SBMLmerge, a System for Combining Biochemical Network Models

Marvin Schulz* schulzma@molgen.mpg.de
Jannis Uhlendorf* uhlndorf@molgen.mpg.de
Edda Klipp klipp@molgen.mpg.de
Wolfram Liebermeister lieberme@molgen.mpg.de

Max Planck Institute for Molecular Genetics, Ihnestraße 63-73, 14195 Berlin, Germany

Abstract

The Systems Biology Markup Language (SBML) is an XML-based format for representing mathematical models of biochemical reaction networks, and it is likely to become a main standard in the systems biology community. As published mathematical models in cell biology are growing in number and size, modular modelling approaches will gain additional importance. The main issue to be addressed in computer-assisted model combination is the specification and handling of model semantics.

The software SBMLmerge assists the user in combining models of biological subsystems to larger biochemical networks. First, the program helps the user in annotating all model elements with unique identifiers pointing to databases such as KEGG or Gene Ontology. Second, during merging, SBMLmerge detects and resolves various syntactic and semantic problems. Typical problems are conflicting variable names, elements which appear in more than one input model, and mathematical problems arising from the combination of equations. If the input models make contradicting statements about a biochemical quantity, the user is asked to choose between them. In the end the merging process results in a new, valid SBML model.

Keywords: model combination, SBML, MIRIAM, metabolic model

1 Introduction

The molecular processes in cells form a huge network, which makes detailed mathematical modelling and simulation extremely difficult. A potential strategy to obtain large cell models is to construct them "bottom-up" from smaller modules that can be fitted and understood more easily. Snoep *et al.* demonstrated this approach by combining a model of glycolysis with a model of the glyoxylate pathway [10]. A growing number of models is publicly available in databases such as Biomodels [6] or JWS [11]. For the future, we expect that more and more biochemical models will be constructed from smaller models. This requires both (i) standards for the representation of biochemical models (e.g. Systems Biology Markup Language (SBML) [2], CellML [8]) and (ii) standardised experimental conditions. In addition, modellers will also need efficient algorithms for model merging and for the detection of problems during the merging process.

We developed the software SBMLmerge to facilitate the bottom-up modelling approach, providing a general method for the combination of biochemical networks given in SBML format. The models to be integrated can contain biologically identical compounds or reactions. Out of these identical elements, only one representative must be retained in the output model. This is illustrated in Figure 1: the first reactions of glycolysis and glycogen production are merged by SBMLmerge. As requested, the common model elements appear only once in the output model.

*These authors have contributed equally to this work.

