies

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

1.3 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.
- *Red*: Text and items colored red represent changes that have been made since the last time the proposal was officially distributed.

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. For example, in this case **LocalParameter** elements are used which (implicitly) rely on 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. An example of such an annotation is the widely used 'gene protein association'. While a step in the right direction, this encoding is not suitable for direct translation into SBML Level 3.

It is perhaps worth noting that, while SBML Level 2 does have a construct known as **Constraint**, its function is traditionally limited to measuring and reporting a model variable's behavior in time. In contrast, the Flux Balance Constraints package considers a model at steady state and therefore time invariant. Instead it makes use of **Parameter** elements to define the allowable range that a steady-state flux may attain. Therefore 'flux bounds' and **Constraint** elements should be considered complementary to one another. Furthermore, certain attributes that were widely used by the constraint-based modeling community such as the **Species** attribute `charge` were removed in later versions of SBML. This has had the effect that a significant number of constraint-based modelling and metabolic flux analysis software still make use of 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, an initiative which has subsequently developed into the current active package proposal and this document. In reverse chronological order these are:

Brett Olivier (2009) SBML Level 3 FBA package discussion

https://github.com/sbmlteam/sbml-specifications/blob/develop/sbml-level-3/version-1/fbc/spec/history/Olivier_sbml_forum_2009_09_04.pdf

Karthik Raman (2005) Flux annotations in SBML

https://github.com/sbmlteam/sbml-specifications/blob/develop/sbml-level-3/version-1/fbc/spec/history/Raman-flux-annotations.pdf

Church laboratory (pre 2005) Metabolic flux model annotations

https://web.archive.org/web/20210115164041/http://sbml.org/Community/Wiki/Old_known_SBML_annotations_list

3 Package 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 23, 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/version3`

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/version3" 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 **Objective** (Section 3.6), **FluxObjective** (Section 3.7), **GeneProduct** (Section 3.5), **GeneProductAssociation** (Section 3.9), **GeneProductRef** (Section 3.11), **UserDefinedConstraint** (Section 3.14), **UserDefinedConstraintComponent** (Section 3.15) and **KeyValuePair** (Section 3.17) classes. In all cases where the primitive data type `SId` is used in the Flux Balance Constraints package it is used unchanged from its description in SBML Level 3 Version 1 Core. When used as the type of a `fbc:id` attribute, that ID is added to the core `SId` namespace, and must continue to follow those rules for uniqueness: no `fbc:id` may duplicate any other `fbc:id`, nor the `id` of any **Model**, **FunctionDefinition**, **Compartment**, **Species**, **Reaction**, **SpeciesReference**, **ModifierSpeciesReference**, **Event**, or **Parameter**, nor the `package:id` of any other SBML Level 3 package element that is also defined as being in the `SId` namespace.

In the Flux Balance Constraints package the **ListOfObjectives** has an attribute of type `SIdRef` that is used to refer to an 'active' **Objective** as does the extended **Reaction** class which defines two attributes referring to flux capacity constraints. The **GeneProductRef** class declares an attribute of type `SIdRef` which references a **GeneProduct** which itself contains an attribute that can refer to a **Species**. Finally, the **UserDefinedConstraint** and **UserDefinedConstraintComponent** both declare attributes of type `SIdRef` and a more detailed description of what they refer to can be found in the relevant class descriptions.

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`".

3.2.2 *Type* FbcVariableType

The Flux Balance Constraints package defines a new enumerated type `FbcVariableType` which represents the type of variable that occurs in either the **FluxObjective** or **UserDefinedConstraintComponent**. It can have one of two values, "`linear`" or "`quadratic`".

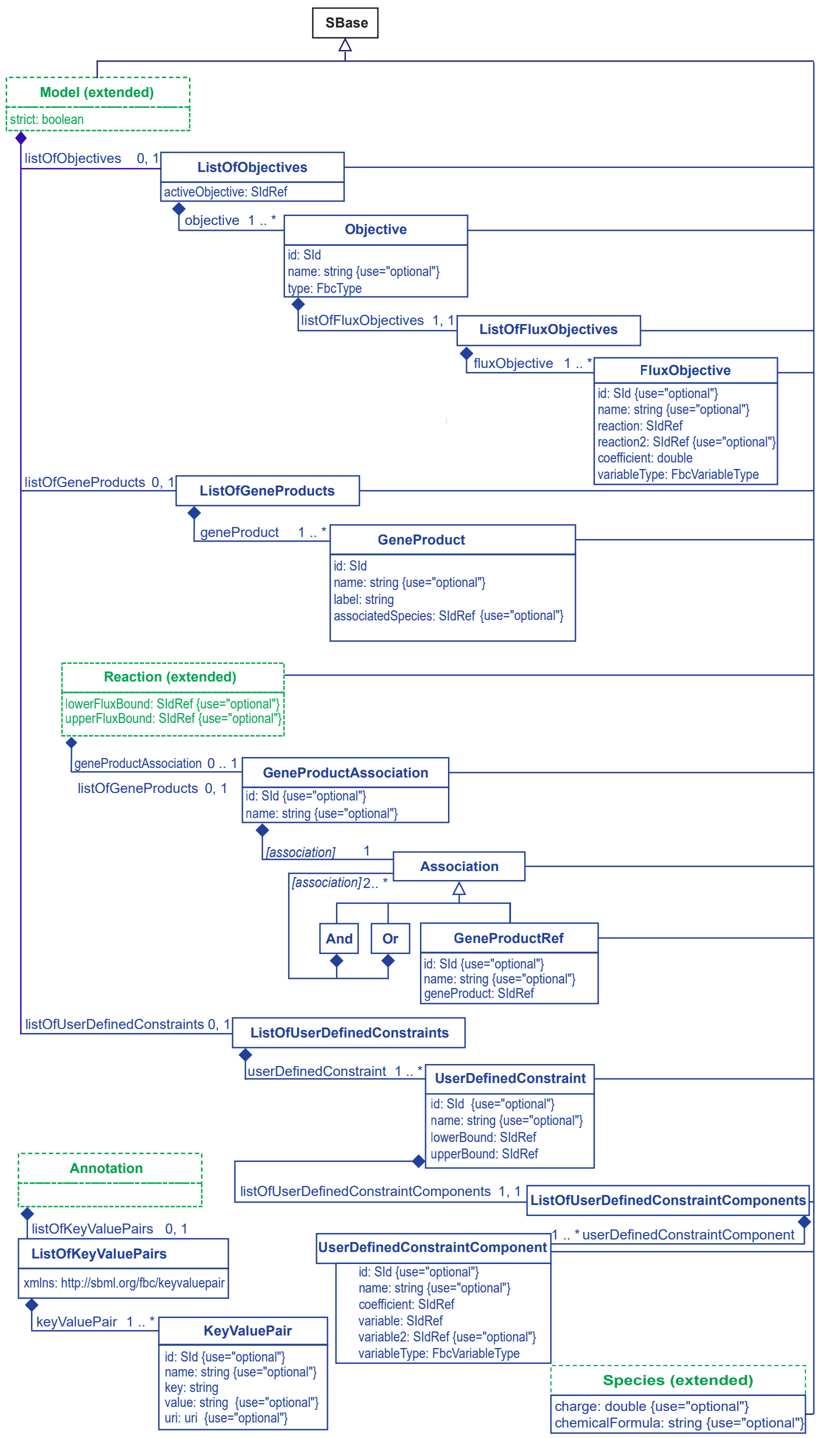

Figure 1: *A UML representation of the Flux Balance Constraints package. Derived from* ***SBase****, the FBC classes inherit support for constructs such as SBML* ***Notes*** *and* ***Annotation****'s. The [association] element name is the name of the class, de-capitalized. In this case, the possible values are "and", "or", or "geneProductRef". See Section 1.3 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 a mandatory boolean attribute `strict` as well as optional lists: `listOfObjectives`, `listOfGeneProducts` and `listOfUserDefinedConstraints`. A **Model** may contain, at most, one of each of these lists.

Figure 2: *A UML representation of the extended **SBML Model** class used in the Flux Balance Constraints package. See Section 1.3 for conventions related to this figure.*

The attribute `strict`

The mandatory attribute `strict`, of type `boolean`, is used to apply an additional set of restrictions to the model. The `strict` attribute ensures that the Flux Balance Constraints package can be used to encode legacy FBA models expressible as either a Linear or Quadratic Program (LP or QP) with software that is unable to analyze arbitrary mathematical expressions. In addition it ensures that a 'strict' model is fully described and mathematically consistent, for example, by ensuring that all fluxes have a valid upper or lower bound.

