This document summarises an email discussion that took place, mostly, in June 2013 and now forms the focus point for an ongoing broader discussion. Initiated by Ben Heavner and Neil Swainston it includes Kieran Smallbone, Sarah Keating, Frank Bergmann and Brett Olivier. Many thanks to all participants for taking part in an extremely productive discussion. Brett 20130713

How to encode the COBRA annotations currently stored as SBML <notes> in SBML L3 FBC v2?

Introduction

With the acceptance of the SBML Level 3 package Flux Balance Constraints (SBML3FBC) it is possible to encode most of the “structural” features of existing COBRA models that are necessary for an FBA models analysis: reactions, species, objectives, chemical formula, charge and flux bounds. However two related sets of information that are widely used (and in some cases are required for COBRA) are not yet covered in the specification (1) the encoding of the so called gene-protein associations (2) a whole set of annotation which is currently stored as in the <notes> field of the SBML Reaction and Species object as XML. The former being dealt with elsewhere while this document will focus on issue 2.

The problem is thus

```
<reaction id="R_GLCRD" name="glucarate dehydratase" reversible="false">
 <notes>
  <html:p>Abbreviation: R_GLCRD</html:p>
  <html:p>Synonyms: _1</html:p>
  <html:p>EC Number: 4.2.1.40</html:p>
  <html:p>SUBSYSTEM: Alternate Carbon Metabolism</html:p>
  <html:p>Equation: [c] : glcr --&gt; 5dh4dglc + h2o</html:p>
  <html:p>Confidence Level: 4</html:p>
  <html:p>JLR</html:p>
  <html:p>genes:</html:p>
  <html:p>LOCUS:b2788#ABBREVIATION:gudX#</html:p>
  <html:p>LOCUS:b2787#ABBREVIATION:gudD#ECNUMBERS:4.2.1.40#</html:p>
  <html:p>proteins:</html:p>
  <html:p>NAME:D-glucarate dehydratase I#ABBREVIATION:GudD#</html:p>
  <html:p>NAME:putative D-glucarate dehydratase 2#ABBREVIATION:YgcY#</html:p>
  <html:p>GENE ASSOCIATION: (b2787) or (b2788)</html:p>
 </notes>
</reaction>
```

This is not the worst of it, Ben started putting together a table of annotations commonly found encoded in the GSR <notes> which starts to show the scope of the problem. It should be noted

that certain of these annotations such as EC number can be easily converted and stored as SBML <annotations> MIRIAM/RDF and ideally should be - the problem here is more of implementation in that (as I understand it) the *SBML Toolbox* and therefore *COBRA* does not currently deal well with <annotation>. While irritating this issue could be dealt with using existing tools such as *SBML Toolbox,* provided there is time and money made available for it.

A much bigger issue is how do you encode the things that are not currently encodable (think in context of automated conversion) SBML <annotation>?

Between a Lock and an RDF place

In discussing the aforementioned problem two positions crystalised that could form the basis of a solution one essentially RDF based, the other a custom annotation.

RDF (mostly quoted directly from an email by Neil)

“”“

The SBML limited subset of RDF is really verbose, with its unnecessary bags and what not. In reality, the following would be valid RDF, and would capture what we do now AND what we would like to take from the notes:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
  <rdf:Description rdf:about="#_metaid_0000052">
    <bqbiol:isVersionOf rdf:resource="http://identifiers.org/obo.eco/ECO:0000000"/>
    <bqbiol:is rdf:resource="http://identifiers.org/kegg.reaction:R00756"/>
    <bqbiol:is rdf:resource="http://identifiers.org/reactome:REACT_736"/>
    <bqbiol:isDescribedBy rdf:resource="http://identifiers.org/pubmed/1988962"/>
    <bqbiol:isVersionOf rdf:resource="http://identifiers.org/ec-code/2.3.1.21"/>
    <bqbiol:inchi>1/p+1/fH/q+1</bqbiol:inchi>
    <bqbiol:confidence rdf:datatype="xsd:integer">2</bqbiol:confidence>
  </rdf:Description>
</rdf:RDF>
```

This contains all of our existing annotations, and the "new" annotations, taken from the notes, BUT doesn't add additional overhead, as we already use RDF parsers, and can be extended, and can take advantage of everything that RDF gives us.

”””

One issue raised by Frank and Sarah is that while a limited subset of RDF is currently supported by libSBML there were no plans to support arbitrary RDF, using the existing mechanism, in the same way. To be more precise, libSBML will not require a full RDF parser in order to build. So libSBML would parse it as regular XML, and hope to match the given XML realization to an “agreed upon” format (just like with the MIRIAM RDF currently supported). Therefore any

additional RDF (additional annotation) should be dealt with by the tool (that is libSBML would provide convenience functions to retrieve the raw RDF, but would leave it to the tools to query the RDF). Thus, while it is a possibility to add such arbitrary RDF to a custom annotation (i.e. not touched by libSBML) this would have to be parsed by the tool, easier in some cases than others (e.g. SBML Toolbox).

KeyValueData (made up by Brett)

KeyValueData was created as a custom <annotation> to encode arbitrary data that may or may not be stored in the <notes> it has a simple syntax:

```
<annotation>
  <listOfKeyValueData xmlns="http://pysces.sourceforge.net/KeyValueData">
     <data id="subsystem" type="string" value="Alternate Carbon Metabolism"/>
     <data id="name" type="string" value="putative D-glucarate dehydratase 2#ABBREVIATION:YgcY#"/>
     <data id="locus" type="string" value="b2788#ABBREVIATION:gudX#"/>
     <data id="equation" type="string" value="[c] : glcr --&gt; 5dh4dglc + h2o"/>
     <data id="genes" type="string" value="None"/>
     <data id="proteins" type="string" value="None"/>
     <data id="confidence_level" type="string" value="4"/>
     <data id="abbreviation" type="string" value="R_GLCRD"/>
     <data id="synonyms" type="string" value="_1"/>
  </listOfKeyValueData>
</annotation>
```

This annotation was developed from a pragmatic perspective as it is relatively simple to read/write and fairly generic. Of course this is also its weakness as it does not attempt to address the annotation issue directly but rather provides a vehicle for a consensus mechanism, everyone needs to agree on the keys. Also, while parsable it is a custom annotation and does require some code (other than libSBML) to interpret. However, due to its flat structure can be done relatively easily with regular expressions (i.e. no tree parsing) and writing is trivial. Incidentally, adding software support for this structure in libSBML (and SBML Toolbox) should be fairly easy and could become part of the FBC package API.

Its life Jim, but not as we know it

Many options were discussed but in the end the two options above were discussed with a seemingly general leaning towards the point of view that while KeyValueData is sub-optimal it could be used as a way of at least getting annotation out the <notes>.

	RDF	KeyValueData	RDF parser in libSBML
Custom annotation[1]	possible	yes	
FBC package[2]	no	not discussed	
SBML Toolbox	no	no	
Other tool support[3]	no	yes	
Another feature			

Questions!

1. Does KeyValueData solve the problem?
2. Should KeyValueData remain an <annotation> or be part of the FBC package?
3. Could we expand KeyValueData to include additional features, e.g. annotation that would make it more useful?
4. Would a custom RDF <annotation> be more useful, is it practical?

[1] read/write by the tool
[2] read/write by libSBML
[3] currently existing