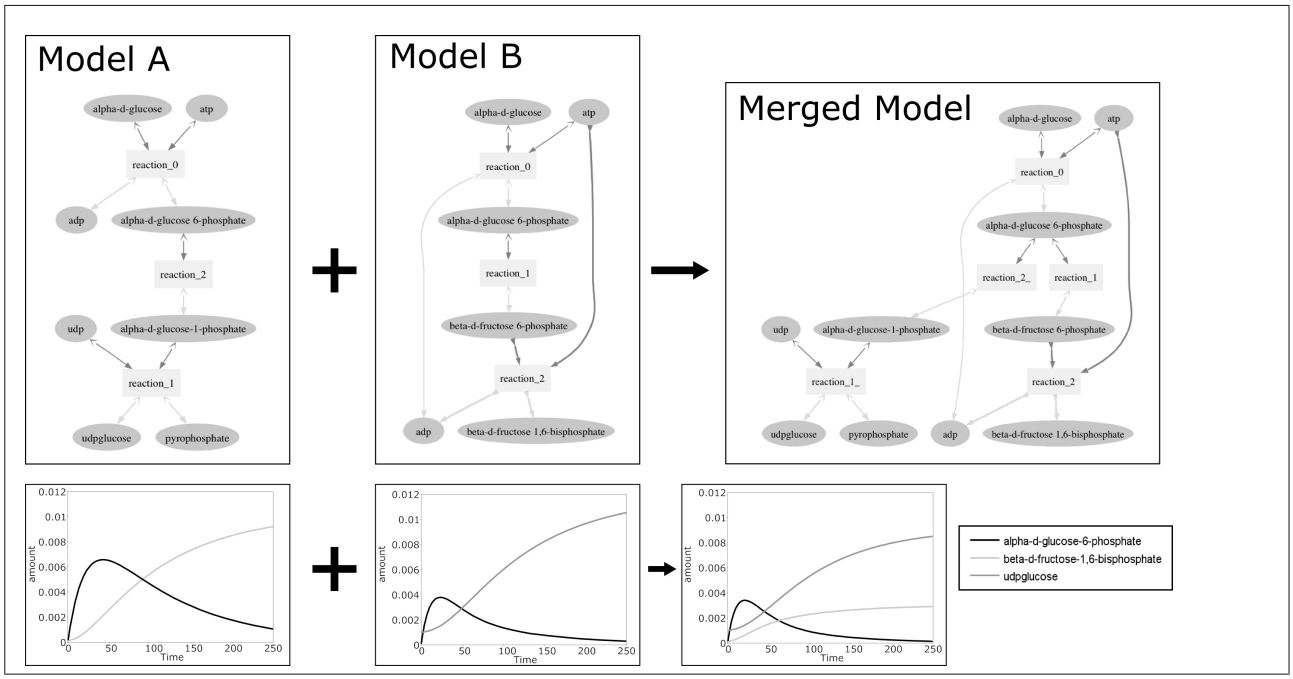

Figure 1: Merging of two small metabolic models. Both models represent the production of glucose-6-phosphate from glucose, followed by the first steps of either glycogen production or glycolysis. From these two models, SBMLmerge produces a merged model which contains the common chemical reaction only once. Model topologies were drawn using SBML2dot. After the merging, all three models were simulated with CellDesigner [3]. The kinetics of the input models have been chosen arbitrarily, and the resulting merged model shows the behaviour to be expected from these input models. Since the production of UDP-glucose in the second model is a little faster than the production of fructose-1,6-bisphosphate in the first model, UDP-glucose has a higher steady state concentration in the merged model. The steady state concentrations of both metabolites are lower in the merged model, since the available amount of glucose is split between them.

SBML [2] is a format for representing mathematical models of biochemical systems. It is based on XML [23] and will probably become a main standard for the exchange of such models. SBML describes biological processes independently of the type of mathematical description or analysis. Therefore, its structure represents the topology of the biochemical network rather than differential equations for metabolite concentrations. An SBML file declares a set of metabolites called *species*, which are connected by a set of chemical reactions. Mathematical expressions for reaction velocities can be specified in the MathML [22] format. If a metabolite is not transformed by any reaction, its concentration can also be described by differential or algebraic equations, so called *rules*. Furthermore it is possible to define cell compartments, which can contain different concentrations of the same metabolites.

2 Challenges when Merging SBML Files

Merging can only lead to a useful model if all input models are correct. There exist a variety of problems that can make a model invalid. For example, the model may contradict the biological reality due to wrong assumptions, for example reactions that do not exist in a certain organism. Also mathematical problems can arise, like under-/overdetermined equation systems, or algebraic equations which cannot be solved consecutively because they contain logical circles.

But also if the input models are valid, various problems can arise during merging, some of which are illustrated in Figure 2. First, there are *syntactical requirements* due to the structure of SBML, like the uniqueness of IDs within a model or the restriction that a variable cannot have multiple assignments.