This is accomplished by defining a set of restrictions which come into effect if `strict` is set to "`true`":

- Each **Reaction** in a **Model** must define attributes `lowerFluxBound` and `upperFluxBound` with each pointing to a valid **Parameter** object defined in the current **Model**.
- Each **Parameter** object referred to by the **Reaction** attributes `lowerFluxBound` and `upperFluxBound` must have their `constant` attribute set to "`true`" and its `value` attribute set to a `double` value which may not be "`NaN`".
- **SpeciesReference** elements of **Reaction**s must have their `stoichiometry` attribute set to a `double` value that is neither "`NaN`" nor "`-INF`" nor "`INF`". In addition their `constant` attribute must be set to "`true`".
- **InitialAssignment** elements may neither target the **Parameter** elements referenced by the **Reaction** attributes `lowerFluxBound` and `upperFluxBound` nor any **SpeciesReference**.
- All defined **FluxObjective** elements must have their `coefficient` attribute set to a `double` value that is neither "`NaN`" nor "`-INF`" nor "`INF`".
- A **Reaction** `lowerFluxBound` attribute may not point to a **Parameter** with a `value` of "`INF`".
- A **Reaction** `upperFluxBound` attribute may not point to a **Parameter** with a `value` of "`-INF`".
- For all reactions, the value of a `lowerFluxBound` must be less than or equal to the value of the `upperFluxBound`.
- A **Parameter** whose `SId` is referenced by a **UserDefinedConstraintComponent** `coefficient` attribute has to be set as `constant` and not take the value "`NaN`" or "`±INF`".
- A non-constant **Parameter** whose `SId` is referenced by a **UserDefinedConstraintComponent** `variable` or `variable2` attribute may not be referenced by any **UserDefinedConstraintComponent** `coefficient`, **UserDefinedConstraint** `lowerBound` & `upperBound` or **Reaction** `lowerFluxBound` & `upperFluxBound` attribute.

While it is not compulsory for a 'strict' FBC model to define an **Objective**, doing so does allow it to be formulated as an LP and optimized, however, this decision is left to the modeler. Note that all other properties of the elements referred to in this list are as specified in the relevant SBML Level 3 Version 1 Core and FBC specifications.

Alternatively, if the value of the `strict` attribute is "`false`" then none of these restrictions apply and the model creator can choose to define FBC models that are not necessarily encodable as an LP or QP. For example, if `strict` is "`false`" the **InitialAssignment** construct may be used to set any valid numerical entity, including **Parameter** values and stoichiometric coefficients, with any `double`. In addition, **Parameter** elements are no longer required to be flagged as 'constant' thus allowing for an FBC model's use in alternative, hybrid modeling strategies.

3.3.1 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.

3.3.2 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).

3.3.3 The FBC listOfUserDefinedConstraints

As shown in Figure 1 the **ListOfUserDefinedConstraints** is derived from ***SBase*** and inherits the attributes `metaid` and `sboTerm`, as well as the subcomponents for **Annotation** and **Notes**. The **ListOfUserDefinedConstraints** must contain at least one **UserDefinedConstraint** (defined in Section 3.14).

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., **FluxObjective** implicitly attains 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`.

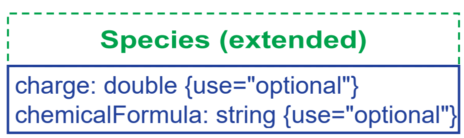

Figure 3: *A UML representation of the extended **SBML Species** class used in the Flux Balance Constraints package. See Section 1.3 for conventions related to this figure.*

The `charge` *attribute*

The optional attribute charge contains a signed double referring to the Species object's charge (in terms of electrons, not the SI unit coulombs). Note, that unlike FBC versions one and two a **Species** may, for the purposes of charge, be interpreted as a pseudoisomer or aggregate molecule and may assume a non-integer value. Non-integer charges should be used with caution as their use may have unintended side-effects, for example, with respect to the accuracy of reaction balancing.

The `chemicalFormula` *attribute*

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

While there are many ways of referring to an elemental composition, the purpose of the `chemicalFormula` attribute is to enable reaction balancing and validation, something of particular importance in constraint-based models.

The format of the `chemicalFormula` should, whenever possible, consist only of atomic names (as in the Periodic Table). Similarly, for enhanced inter-operability, the element order should be arranged according to the Hill system (see Table 1) (Hill, 1900, 2012). Using this notation the number of carbon atoms in a molecule is indicated first, followed by the number of hydrogen

H_2O_4S	C_2H_5Br	BrH
$C_{10}H_{12}N_5O_{13}P_3$	CH_3I	CH_4

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

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. Where there is more than a single atom present, this is indicated with an integer that follows the element symbol.

```
<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"/>

<species metaid="meta_M1" id="M1" compartment="C" boundaryCondition="false"
 initialConcentration="0" hasOnlySubstanceUnits="false" fbc:charge="0"
 fbc:chemicalFormula="RCONH2"/>

<species metaid="meta_M2" id="M2" compartment="C"boundaryCondition="false"
 initialConcentration="0" hasOnlySubstanceUnits="false" fbc:charge="0"
 fbc:chemicalFormula="COX"/>

<species metaid="meta_M3" id="M3" compartment="c" boundaryCondition="false"
 initialConcentration="0" hasOnlySubstanceUnits="false" fbc:charge="0"
 fbc:chemicalFormula="C2H4O2(CH2)n"/>
```

However, in certain situations, it may become necessary to use a generic symbol to represent an undefined or generic component of a user-defined compound. Generic components can only be specified as the symbols R or X. Furthermore, the

$C_{10}H_{12}N_5O_{13}P_3$	$RCONH_2$	COX	$C_2H_4O_2(CH_2)_n$

Table 2: *Examples of chemical formulas written using the allowed non-Hill symbols described in Section 3.4.*

undefined parenthesised group index $(\ldots)_n$ may also be used to indicate an arbitrary repetition of a chemical group. Note that a parenthesized group may only be followed by the subscript n. Integer values, for example, $(\ldots)_2$ and expressions such as $(\ldots)_{n-1}$ are considered invalid chemical formula.

Please note, the use of R, X and $(\ldots)_n$ is not generally advised, as any **Reaction** in which such a **Species** occurs cannot necessarily be balanced and may lead to the construction of an invalid model. To highlight this potential problem, any `chemicalFormula` that contains any of the aforementioned, non-Hill compatible symbols will raise a 'best practices' warning on model validation.

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`.

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

A **GeneProduct** has a required attribute `id` of type `SId` and an optional attribute `name` of type `string`. The unique `id` attribute is required to enable a **GeneProduct** to be referenced from a **GeneProductRef** used in a **GeneProductAssociation**.

The `label` *attribute*

The primary purpose of a **GeneProduct** is to uniquely reference a gene or implied gene product. As there is, currently, no restriction on the format of these these references they cannot be assumed to conform to an **SBML** `SId` syntax. Therefore the

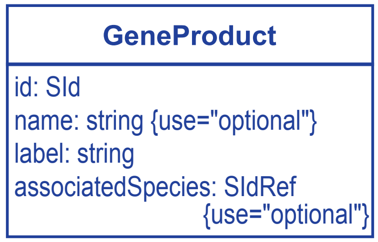

Figure 4: *A UML representation of the Flux Balance Constraints package **GeneProduct** class. For a complete description see Figure 1 as well as Section 1.3 for conventions related to this figure.*

Flux Balance Constraints package defines the required attribute `label`, of type `string`, for this purpose.

While ideally some form of restriction could, in principle, be placed on the value of `label`, at this point in time it is only possible to suggest that this attribute's value should conform to the definition of an `SId`. For example, consider an existing GPR annotation, as encountered in legacy **SBML** Level 2 encoded models:

```
<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"/>
```

Furthermore, it is a highly recommended 'best practice' that a **GeneProduct** 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 `label` attribute and enhance interoperability. For an example of this approach see Section 5.1.

The `associatedSpecies` *attribute*

A **GeneProduct** may, optionally, refer to a **Species** so as to provide compatibility with the Manchester style encoding of gene-protein associations. In this case the attribute `associatedSpecies` is of type `SIdRef` and, if defined, must point to an existing **Species** in the model.

3.6 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:

$$maximize \mapsto \text{“}\texttt{maximize}\text{”}$$
$$minimize \mapsto \text{“}\texttt{minimize}\text{”}$$

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**.

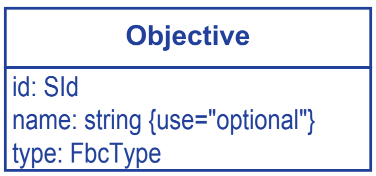

Figure 5: *A UML representation of the Flux Balance Constraints package* ***Objective*** *class. For a complete description see Figure 1 as well as Section 1.3 for conventions related to this figure.*

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:

```
<fbc:listOfObjectives fbc:activeObjective="obj1">
 <fbc:objective fbc:id="obj1" fbc:type="maximize">
  <fbc:listOfFluxObjectives>
   <fbc:fluxObjective fbc:reaction="R101" fbc:coefficient="1" fbc:variableType="linear"/>
  </fbc:listOfFluxObjectives>
 </fbc:objective>
 <fbc:objective fbc:id="obj2" fbc:type="minimize">
  <fbc:listOfFluxObjectives>
   <fbc:fluxObjective fbc:reaction="R102" fbc:coefficient="-2.5" fbc:variableType="linear"/>
   <fbc:fluxObjective fbc:reaction="R103" fbc:coefficient="1" fbc:variableType="linear"/>
  </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 subcomponents for **Annotation** and **Notes**. The **FluxObjective** class is a relatively simple container for model variables weighted by a signed numerical coefficient. It is defined as being either a 'linear' or 'quadratic' component.

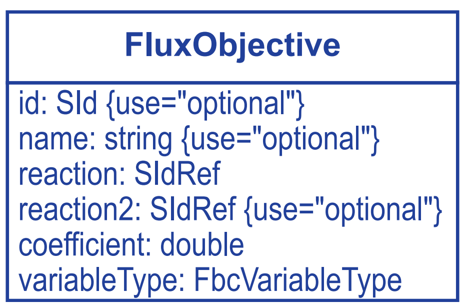

Figure 6: *A UML representation of the Flux Balance Constraints package **FluxObjective** class. For a complete description see Figure 1 as well as Section 1.3 for conventions related to this figure.*

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`, `reaction2` and `coefficient` attributes

The required `reaction` and optional `reaction2` attributes are of type `SIdRef` and are 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** expression.

The `variableType` attribute

The required `variableType` attribute contains a `FbcVariableType` that represents the type of exponent contained by the **FluxObjective**. For example, where J represents a steady-state flux in the `reaction` attribute, the `FbcVariableType` allows the definition of either a "`linear`", J^1 or "`quadratic`", J^2 term. In addition, a "`quadratic`" component may be defined using the optional attribute `reaction2`. For example, with `reaction2` represented by J_{R2} it is possible to define a two variable "`quadratic`" **FluxObjective** of the form $J^1 \times J^1_{R2}$.

Two examples of objective functions encoded in SBML

A common linear objective function (expressed in LP format): `Maximize: 1 R1`

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

An advanced objective function with both a linear and quadratic term (expressed in LP format): `Minimize: 1 R1 + [4 R2^2]/2`

```
<fbc:listOfObjectives fbc:activeObjective="obj2">
 <fbc:objective fbc:id="obj2" fbc:type="minimize">
  <fbc:listOfFluxObjectives>
   <fbc:fluxObjective fbc:reaction="R1" fbc:coefficient="1" fbc:variableType="linear"/>
   <fbc:fluxObjective fbc:reaction="R2" fbc:coefficient="2" fbc:variableType="quadratic"/>
  </fbc:listOfFluxObjectives>
 </fbc:objective>
</fbc:listOfObjectives>
```

Note that the LP format (CPLEX) expression of the quadratic term is written as $\frac{2\times 2\times R2^2}{2}$ and it is left up to the software to make any necessary translations of these terms to the format required by specific optimization libraries.

Units

As described above the linear **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 `linear` **FluxObjective**, ($n \cdot J$) has the unit $\frac{extent}{time}$ where the units of reaction "extent" and "time" are defined globally. Analogously, in the case of a `quadratic` **FluxObjective**, $n \cdot J^2$ this would be $\frac{extent^2}{time^2}$.

3.8 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 **GeneProductAssociation** as well as two optional attributes `lowerFluxBound` and `upperFluxBound`.

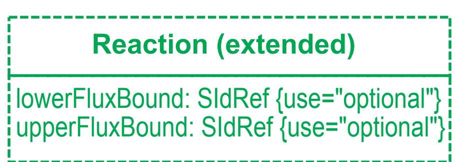

Figure 7: *A UML representation of the extended **SBML Reaction** class used in the Flux Balance Constraints package. For a complete description see Figure 1 as well as Section 1.3 for conventions related to this figure.*

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 a **Reaction** flux.

The attributes must refer to an existing **Parameter** in the model and in the case that equal bounds are required for a reaction, both attributes should point to the same **Parameter**. The Flux Balance Constraints package specifies Systems Biology Ontology

(SBO) terms that can be used to define the character of such a **Parameter**, please see the appendix describing SBO for more details.

Using a **Parameter** in this way makes it possible for other SBML elements to interact with these parameters depending on the value of the `fbc:strict` attribute of the **Model** (see also Section 3.3 on page 8).

For example, if the value of the `strict` attribute is "`false`", even in the case of a constant **Parameter** (i.e. SBML parameters that have their `constant` attribute set to "`true`") the value of a flux bound's value could be calculated using an **InitialAssignment**. Should the parameter not be constant, then its value can be additionally updated by all SBML Level 3 Version 1 Core constructs (for example **EventAssignment**, **AssignmentRule** and **AlgebraicRule**). However none of the aforementioned applies when the `strict` attribute is set to "`true`".

Encoding the flux bounds

To generate a list of (in)equalities for each reaction from the parameters reference in `upperFluxBound` and `lowerFluxBounds`, one must first resolve the reference to the underlying **Parameter** using the relationship shown in Equation 1:

$$\texttt{lowerBound} \le \textbf{Parameter} \le \texttt{upperBound} \tag{1}$$

In SBML Level 3 Version 1 with FBC Version 3 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="negInf" fbc:upperFluxBound="posInf" ... />
 <reaction id="R5" fbc:lowerFluxBound="R5b" fbc:upperFluxBound="R5b" ... />
</listOfReactions>
```

3.9 The FBC GeneProductAssociation class

The Flux Balance Constraints package defines a **GeneProductAssociation** 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 **GeneProductAssociation** class extends **Reaction** with one or more genes (or gene products). Where more than one gene (or gene product) is present in an association they are written as logical expressions and thereby related to one another using logical 'and' and 'or' operators.

The `id` and `name` attributes

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

The `association` element

Each **GeneProductAssociation** contains a single ***Association***, however, as described in Section 3.10 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**.

Encoding the GeneProductAssociation

As described in Section 3.9, the **GeneProductAssociation** is simply a container that contains one of three types of ***Association*** either holding a single **GeneProductRef** or two or more ***Association*** elements 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)

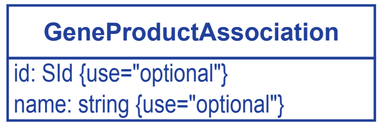

Figure 8: *A UML representation of the Flux Balance Constraints package* ***GeneProductAssociation*** *class. For a complete description see Figure 1 as well as Section 1.3 for conventions related to this figure.*

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

can now be encoded 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:geneProductAssociation 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:geneProductRef fbc:geneProduct="g_b3768"/>
     <fbc:geneProductRef fbc:geneProduct="g_b3769"/>
     <fbc:geneProductRef fbc:geneProduct="g_b3767"/>
    </fbc:and>
   </fbc:or>
 </fbc:geneProductAssociation>
</reaction>
```

3.10 The FBC Association class

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 3.11), **And** (Section 3.12) and **Or** (Section 3.13).

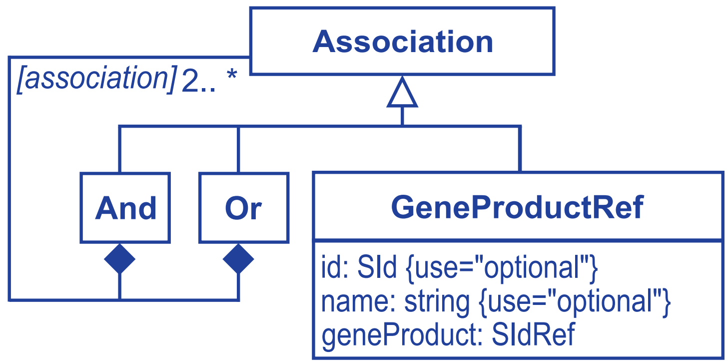

*Figure 9: A UML representation of the Flux Balance Constraints package **Association** and derived classes. The [association] element name is the name of the class, de-capitalized. In this case, the possible values are "and", "or", or "geneProductRef". For a complete description see Figure 1 as well as Section 1.3 for conventions related to this figure.*

3.11 The FBC GeneProductRef class

The Flux Balance Constraints package defines a **GeneProductRef** class that references a **GeneProduct** declared in **ListOfGeneProducts**, a child of **Model** (see Section 3.5). 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.

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

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

The `geneProduct` *attribute*

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

3.12 The FBC And class

The Flux Balance Constraints package defines an **And** class that 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. 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:geneProductAssociation fbc:id="ga_18">
  <fbc:and>
   <fbc:geneProductRef fbc:geneProduct="g_b3670"/>
   <fbc:geneProductRef fbc:geneProduct="g_b3671"/>
  </fbc:and>
 </fbc:geneProductAssociation>
</reaction>
```