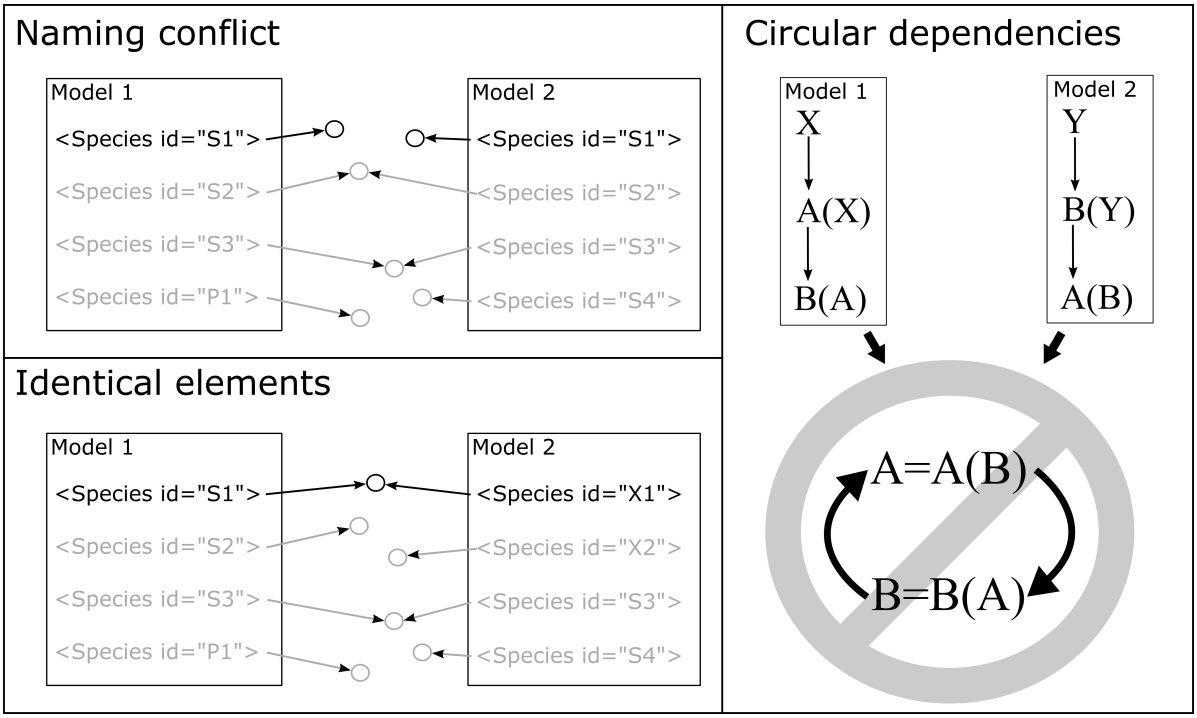

Figure 2: Problems that can arise during merging of biochemical models. Top left: In the two models, two distinct elements carry the same name. Bottom left: Two identical elements carry different names. Right: Loops of algebraic equations. In model 1, variable A is computed from variable B, and in model 2, B depends on A. Combining these two equations would result in a loop that leaves both variables undetermined.

Two separate models can both contain a species with the ID "A", but when these models are merged, one of these species has to be either omitted or renamed in the output file.

In addition, there are *semantical problems* arising from the biologically meaning of elements. For example, the merging algorithm has to identify species that represent identical substances. Since identical substances with the same physical location can appear only once in a model, such multiple references have to be replaced by a single SBML species during the merging process. Also, the program has to detect biologically contradicting elements, such as *overlapping* compartments. For instance, "cell" and "mitochondrion" are overlapping, since the mitochondrion is a part of the cell. Overlapping compartments must be avoided since the corresponding metabolite concentrations would be dependent variables. This is especially undesirable if the variables have been inherited from different models, because then there is no information on their mathematical relation.

Another kind of problems are *loops of algebraic equations*, which resemble circular definitions. For example, if one model states that the amount of ATP depends on the amount of ADP (e.g. $ATP = 2 \cdot ADP$), while the statement is adverse in the second model (e.g. $ADP = 0.5 \cdot ATP$), only one of the equations can be kept, otherwise the resulting model would not be computable.

In any case, SBMLmerge needs to identify the biological meaning of all model elements in order to find identical or contradicting elements. To clarify the relation between model elements and the corresponding biological entities, we have to distinguish between different kinds of attributes in the SBML elements.