3.13 The FBC Or class

The Flux Balance Constraints package defines an **Or** class that represents a gene (or gene product) and 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. This class represents a set of two or more ***Association*** elements 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:geneProductAssociation fbc:id="ga_16">
  <fbc:or>
   <fbc:geneProductRef fbc:geneProduct="g_b2662"/>
   <fbc:geneProductRef fbc:geneProduct="g_b1302"/>
  </fbc:or>
 </fbc:geneProductAssociation>
</reaction>
```

3.14 The FBC UserDefinedConstraint class

The FBC **UserDefinedConstraint** class is derived from **SBML *SBase*** and inherits `metaid` and `sboTerm`, as well as the subcomponents for **Annotation** and **Notes**. It's purpose is to allow the definition of non-stoichiometric constraints, that is, constraints that are not defined by the stoichiometrically coupled reaction network. In order to achieve this a new class of constraint is defined, the **UserDefinedConstraint**.

Analogous to the attributes described in the **Reaction** class (Section 3.8), the `lowerBound` and `upperBound` define the upper and lower bounds of the **UserDefinedConstraint**, such that:

$$\texttt{lowerBound} \leq \textbf{UserDefinedConstraint} \leq \texttt{upperBound} \tag{2}$$

The **UserDefinedConstraint** contains a **ListOfUserDefinedConstraintComponents** representing a linear combination of **UserDefinedConstraintComponents**. Similar to a **FluxObjective** each **UserDefinedConstraintComponent** contains a coefficient and model variable(s). In the case of **UserDefinedConstraintComponent** the `coefficient` refers to a constant **Parameter** while the `variable` and optional `variable2` attributes may either refer to a **Reaction** or to a non-constant **Parameter**. This allows for the creation of non-reaction variables. Quadratic constraint components are also supported, see the attribute descriptions for more detail.

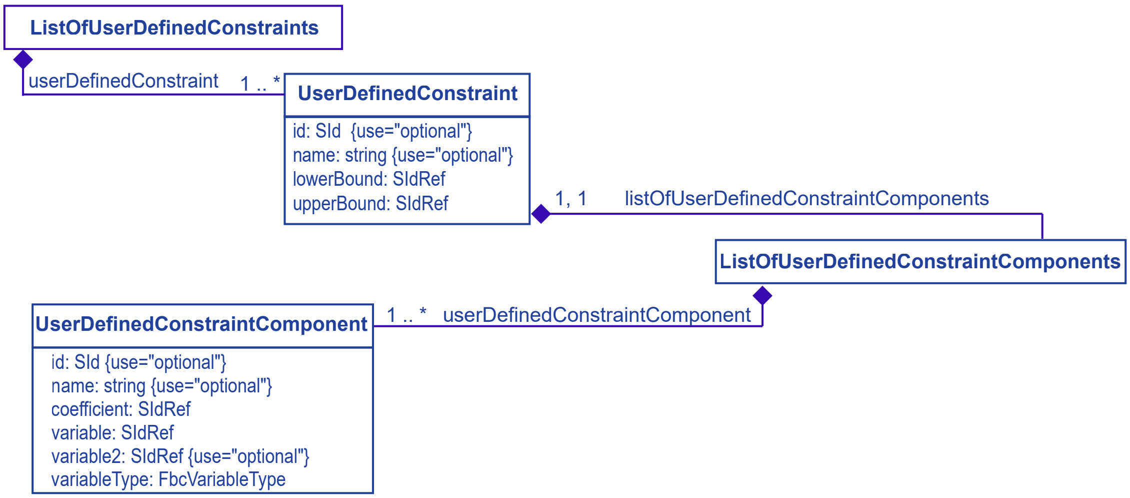

Figure 10: *A UML representation of the **SBML Model** class extended in the Flux Balance Constraints package by the **ListOfUserDefinedConstraints**. See Section 1.3 for conventions related to this figure.*

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

A **UserDefinedConstraint** has an optional `id` of type `SId` and an optional attribute `name` of type `string`.

The `lowerBound` *attribute*

The required `lowerBound` attribute contains an `SIdRef` that references a **Parameter** which contains the lower boundary value of the **UserDefinedConstraint**.

The `upperBound` *attribute*

The required `upperBound` attribute contains an `SIdRef` that references a **Parameter** which contains the upper boundary value of the **UserDefinedConstraint**.

The `listOfUserDefinedConstraintComponents` *element*

The element `listOfUserDefinedConstraintComponents` which contains a **ListOfUserDefinedConstraintComponents** is derived from and functions like a typical **SBML ListOf___** class with the restriction that it must contain one or more elements of type **UserDefinedConstraintComponent** (see Section 3.15).This implies that if a **UserDefinedConstraint** is defined there should be at least one **UserDefinedConstraintComponent** contained in a **ListOfUserDefinedConstraintComponents**.

3.15 The FBC UserDefinedConstraintComponent class

The FBC **UserDefinedConstraintComponent** class is derived from **SBML *SBase*** and inherits `metaid` and `sboTerm`, as well as the subcomponents for **Annotation** and **Notes**. The **UserDefinedConstraintComponent** class is a relatively simple container for a variable and a variable type specifier which is weighted by a signed coefficient.

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

An **UserDefinedConstraintComponent** has an optional `id` of type `SId` and an optional attribute `name` of type `string`.

The `coefficient` *attribute*

The required `coefficient` attribute contains an `SIdRef` that is restricted to reference only a **Parameter** which holds the coefficient value. In **strict** mode a **Parameter** whose `SId` is referenced by a `coefficient`, has to be set as `constant` and not take the value "`NaN`" or "`±INF`".

The `variable` *attribute*

The required `variable` attribute and optional `variable2` contains an `SIdRef` that is restricted to reference the `SId` of either a **Reaction** or non-constant **Parameter**. In **strict** mode such a non-constant **Parameter** whose `SId` is referenced by a **UserDefinedConstraintComponent** `variable` or `variable2` attribute may not be referenced by any **UserDefinedConstraintComponent** `coefficient`, **UserDefinedConstraint** `lowerBound` & `upperBound` or **Reaction** `lowerFluxBound` & `upperFluxBound` attribute.

The `variableType` *attribute*

The required `variableType` attribute contains a `FbcVariableType` that indicates whether a variable should be considered as "`linear`" or "`quadratic`".

An example of encoding two user defined constraints in SBML

The following example illustrates the encoding of two **UserDefinedConstraint**s. The first uses only **Reaction**s as variables, while the second introduces an artificial, non-constant, **Parameter** as a variable.

$$RGLX - RBTK = 5 \tag{3}$$

$$2 \cdot p1var - RGDP \geq 2 \tag{4}$$

```
<listOfParameters>
 <parameter id="uc1" value="5" constant="True"/>
 <parameter id="uc2lb" value="2" constant="True"/>
 <parameter id="uc2ub" value="INF" constant="True"/>
 <parameter id="ucco1a" value="1" constant="True"/>
 <parameter id="ucco1b" value="-1" constant="True"/>
 <parameter id="ucco2a" value="2" constant="True"/>
 <parameter id="ucco2b" value="-1" constant="True"/>
 <parameter id="p1var" value="NaN" constant="False"/>
</listOfParameters>

<fbc:listOfUserConstraints>
 <fbc:userContraint fbc:id="uc1" fbc:lowerBound="uc1" fbc:upperBound="uc1">
  <fbc:listOfUserConstraintComponents>
   <fbc:userConstraintComponent fbc:coefficient="ucco1a" fbc:variable="RGLX"
    variableType="linear"/>
   <fbc:userConstraintComponent fbc:coefficient="ucco1b" fbc:variable="RBTK"
    variableType="linear"/>
  </fbc:listOfUserConstraintComponents>
 </fbc:userContraint>
 <fbc:userContraint fbc:id="uc2" fbc:lowerBound="uc2lb" fbc:upperBound="uc2ub">
  <fbc:listOfUserConstraintComponents>
   <fbc:userConstraintComponent fbc:coefficient="ucco2a" fbc:variable="p1var"
    variableType="linear"/>
   <fbc:userConstraintComponent fbc:coefficient="ucco2b" fbc:variable="RGDP"
    variableType="linear"/>
  </fbc:listOfUserConstraintComponents>
 </fbc:userContraint>
</fbc:listOfUserConstraints>
```

3.16 The FBC ListOfKeyValuePairs class

The optional **ListOfKeyValuePairs**, see Figure 11 for details, forms the basis of a controlled annotation defined by the Flux Balance Constraints package. This element defines a 'descriptive list' of data keys and associated values which can be used to store metadata that is not encodable as a typical **SBML Annotation**.