Element specific attributes are used to describe the biological entity that the element refers to. They can be divided into two classes:

Identifying attributes define the element's name or how it is called in databases.

For example, for a species tag the ID, the metaID, the name, and the annotation attributes specify the chemical substance described.

Describing attributes specify the biological identity of an element.
An important information about a catalysed reaction is its set of substrates and products. This information is crucial for the biological function of a reaction, but it still cannot be called *identifying*. For example, there might be reactions, catalysed by unmentioned isoenzymes, which have the same sets of reactants and products, but can still be regarded as different, meaning that they can appear together in the same model.

Model specific attributes contain statements that are made by the model. For example, the *constant* attribute of a species states whether the concentration of a species is constant or variable over time.

For model merging, the *identifying* attributes are used to search models for elements that refer to the same biological entity. Two elements which have the same value for any of these *identifying* attributes are called *conflicting*. If conflicting elements also share common values for their describing attributes, they are considered *identical*, but if these values differ, they are considered *contradicting*.

We have already seen that model annotation and model merging can be treated separately one after the other. Altogether, the SBMLmerge program consists of several subroutines: **SBMLannotate** assists the user in specifying the biological meaning of model elements, **SBMLcheck** performs various consistency checks on the syntactical and semantical validity of an SBML file, **SBMLmerge** helps the user in consolidating the mathematical and biological information from several models into a single consistent SBML file, and **SBML2dot** draws a graphical representation of the output model. The subprograms can also be executed individually within other applications. In the following sections, we shall have a closer look at each of them. The functionality of the programs is described in detail in the user manual [16].

3 SBMLannotate

Model merging requires that the biological meaning of the model elements has been declared before. For this reason, a necessary step before model merging is to annotate all SBML elements, by linking them to IDs from databases explaining their biological meaning. SBMLannotate lets the user navigate through an SBML file to annotate the compartments, species, and reactions. For each element, SBMLannotate suggests related database IDs to the user, either based on the SBML code of this element or based on search strings provided by the user. To speed up the search for IDs, a local database is used instead of web-services. SBMLannotate has also an option for automatically adding annotations to all elements that carry unambiguous names or IDs.

All gathered biological information is stored in SBML's *annotation* elements as a string in XML format. A standard for how to store links to databases inside the SBML code has been developed by the MIRIAM [7] initiative. This standard allows to express different kinds of logical relations between model elements and database entries. These relations, which have been taken from Dublin Core (DC) [12], are given in Table 1. Currently only the DC term *relation*, which links the SBML element to a specific database element, is supported by SBMLmerge.
Figure 3 shows an example for a species that has been assigned a database ID in a MIRIAM compliant annotation, using RDF [21] and Dublin Core.

3.1 SBMLannotate ID Database

The SBMLannotate ID database relates *identifying* SBML content to IDs from various databases. Its structure is fairly simple: for the conversion of species into IDs, it contains a list of approximately 40000

Table 1: Dublin Core relations used in the MIRIAM format. Currently only the DC term *relation* is supported by SBMLmerge.

DC term	Use cases for species
relation	The substance in the database is the one described by the SBML species.
isPartOf	The SBML species is a part of the complex species explained in the database.
hasParts	The SBML species is a complex, which consists of parts that are referred to by these database entries.
isVersionOf	The SBML species is one substance out of the set of substances referred to in the database.
hasVersion	The SBML species represents a set of substances which are described by the database entries.
isReferencedBy	The SBML species is referenced by the given resource.

substrate names and their links to the databases Gene Ontology [1], KEGG Compound [5], ChEBI [19], PubChem [13], 3DMET [17] and CAS [18]. This information has mainly been extracted from Gene Ontology and KEGG, which contain cross-linking information to other databases. For the conversion from compartments into IDs, the database contains a list of names for physical compartments (like mitochondrion, cytoplasm, etc.) mapped to IDs from Gene Ontology. For chemical reactions, it contains a list of KEGG Reaction IDs together with the KEGG Compound IDs of all substances that participate in each reaction. This information is used by SBMLannotate to assign annotations to the compartments, species, and reactions from a given SBML model.

3.2 Automatic Proposal of IDs

SBMLannotate contains a subroutine which finds database IDs related to a given SBML element. First, the identifying information is gathered from the SBML file. In the case of compartments or species, potential names are extracted from the identifying attributes of the elements. In the case of reactions, KEGG Compound IDs for the substrates and products are looked up in the SBML code. Afterwards, this information is used to search for potential IDs in the database: In the case of compartments or species, potential names are simply compared with the list of known names. Every gathered name that appears in this list is suggested to the user together with its associated database IDs. For reactions, the database search works differently. Here the user is presented a list of all reactions that share all educts and products with the reaction in the model (while possibly containing other substances).

3.3 Fuzzy Database Search

Besides the automatic search for names inside the SBML code, the user can also type in presumable names for compartments and species. SBMLannotate uses a fuzzy matching algorithm to look up these names in the ID database and to suggest the most similar ones. The algorithm ("Gestalt Pattern Matching") is explained in Figure 4. Despite its simplicity, the algorithm works well in practice since the database contains many alternative substance names. For example, it contains the names "2-Deoxy-D-arabino-hexose" and "D-arabino-2-Deoxyhexose", which both correspond to the KEGG Compound "C00586".

```
<species id="s1" metaid="s1" name="glucose-6-phosphate" compartment="cytosol">
  <annotation>
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns"
             xmlns:dc="http://purl.org/dc/elements/1.1/">
      <rdf:Description rdf:about="#s1">
        <dc:relation>
          <rdf:Bag>
            <rdf:li rdf:resource="http://www.genome.jp/kegg/compound/#C00092" />
          </rdf:Bag>
        </dc:relation>
      </rdf:Description>
    </rdf:RDF>
  </annotation>
</species>
```

Figure 3: SBML species with a MIRIAM compliant annotation. The annotation points to the entry C00092 in the KEGG Compounds database, which represents glucose-6-phosphate. To create such an annotation, SBMLannotate suggests a list of database IDs, based on information extracted from the SBML code. For this species, SBMLannotate would have suggested the ID "C00092" from the KEGG Compound database, as the name "glucose-6-phosphate" is listed together with "C00092" in the SBMLmerge ID database. After the user has selected this ID, the annotation is inserted automatically into the SBML element.

4 SBMLcheck

SBMLcheck screens SBML models for semantic problems and reports them to the user. Its purpose is to ensure that only correct models are imported into SBMLmerge. Combining correct models with SBMLmerge will either lead to a correct output model, or if the merging process fails, to no output at all. Hence to obtain a correct output model, it is crucial to ensure the validity of the input models.

In general, the definition of validity depends on the purpose of a model. A model formulated in terms of chemical reactions will have other requirements for being valid than a model defined by differential equations. SBMLcheck checks the model for requirements that are universal for different ways of modelling and that are relevant for model combination. Models can be invalid for syntactic or semantic reasons: tools to check SBML files for syntactical problems [15] are already available, so SBMLcheck focuses on detecting semantical problems.

It performs the following checks:

1. *Syntax check*
   When a file is processed, it is first checked for correct SBML syntax.

2. *Correct annotations*
   Like SBMLmerge, SBMLcheck requires correctly annotated models. SBMLcheck searches for elements that are not annotated and displays them to the user. Correct annotations are necessary for the following checks.

3. *Mathematical rules*
   According to the specification of SBML, all variables that appear in an assignment rule must have been specified by a preceding rule. This excludes loops between algebraic rules. SBMLcheck verifies that the rules can be evaluated consecutively in the order they are given.