```
<annotation>
 <listOfKeyValuePairs xmlns="http://sbml.org/fbc/keyvaluepair">
  <keyValuePair key="trusted_source" uri="https://github.com/sbmlteam/" value="42"/>
  <keyValuePair key="toolset" uri="urn:awesometool.com:keyvaluepair" value="expert"/>
  <keyValuePair key="score" value="0.9"/>
  <keyValuePair key="xfactor"/>
 </listOfKeyValuePairs>
</annotation>
```

As such, it is analogous and supplemental to the official **SBML** RDF annotation used to support MIRIAM compliant annotations. The **ListOfKeyValuePairs** is an optional element of an **Annotation** that declares the `xmlns` attribute to be "`http://sbml.org/fbc/keyvaluepair`" and then strictly applies the **ListOfKeyValuePairs** format. As it is defined as an **Annotation**, software may choose to support reading and interpreting the **ListOfKeyValuePairs** content and optionally ignore the annotation and merely read and write it with any other third party **Annotation**s. As is the case with the RDF/MIRIAM **Annotation**s, read and write support for the **ListOfKeyValuePairs** annotation is included in the **SBML** support libraries and easily accessible to tool developers.

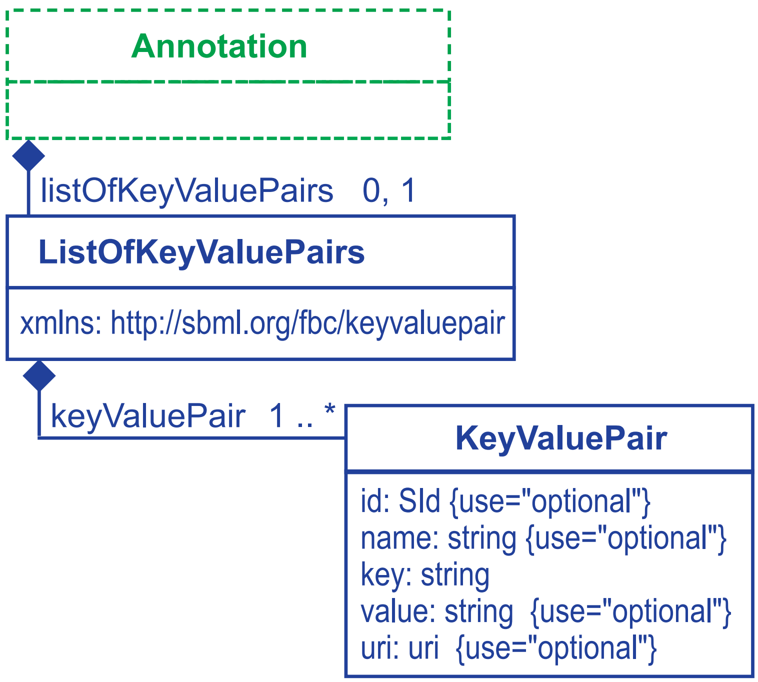

Figure 11: *A UML representation of the **SBML SBase** class extended in the Flux Balance Constraints package by the **ListOfKeyValuePairs**. See Section 1.3 for conventions related to this figure.*

The **ListOfKeyValuePairs** functions like a typical **SBML ListOf___** class with the restriction that it must contain one or more elements of type **KeyValuePair** (see Section 3.17). In addition it defines a single mandatory attribute, `xmlns`, which identifies the annotation as belonging to the Flux Balance Constraints package.

The `xmlns` attribute

The `xmlns` is a mandatory component of the **ListOfKeyValuePairs** of type `uri`. It takes a single value: `http://sbml.org/fbc/keyvaluepair`.

3.17 The FBC KeyValuePair class

The FBC **KeyValuePair** class defines a typical key–value pair with a single, mandatory attribute, the `key`, as well as four optional attributes: `value`, `uri`, `id` and `name`.

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

A **KeyValuePair** defines two optional attributes: `id` an attribute of type `SId` and `name` of type `string`.

The `key` *attribute*

The `key` is the mandatory component of the **KeyValuePair** pair and is of type `string`. It has the special property that every `key` in an enclosing **ListOfKeyValuePairs** must be unique.

The `value` *attribute*

The optional `value` attribute is of type `string` and contains the matching `value` associated with a particular `key`. If not present, the **KeyValuePair** is defined as having no value.

The `uri` *attribute*

The optional attribute `uri` is of type `uri`. The `uri` points to a resource that defines the meaning of the `key` component of the **KeyValuePair**, see Table 3 for examples of the types of values a `uri` may take. As a best practice, it is highly recommended that tools implementing a **KeyValuePair** also implement support for the `uri` attribute.

Type	Example	Description
urn	urn:awesometool.com:keyvaluepair	a tool specific namespace declaration
url	`https://github.com/awesometool/keyvaluepair.md`	a **url** containing a set of `key` definitions

Table 3: *Examples of how the* `uri` *attribute can be used to identify* `key` *definitions by way of a* ***urn*** *or* ***url****.*

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

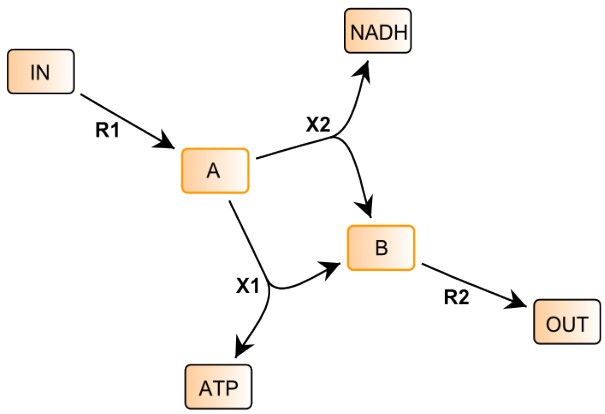

Figure 12: *FBC syntax example: a simple four reaction pathway. The reactions are R1, R2, X1, X2 with fixed species IN, OUT, ATP, NADH and variable species A, B.*

As shown in Figure 12 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 and CBMPy (Bergmann, 2012; Olivier, 2021). 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 4 where each reaction is represented as a column. Each row in the stoichiometric matrix represents the differential equation describing the change of a variable **Species**. In **SBML** such variable **Species** have the attribute `boundaryCondition` set to "`false`". This is in contrast to "fixed species" (where the attribute `boundaryCondition` is set to "`true`") which do not appear in the stoichiometric matrix. Please see the SBML Level 3 Version 1 Core specification for more details.

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

Table 4: *Example one: stoichiometric matrix,* $\mathbf{N}$

While the stoichiometry contains the structural properties of the reaction network the full description of a biological model can, for example, 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, $\mathbf{N}$ and the vector of reaction rates, $\mathbf{v}$).

$$\frac{ds}{dt} = \mathbf{N}\mathbf{v} \tag{5}$$

The formulation of the kinetic model, as shown in Equation 5 is typical of the kind that can already be described using SBML Level 3 Version 1 Core where the vector $\mathbf{v}$ 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 6):

$$\mathbf{N}\mathbf{J} = 0 \tag{6}$$

Note that the rate vector $\mathbf{v}$ is now represented as the steady-state flux vector $\mathbf{J}$. 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 function 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 commonly used IBM™ CPLEX format.

4.1.2 Flux Bounds

A capacity constraint or flux bound describes the limits that the flux through a certain reaction may attain at steady state. In this example the maximum limit (upper bound) on the flux through reaction *R1* is set to be one with the minimum value (lower bound) set to zero (with an arbitrary unit of flux). In LP format this may be written as:

```
Bounds
R1 <= 1.0
R1 >= 0.0
```

the same information encoded as XML:

```
<listOfParameters>
 <parameter id="R1l" constant="true" value="0" sboTerm="SBO:0000625" />
 <parameter id="R1u" constant="true" value="1" sboTerm="SBO:0000625" />
</listOfParameters>

...

<reaction id="R1" reversible="false" fast="false"
  fbc:upperFluxBound="R1l" fbc:upperFluxBound="R1u">
 <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:variableType="linear"/>
  </fbc:listOfFluxObjectives>
 </fbc:objective>
</fbc:listOfObjectives>
```

4.1.4 Complete worked example

To conclude we show how the complete model described in Figure 12 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
0.0 <= 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 7, with one possible solution for **J**.

$$\begin{pmatrix} 1 & 0 & -1 & -1 \\ 0 & -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1.0 \\ \mathbf{1.0} \\ 0.0 \\ 1.0 \end{pmatrix} = \mathbf{0} \qquad (7)$$

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/version3"
  level="3" version="1" fbc:required="false" sboTerm="SBO:0000624">
 <model id="fbcSpecExample1" timeUnits="time" fbc:strict="true">
  <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" constant="false" boundaryCondition="true"/>
   <species id="OUT" compartment="compartment" initialConcentration="0" substanceUnits="substance"
    hasOnlySubstanceUnits="false" constant="false" boundaryCondition="true"/>
   <species id="A" compartment="compartment" initialConcentration="0" substanceUnits="substance"
    hasOnlySubstanceUnits="false" constant="false" boundaryCondition="false"/>
   <species id="B" compartment="compartment" initialConcentration="0" substanceUnits="substance"
    hasOnlySubstanceUnits="false" constant="false" boundaryCondition="false"/>
   <species id="ATP" compartment="compartment" initialConcentration="0" substanceUnits="substance"
    hasOnlySubstanceUnits="false" constant="false" boundaryCondition="true"/>
   <species id="NADH" compartment="compartment" initialConcentration="0" substanceUnits="substance"
    hasOnlySubstanceUnits="false" constant="false" boundaryCondition="true"/>
  </listOfSpecies>
  <listOfParameters>
   <parameter id="R1u" constant="true" value="1" sboTerm="SBO:0000625" />
   <parameter constant="true" id="irrLow" value="0" sboTerm="SBO:0000625" />
   <parameter constant="true" id="posInf" value="INF" sboTerm="SBO:0000626" />
  </listOfParameters>
  <listOfReactions>
   <reaction id="R1" reversible="false" fast="false"
    fbc:lowerFluxBound="irrLow" 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="irrLow" fbc:upperFluxBound="posInf" >
    <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="irrLow" fbc:upperFluxBound="posInf" >
    <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="irrLow" fbc:upperFluxBound="posInf" >
    <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:variableType="linear"/>
    </fbc:listOfFluxObjectives>
```

```
    </fbc:objective>
  </fbc:listOfObjectives>
 </model>
</sbml>
```

5 Best practices

In this section, we illustrate a number of practices for using and interpreting various constructs in the Flux Balance Constraints package. These recommendations are non-normative, ignoring them will not render a model invalid, rather they are suggested as ways of enhancing model inter-operability. For a description of the additional SBO terms relevant to FBC models please see the SBO appendix.

5.1 Examples contrasting the 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 SBML Level 2 Version 1 format (Becker et al., 2007; Reed et al., 2003; Schellenberger et al., 2010) that have been translated into SBML Level 3 Version 1 FBC Version 2.

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:variableType="linear"/>
  </fbc:listOfFluxObjectives>
 </fbc:objective>
</fbc:listOfObjectives>
```

Species definition

SBML Level 2 Species annotation version 1

Examine the **SBML** Level 2 Version 1 **Species** definition. Note how the `name` attribute is overloaded with the chemical formula in a tool specific way.

```
<species id="M_atp_c" name="ATP_C10H12N5O13P3"
  compartment="Cytosol" charge="-4" />
```

SBML Level 2 Species annotation version 2

A variation of the previous syntax that appeared in later models.