4. *Overlapping annotations*
   SBMLcheck checks all elements for their annotations and warns the user if two elements carry

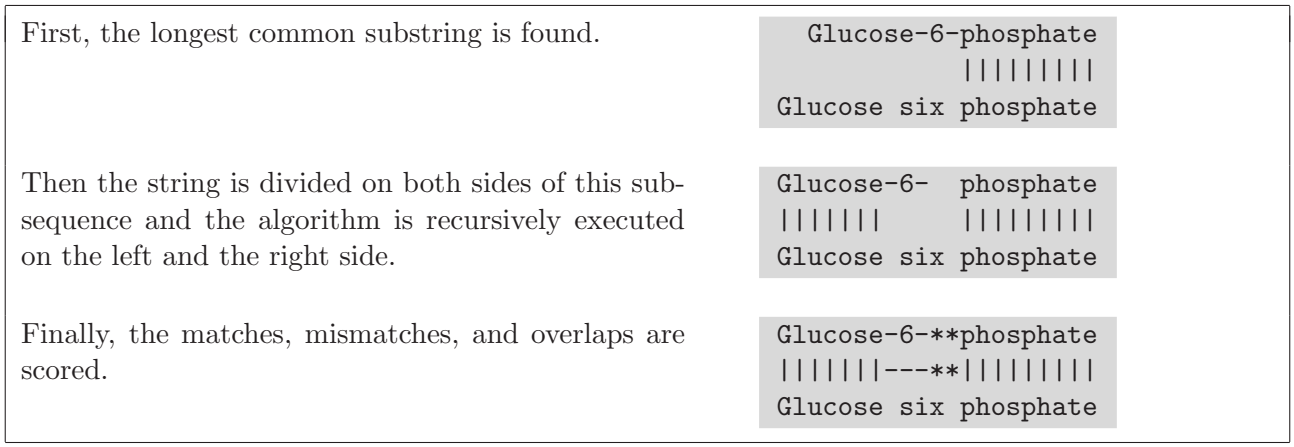

Figure 4: Gestalt pattern matching. The algorithm developed by Ratcliff and Obershelp [9] compares two different strings by recursively finding their longest common subsequence. It is implemented in Python [14] in the module *difflib*. In this example, the algorithm is used to compare the strings "Glucose-6-phosphate" and "Glucose six phosphate".

the same annotation, which would indicate overlapping elements. To check for overlapping compartments, SBMLcheck uses the Gene Ontology, which represents spatial relations between cell compartments as a directed acyclic graph. For example, this graph indicates that the mitochondrial matrix is situated inside the mitochondrial lumen, which again is part of the mitochondrion. To use Gene Ontology, it is necessary that all compartments have been annotated with Gene Ontology identifiers.

5. *Balance of atom numbers*
   SBMLcheck investigates the balance of atom numbers in reactions. In a correct reaction, the number of atoms of each chemical element has to be the same on both sides of the reaction formula. A difference indicates that (i) either some metabolites (e.g. water) are missing or (ii) an error has occurred during the modelling of the reaction. SBMLcheck counts the number of atoms of each type on both sides and raises a warning if they do not match. This kind of inconsistency does not preclude models from being merged. The atomic composition of the metabolites is looked up in the KEGG database.

5 SBMLmerge

SBMLmerge first merges the lists of elements in the annotated input files. The resulting list is then searched for *conflicting* elements by pairwise comparison, based on the *identifying* attributes, including the annotations. If two conflicting elements are found, their *describing* attributes are compared. The values for these attributes can be identical for both conflicting elements, or they can differ in one or more values: if all attribute values are identical, the elements are assumed to have the same biological meaning. In order to keep the models semantically valid, SBMLmerge has to delete one of these elements from the file. On the other hand, if the elements differ in at least one attribute, they are not considered identical, so both of them are retained in the output model.

To be more specific, we shall describe two particular conflicts that can arise during the merging process:

1. *Naming conflicts between species elements*
   As mentioned before, the *identifying* attributes for species are metaID, ID, name, and annotation. If two species have been annotated with identical database links, their *describing* attributes are

compared. For species, the compartment attribute is *describing*, since a species cannot be defined twice in the same compartment, but in different ones. So if two species with identical annotations are found in the same compartment, they are regarded as *identical* and replaced by a single species.