```
<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 FBC Species attributes

With the adoption of **SBML** FBC these **Species** properties can now 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 FBC Reaction

As an example of a good annotation practice the EC number stored in the **Notes** element has been converted into MIRIAM compliant RDF. The Flux Balance Constraints package also facilitates the structured definition and use of gene protein associations and flux capacity constraints.

```xml
<reaction metaid="meta_R_GTHS" id="R_GTHS" name="glutathione␣synthetase" reversible="false"
          fbc:lowerFluxBound="R_GTHS_l" fbc:upperFluxBound="R_GTHS_u">
 <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:bqbiol="http://biomodels.net/biology-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:geneProductAssociation fbc:id="gpr_GTHS">
  <fbc:geneProductRef fbc:geneProduct="g_b2947"/>
 </fbc:geneProductAssociation>
 <listOfReactants>
  <speciesReference constant="true" species="M_atp_c" stoichiometry="1"/>
  <speciesReference constant="true" species="M_glucys_c" stoichiometry="1"/>
  <speciesReference constant="true" species="M_gly_c" stoichiometry="1"/>
  </listOfReactants>
 <listOfProducts>
  <speciesReference constant="true" species="M_adp_c" stoichiometry="1"/>
  <speciesReference constant="true" species="M_gthrd_c" stoichiometry="1"/>
  <speciesReference constant="true" species="M_h_c" stoichiometry="1"/>
  <speciesReference constant="true" species="M_pi_c" stoichiometry="1"/>
 </listOfProducts>
</reaction>
<fbc:listOfGeneProducts>
  <fbc:geneProduct metaid="meta_g_b2947" fbc:id="g_b2947" fbc:label="b2947">
   <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:bqbiol="http://biomodels.net/biology-qualifiers/"
     <rdf:Description rdf:about="#meta_g_b2947">
      <bqbiol:isEncodedBy>
       <rdf:Bag>
        <rdf:li rdf:resource="http://identifiers.org/ncbigene/947445"/>
       </rdf:Bag>
      </bqbiol:isEncodedBy>
     </rdf:Description>
    </rdf:RDF>
   </annotation>
  </fbc:geneProduct>
</fbc:listOfGeneProducts>
<listOfParameters>
 <parameter id="R_GTHS_l" constant="true" sboTerm="SBO:0000625" value="0" sboTerm="SBO:0000625" />
 <parameter metaid="meta_R_GTHS_u" id="R_GTHS_u" constant="true" value="inf" sboTerm="SBO:0000625">
   <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:bqbiol="http://biomodels.net/biology-qualifiers/"
     <rdf:Description rdf:about="#meta_R_GTHS_u">
      <bqbiol:isDescribedBy><rdf:Bag>
       <rdf:li rdf:resource="http://identifiers.org/pmc/PMC193654"/>
      </rdf:Bag></bqbiol:isDescribedBy>
     </rdf:Description>
    </rdf:RDF>
   </annotation>
 </parameter>
</listOfParameters>
```

5.2 An example of a strict FBC Version 2 model (XML)

This section highlights the best practices for a complete FBC Version 2 model. To improve readability, detailed annotations as described in Section Section 5.1 and unit definitions have been omitted.

```
<?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/version3"
  level="3" version="1" fbc:required="false" sboTerm="SBO:0000624">
 <model id="fbcSpecExample1" timeUnits="time" fbc:strict="true">
  <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="lb" name="arbitrary␣lower␣bound" constant="true" value="-INF"
    sboTerm="SBO:0000626" />
   <parameter id="ub" name="arbitrary␣upper␣bound" constant="true" value="INF"
    sboTerm="SBO:0000626" />
   <parameter id="R1l" name="lower␣bound" constant="true" value="0"
    sboTerm="SBO:0000625" />
   <parameter id="R1u" name="uptake␣upper␣bound" constant="true" value="1"
    sboTerm="SBO:0000625" />
   <parameter id="X2l" name="export␣only␣lower␣bound" constant="true" value="0"
    sboTerm="SBO:0000625" />
  </listOfParameters>
  <fbc:listOfGeneProducts>
   <fbc:geneProduct fbc:id="g1" fbc:name="dog␣gene" fbc:label="PetGeneDB1"/>
   <fbc:geneProduct fbc:id="g2" fbc:name="cat␣gene" fbc:label="PetGeneDB2"/>
   <fbc:geneProduct fbc:id="g3" fbc:name="mouse␣gene" fbc:label="PetGeneDB3"/>
   <fbc:geneProduct fbc:id="g4" fbc:name="bird␣gene" fbc:label="PetGeneDB4"/>
  </fbc:listOfGeneProducts>
  <listOfReactions>
   <reaction id="R1" reversible="false" fast="false"
     fbc:lowerFluxBound="R1l" fbc:upperFluxBound="R1u" >
    <fbc:geneProductAssociation>
     <fbc:geneProductRef fbc:geneProduct="g1"/>
    </fbc:geneProductAssociation>
    <listOfReactants>
     <speciesReference species="IN" stoichiometry="1" constant="true"/>
    </listOfReactants>
    <listOfProducts>
     <speciesReference species="A" stoichiometry="1" constant="true"/>
    </listOfProducts>
   </reaction>
   <reaction id="R2" reversible="true" fast="false"
     fbc:lowerFluxBound="lb" fbc:upperFluxBound="ub">
    <fbc:geneProductAssociation fbc:id="andGPR">
     <fbc:and>
      <fbc:geneProductRef fbc:geneProduct="g1"/>
      <fbc:geneProductRef fbc:geneProduct="g4"/>
     </fbc:and>
    </fbc:geneProductAssociation>
```

```
    <listOfReactants>
     <speciesReference species="B" stoichiometry="1" constant="true"/>
    </listOfReactants>
    <listOfProducts>
     <speciesReference species="OUT" stoichiometry="1" constant="true"/>
    </listOfProducts>
   </reaction>
   <reaction id="X1" reversible="true" fast="false"
     fbc:lowerFluxBound="lb" fbc:upperFluxBound="ub" >
    <fbc:geneProductAssociation fbc:id="orGPR">
     <fbc:or>
      <fbc:geneProductRef fbc:geneProduct="g2"/>
      <fbc:geneProductRef fbc:geneProduct="g3"/>
     </fbc:or>
    </fbc:geneProductAssociation>
    <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="true" fast="false"
     fbc:lowerFluxBound="X2l" fbc:upperFluxBound="ub" >
    <fbc:geneProductAssociation fbc:id="allGPR">
     <fbc:or>
      <fbc:and>
       <fbc:geneProductRef fbc:geneProduct="g1"/>
       <fbc:geneProductRef fbc:geneProduct="g4"/>
      </fbc:and>
      <fbc:and>
       <fbc:geneProductRef fbc:geneProduct="g1"/>
       <fbc:geneProductRef fbc:geneProduct="g3"/>
      </fbc:and>
     </fbc:or>
    </fbc:geneProductAssociation>
    <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:variableType="linear"/>
    </fbc:listOfFluxObjectives>
   </fbc:objective>
  </fbc:listOfObjectives>
 </model>
</sbml>
```

5.3 Key value pairs in FBC Version 3

The **KeyValuePair** construct should only be used for annotations that cannot be encoded with either a different SBML L3 Package or the standard format for annotations used within SBML.

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 FBC 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 FBC Package 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 FBC Package 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 FBC Package 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 FBC Package 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 FBC Package namespace.

General rules about this package

fbc-10101 ☑ To conform to the FBC Package specification for SBML Level 3 Version 1, an SBML document must declare "`http://www.sbml.org/sbml/level3/version1/fbc/version3`" as the XMLNamespace to use for elements of this package. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3 Section 3.1 on page 6.)

fbc-10102 ☑ Wherever they appear in an SBML document, elements and attributes from the FBC Package must use the "`http://www.sbml.org/sbml/level3/version1/fbc/version3`" namespace, declaring so either explicitly or implicitly. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3 Section 3.1 on page 6.)

General rules about identifiers

fbc-10301 ☑ (Extends validation rule #10301 in the 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 **GeneProduct**, **Objective**, **FluxObjective**, **GeneProductAssociation**, **UserDefinedConstraint** and UserDefinedConstraintComponent objects defined by the Flux Balance Constraints package. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.2 on page 6.)

fbc-10302 ☑ The value of a `fbc:id` must conform to the syntax of the **SBML** data type `SId` (Reference: SBML Level 3 Version 1 Core, Section 3.1.7.)

Rules for the extended SBML class

fbc-20101 ☑ In all SBML documents using the FBC Package, the **SBML** object must have the `fbc:required` attribute. (Reference: 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 data type `boolean`. (Reference: 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`". (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3 Section 3.1 on page 6.)

Rules for extended Model object

fbc-20201 ☑ A **Model** object may contain one and only one instance of each of the **ListOfObjectives**, **ListOfFluxBounds**, **ListOfGeneProducts** and **ListOfUserDefinedConstraints** elements. No other elements from the SBML Level 3 Flux Balance Constraints namespaces are permitted on a **Model** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, 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**, **ListOfGeneProducts**, **ListOfObjectives** and ListOfUserDefinedConstraints. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20203 ☑ (This validation rule does not apply in Flux Balance Constraints Version 3.)

fbc-20204 ☑ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfObjectives** container object may only contain **Objective** objects. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, **??** on page ??.)

fbc-20205 ☑ (This validation rule does not apply in Flux Balance Constraints Version 3.)

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 `fbc: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 3, 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 3, Section 3.3.1 on page 9).

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 3, Section 3.3.1 on page 9.)

fbc-20209 ☑ A **Model** object must have the required attributes `fbc:strict` and `fbc:strict`. No other attributes from the SBML Level 3 Flux Balance Constraints namespaces are permitted on a **Model** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20210 ☑ The attribute `fbc:strict` on a **Model** must have a value of data type `boolean`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20211 ☑ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfGeneProducts** container object may only contain **GeneProduct** objects. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20212 ☑ A **ListOfGeneProducts** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfGeneProducts** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20213 ☑ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfUserDefinedConstraints** container object may only contain **UserDefinedConstraint** objects. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20214 ☑ A **ListOfUserDefinedConstraints** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfUserDefinedConstraints** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

Rules for extended Species object

fbc-20301 ☑ A **Species** object may have the optional attributes `fbc:charge` and `fbc:chemicalFormula`. No other attributes from the SBML Level 3 Flux Balance Constraints namespaces are permitted on a **Species** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.4 on page 9.)

fbc-20302 ☑ (This validation rule does not apply in Flux Balance Constraints Version 3.)

fbc-20303 ☑ The attribute `fbc:chemicalFormula` on a **Species** must have a value of data type `string`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.4 on page 9.)

fbc-20304 ☑ The attribute `fbc:charge` on a **Species** must have a value of data type `double`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.4 on page 9.)

Rules for Objective object

fbc-20501 ☑ An **Objective** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on an **Objective**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20502 ☑ An **Objective** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on an **Objective**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20503 ☑ An **Objective** object must have the required attributes `fbc:id` and `fbc:type`, and may have the optional attributes `fbc:name`. No other attributes from the SBML Level 3 Flux Balance Constraints namespaces are permitted on an **Objective** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.6 on page 11.)

fbc-20504 ☑ The attribute `fbc:name` on an **Objective** must have a value of data type `string`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.6 on page 11.)

fbc-20505 ☑ The value of the attribute `fbc:type` of an **Objective** object must conform to the syntax of SBML data type `FbcType` and may only take on the allowed values of `FbcType` defined in SBML; that is, the value must be one of the following: "`maximize`" or "`minimize`". (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.6 on page 11.)

fbc-20506 ☑ An **Objective** object may contain one and only one instance of the **ListOfFluxObjectives** elements. No other elements from the SBML Level 3 Flux Balance Constraints namespaces are permitted on an **Objective** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.6 on page 11.)

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

fbc-20509 ☑ A **ListOfFluxObjectives** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfFluxObjectives** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, 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 namespaces are permitted on a **FluxObjective**. (Reference: 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 namespaces are permitted on a **FluxObjective**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20603 ☑ A **FluxObjective** object must have the required attributes `fbc:reaction`, `fbc:coefficient` and `fbc:variableType`, and may have the optional attributes `fbc:id` and `fbc:name`. No other attributes from the SBML Level 3 Flux Balance Constraints namespaces are permitted on a **FluxObjective** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.7 on page 13.)

fbc-20604 ☑ The attribute `fbc:name` on a **FluxObjective** must have a value of data type `string`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.7 on page 13.)

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 3, Section 3.7 on page 13.)

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. (Reference: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.7 on page 13.)

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 3, Section 3.7 on page 13.)

fbc-20608 ☑ When the value of the **Model**s `fbc:strict` attribute is "`true`", the value of the attribute `fbc:coefficient` of a **FluxObjective** object must not be set to "`NaN`", "`-INF`" or "`INF`". (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20609 ☑ The value of the attribute `fbc:variableType` of a **FluxObjective** object must conform to the syntax of SBML data type `FbcVariableType` and may only take on the allowed values of `FbcVariableType` defined in SBML; that is, the value must be one of the following: "`linear`" or "`quadratic`". (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.7 on page 13.)

Rules for extended Reaction object

fbc-20701 ☑ A **Reaction** object may contain one and only one instance of the **GeneProductAssociation** element. No other elements from the SBML Level 3 Flux Balance Constraints namespaces are permitted on a **Reaction** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.8 on page 14.)

fbc-20702 ☑ A **Reaction** object may have the optional attributes `fbc:lowerFluxBound` and `fbc:upperFluxBound`. No other attributes from the SBML Level 3 Flux Balance Constraints namespaces are permitted on a **Reaction** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.8 on page 14.)

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 3, Section 3.8 on page 14.)

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 3, Section 3.8 on page 14.)

fbc-20705 ☑ The value of the attribute `fbc:lowerFluxBound` of a **Reaction** object must be the identifier of an existing **Parameter** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.8 on page 14.)

fbc-20706 ☑ The value of the attribute `fbc:upperFluxBound` of a **Reaction** object must be the identifier of an existing **Parameter** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.8 on page 14.)

fbc-20707 ☑ When the value of the **Model**s `fbc:strict` attribute is "`true`", a **Reaction** must define the attributes `fbc:lowerFluxBound` and `fbc:upperFluxBound`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20708 ☑ When the value of the **Model**s `fbc:strict` attribute is "`true`", the **Parameter** objects referred to by the attributes `fbc:lowerFluxBound` and `fbc:upperFluxBound` must have their `constant` attribute set to "`true`". (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20709 ☑ When the value of the **Model**s `fbc:strict` attribute is "`true`", the **Parameter** objects referred to by the attributes `fbc:lowerFluxBound` and `fbc:upperFluxBound` must have a defined value for their `value` attribute, which may not be "`NaN`". (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20710 ☑ When the value of the **Model**s `fbc:strict` attribute is "`true`", the **Parameter** objects referred to by the attributes `fbc:lowerFluxBound` and `fbc:upperFluxBound` may not be targeted by an **InitialAssignment**. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20711 ☑ When the value of the **Model**s `fbc:strict` attribute is "`true`", the value of the **Parameter** object referred to by the attribute `fbc:lowerFluxBound` may not have the value "`INF`". (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20712 ☑ When the value of the **Model**s `fbc:strict` attribute is "`true`", the value of the **Parameter** object referred to by the attribute `fbc:upperFluxBound` may not have the value "`-INF`". (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20713 ☑ When the value of the **Model**s `fbc:strict` attribute is "`true`", the value of the **Parameter** object referred to by the attribute `fbc:lowerFluxBound` must be less than or equal to the value of the **Parameter** object referred to by the attribute `fbc:upperFluxBound` . (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20714 ☑ When the value of the **Model**s `fbc:strict` attribute is "`true`", the `constant` attribute of **SpeciesReference** elements of a **Reaction** must be set to "`true`". (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20715 ☑ When the value of the **Model**s `fbc:strict` attribute is "`true`", the value of a **SpeciesReference**'s `stoichiometry` attribute must not be set to "`NaN`", "`-INF`" or "`INF`". (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

fbc-20716 ☑ When the value of the **Model**s `fbc:strict` attribute is "`true`", the **SpeciesReference** elements of a **Reaction** may not be targeted by an **InitialAssignment**. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.3 on page 8.)

Rules for GeneProductAssociation object

fbc-20801 ☑ A **GeneProductAssociation** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **GeneProductAssociation**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

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

fbc-20803 ☑ A **GeneProductAssociation** object may have the optional attributes `fbc:id` and `fbc:name`. No other attributes from the SBML Level 3 Flux Balance Constraints namespaces are permitted on a **GeneProductAssociation** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.9 on page 15.)

fbc-20804 ☑ The attribute `fbc:id` on a **GeneProductAssociation** must be of the data type `SId`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.9 on page 15.)

fbc-20805 ☑ A **GeneProductAssociation** 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 3, Section 3.9 on page 15.)

fbc-20806 ☑ The attribute `fbc:name` on a **GeneProductAssociation** must be of the data type `string`. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.9 on page 15.)

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 namespaces are permitted on a **GeneProductRef**. (Reference: 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 namespaces are permitted on a **GeneProductRef**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-20903 ☑ A **GeneProductRef** object must have the required attributes `fbc:geneProduct` and may have the optional attributes `fbc:id` and `fbc:name`. No other attributes from the SBML Level 3 Flux Balance Constraints namespaces are permitted on a **GeneProductRef** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.11 on page 17.)

fbc-20904 ☑ 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 3, Section 3.11 on page 17.)

fbc-20908 ☑ The value of the attribute `fbc:geneProduct` of a **GeneProductRef** object must be the identifier of an existing **GeneProduct** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.11 on page 17.)

Rules for And object

fbc-21001 ☑ A **And** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **And**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21002 ☑ A **And** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **And**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21003 ☑ 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 3, Section 3.12 on page 18.)

Rules for Or object

fbc-21101 ☑ A **Or** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **Or**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21102 ☑ A **Or** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **Or**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21103 ☑ 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 3, Section 3.13 on page 18.)

Rules for GeneProduct object

fbc-21201 ☑ A **GeneProduct** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **GeneProduct**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21202 ☑ A **GeneProduct** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **GeneProduct**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21203 ☑ A **GeneProduct** object must have the required attributes `fbc:id` and `fbc:label`, and may have the optional attributes `fbc:name` and `fbc:associatedSpecies`. No other attributes from the SBML Level 3 Flux

Balance Constraints namespaces are permitted on a **GeneProduct** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.5 on page 10.)

fbc-21204 ☑ The attribute `fbc:label` on a **GeneProduct** must have a value of data type `string`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.5 on page 10.)

fbc-21205 ☑ The attribute `fbc:label` on a **GeneProduct** must be unique among the set of all **GeneProduct** elements defined in the **Model**. (References: SBML Level 3 Package Specification for Flux Balance Constraints, Version 3, Section 3.5 on page 10.)

fbc-21206 ☑ The attribute `fbc:name` on a **GeneProduct** must have a value of data type `string`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.5 on page 10.)

fbc-21207 ☑ The value of the attribute `fbc:associatedSpecies` of a **GeneProduct** object must be the identifier of an existing **Species** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.5 on page 10.)

Rules for UserDefinedConstraintComponent object

fbc-21301 ☑ An **UserDefinedConstraintComponent** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on an **UserDefinedConstraintComponent**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21302 ☑ An **UserDefinedConstraintComponent** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on an **UserDefinedConstraintComponent**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21303 ☑ An **UserDefinedConstraintComponent** object must have the required attributes `fbc:coefficient`, `fbc:variable` and `fbc:variableType`, and may have the optional attributes `fbc:id` and `fbc:name`. No other attributes from the SBML Level 3 Flux Balance Constraints namespaces are permitted on an **UserDefinedConstraintComponent** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.15 on page 19.)

fbc-21304 ☑ The attribute `fbc:coefficient` on an **UserDefinedConstraintComponent** must have a value of data type `double`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.15 on page 19.)

fbc-21305 ☑ The value of the attribute `fbc:variable` of an **UserDefinedConstraintComponent** object must be the identifier of an existing **Reaction** or **Parameter** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.15 on page 19.)

fbc-21306 ☑ The value of the attribute `fbc:variableType` of an **UserDefinedConstraintComponent** object must conform to the syntax of SBML data type `FbcVariableType` and may only take on the allowed values of `FbcVariableType` defined in SBML; that is, the value must be one of the following: "`linear`" or "`quadratic`". (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.15 on page 19.)

fbc-21307 ☑ The attribute `fbc:name` on an **UserDefinedConstraintComponent** must have a value of data type `string`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.15 on page 19.)

Rules for UserDefinedConstraint object

fbc-21401 ☑ An **UserDefinedConstraint** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on an **UserDefinedConstraint**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21402 ☑ An **UserDefinedConstraint** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on an **UserDefinedConstraint**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21403 ☑ An **UserDefinedConstraint** object must have the required attributes `fbc:lowerBound` and `fbc:upperBound`, and may have the optional attributes `fbc:id` and `fbc:name`. No other attributes from the SBML Level 3 Flux Balance Constraints namespaces are permitted on an **UserDefinedConstraint** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.14 on page 18.)

fbc-21404 ☑ An **UserDefinedConstraint** object must contain one and only one instance of the **ListOfUserDefinedConstraintComponents** element. No other elements from the SBML Level 3 Flux Balance Constraints namespaces are permitted on an **UserDefinedConstraint** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.14 on page 18.)

fbc-21405 ☑ The value of the attribute `fbc:lowerBound` of an **UserDefinedConstraint** object must be the identifier of an existing **Parameter** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.14 on page 18.)

fbc-21406 ☑ The value of the attribute `fbc:upperBound` of an **UserDefinedConstraint** object must be the identifier of an existing **Parameter** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.14 on page 18.)

fbc-21407 ☑ The attribute `fbc:name` on an **UserDefinedConstraint** must have a value of data type `string`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.14 on page 18.)

fbc-21408 ☑ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfUserDefinedConstraintComponents** container object may only contain **UserDefinedConstraintComponent** objects. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.14 on page 19.)

fbc-21409 ☑ A **ListOfUserDefinedConstraintComponents** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfUserDefinedConstraintComponents** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.14 on page 19.)

Rules for KeyValuePair object

fbc-21501 ☑ A **KeyValuePair** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **KeyValuePair**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21502 ☑ A **KeyValuePair** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **KeyValuePair**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

fbc-21503 ☑ A **KeyValuePair** object must have the required attribute `fbc:key`, and may have the optional attributes `fbc:id`, `fbc:name`, `fbc:value` and `fbc:uri`. No other attributes from the SBML Level 3 Flux Balance Constraints namespaces are permitted on a **KeyValuePair** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.17 on page 22.)

fbc-21504 ☑ The attribute `fbc:key` on a **KeyValuePair** must have a value of data type `string`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.17 on page 22.)

fbc-21505 ☑ The attribute `fbc:name` on a **KeyValuePair** must have a value of data type `string`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.17 on page 22.)

fbc-21506 ☑ The attribute `fbc:value` on a **KeyValuePair** must have a value of data type `string`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.17 on page 22.)

fbc-21507 ☑ The attribute `fbc:uri` on a **KeyValuePair** must have a value of data type `string`. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.17 on page 22.)

fbc-201508 ☑ A **ListOfKeyValuePairs** object must have the required attribute `fbc:xmlns`. No other attributes from the SBML Level 3 Flux Balance Constraints namespaces are permitted on a **ListOfKeyValuePairs** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.16 on page 21.)

fbc-21509 ☑ A **ListOfKeyValuePairs** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfKeyValuePairs** object. (Reference: SBML Level 3 Specification for Flux Balance Constraints, Version 3, Section 3.16 on page 21.)

B The Systems Biology Ontology and the sboTerm attribute

The following text on the usage of SBO has been extracted from the SBML Level 3 Version 1 Core specification and is provided here for your convenience. Please consult the official documentation for more details. Table 5 shows the additional SBO terms specified by the Flux Balance Constraints package.

SBML Element	SBO Name	SBO term
Document	flux balance framework	SBO:0000624
Parameter	flux bound	SBO:0000625
Parameter	default flux bound	SBO:0000626

Table 5: *SBML components and the additional types of SBO terms specified in the Flux Balance Constraints package that may be assigned to them. Note that the important aspect here is the set of specific SBO identifiers, not the SBO term names, because the names may change as SBO continues to evolve. See the text and SBO online resources for detailed descriptions on the meaning and usage of these terms.*

The values of `id` attributes on SBML components allow the components to be cross-referenced within a model. The values of `name` attributes on SBML components provide the opportunity to assign them meaningful labels suitable for display to humans. The specific identifiers and labels used in a model necessarily must be unrestricted by SBML, so that software and users are free to pick whatever they need. However, this freedom makes it more difficult for software tools to determine, without additional human intervention, the semantics of models more precisely than the semantics provided by the SBML object classes defined in other sections of this document.

An approach to solving this problem is to associate model components with terms from carefully curated controlled vocabularies (CVs). This is the purpose of the optional `sboTerm` attribute provided on the SBML class ***SBase***. The `sboTerm` attribute always refers to terms belonging to the Systems Biology Ontology[1]. In this section, we discuss the `sboTerm` attribute, SBO, the motivations and theory behind their introduction, and guidelines for their use.

SBO is not part of SBML; it is being developed separately, to allow the modeling community to evolve the ontology independently of SBML. However, the terms in the ontology are being designed keeping SBML components in mind, and are classified into subsets that can be directly related with SBML components such as reaction rate expressions, parameters, and a few others, see below. The use of `sboTerm` attributes is optional, and the presence of `sboTerm` on an element does not change the way the model is *interpreted*. Annotating SBML elements with SBO terms adds additional semantic information that may be used to *convert* the model into another model, or another format. Although SBO support provides an important source of information to understand the meaning of a model, software does not need to support `sboTerm` to be considered SBML-compliant.

Although the use of SBO can be beneficial, it is critical to keep in mind that the presence of an `sboTerm` value on an object *must not change the fundamental mathematical meaning* of the model. An SBML model must be defined such that it stands on its own and does not depend on additional information added by SBO terms for a correct mathematical interpretation. SBO term definitions will not imply any alternative mathematical semantics for any SBML object labeled with that term. Two important reasons motivate this principle. First, it would be too limiting to require all software tools to be able to understand the SBO vocabularies in addition to understanding SBML. Supporting SBO is not only additional work for the software developer; for some kinds of applications, it may not make sense. If SBO terms on a model are optional, it follows that the SBML model *must* remain unambiguous and fully interpretable without them, because an application reading the model may ignore the terms. Second, we believe allowing the use of `sboTerm` to alter the mathematical meaning of a model would allow too much leeway to shoehorn inconsistent concepts into SBML objects, ultimately reducing the interoperability of the models.

[1] http://biomodels.net/SBO/

Acknowledgments

First of all we would like to acknowledge the constraint-based modelling and **SBML** communities of researchers, sofware developers, modellers and anyone who has contributed to the adoption of the **SBML** FBC standard (too many to list here) by implementing support in your tools and using it to encode your models. In particular, we would like to thank everyone who contributed, in various ways, to the development of both the original proposal and this specification.

For financial/travel/technical support and highly productions discussions 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 to Thomas Pfau for contributing text to and critically reading the version 3 specification.

We also would like to thank (in alphabetical order) the people involved in the development of the Flux Balance Constraints package. Andreas Dräger, Ali Ebrahim, Ronan Fleming, Ben Heavner, Daniel Hyduke, Sarah Keating, Matthias Koenig, Nicolas Le Novère, Chris Myers, Thomas Pfau, Nicolas Rodriguez, Kieran Smallbone, Lucian Smith, Neil Swainston, Alex Thomas, all past and present members of the FBC Package Working Group and contributors to the discussion on the FBC mailing list, the libSBML and JSBML development teams and all others who contributed to discussions on various occasions. If we have not added you to this list and you feel you should be here please contact us.

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.

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/.

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. (2021). CBMPy: Constraint Based Modelling in Python. Available via the World Wide Web at https://doi.org/10.5281/zenodo.3358764.

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.

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/.