2. *Conflicts between assignment rules*
   Multiple assignments of the same variable are either redundant or contradictory and are forbidden in the SBML language. In order to find multiple assignments during merging, the *left hand side* of assignment rules (the variable, which is changed by the rule) is regarded as the *identifying* attribute. Accordingly, two rules that modify identical variables are regarded as *conflicting*. Since no further *describing* attributes of rules are taken into account, *conflicting* assignment rules are always regarded as *contradicting*, and so the user has to choose one of them to be deleted from the output model.

As mentioned before, dependence circles between rules are also forbidden (e.g. A appears in rule for B, B appears in rule for C, and C appears in rule for A). When lists of rules from the input models are simply joined, such circles can easily arise. SBMLmerge detects them and suggests ways to remove them from the output model. Whenever a circle is found, the program proposes how it can be removed by omitting or modifying a rule.

A detailed list of further problems handled by SBMLmerge can be found in the user manual [16].

6 SBML2dot

After models have been merged, SBML2dot plots a network graph of the output model (see Figure 1). For the actual drawing, SBML2dot calls GraphViz [4], which can draw pictures in various formats from a given input file in dot format. The resulting network graph contains the species and reactions of a model and their relations to each other. The user can choose between different colouring schemes. For example, colours can indicate the models from which species and reactions were inherited. The program can also retrieve species names from the SBMLannotate database, based on the information given in the annotation tag.

7 Discussion

Summary. SBMLmerge helps the user to combine several biochemical models given in SBML format. To handle the semantics of single SBML models, we have developed SBMLannotate for attachment of unique identifiers to model elements, and SBMLcheck for verification of the semantic consistency of SBML files. First, the user assigns biological annotations to elements. Based on these annotations, SBMLmerge can decide whether elements describe the same or related biological quantities. The output models are valid SBML files that perform reasonably in simulations, as shown in Figure 1.

The program is distributed under the GNU public license [20] and can be downloaded or accessed via a web interface at `http://sysbio.molgen.mpg.de/sbmlmerge`. The algorithm underlying SBMLmerge can also be used as a guideline to combine models (not necessarily in SBML format) by hand.

Qualifying relations. At present, SBMLmerge supports only the Dublin Core term *relation* for annotation of elements. This "is_a" relationship suffices to annotate simple molecules and chemical reactions, but to handle some of the problems mentioned, we plan to support other relations from Table 1 as well. For instance, a qualifying relation "has_parts" could be used for annotating phosphorylated molecules or protein complexes that consist of several subunits. Another problem are imprecisely specified elements, like "glucose" instead of "α-D-glucose", and lumped metabolites or reactions. Lumped metabolites refer to a pool of metabolite concentrations: in the glycolysis model used

in [10], for instance, the species "triosephosphates" comprises the two distinct substances dihydroxyacetone phosphate and glycerine 3-phosphate. Combining models with different levels of accuracy will cause problems because a lumped metabolite may overlap with a precisely defined metabolite from the other model. Conflicts between models containing lumped metabolites or reactions could be detected based on qualifying relations in the form "has_version".

Biological preconditions. Meaningful combined models can only result from models describing the same type of cell under comparable experimental conditions. Bearing this in mind, it is vital for bottom-up modelling to define standardised experimental conditions in order to ensure interoperability between the models. We are aware that this is not possible for all experiments, but we hold that standards for experiments and models will be crucial for systems biology anyhow. In the end, it will always be up to the user to decide whether two models can be merged at all. Here we showed how at least some of the important technical problems can be detected and solved automatically.

Acknowledgments

Marvin Schulz, Jannis Uhlendorf and Edda Klipp are supported by the German Federal Ministry for Education and Research (BMBF,grant 031U109C); they are members of the Berlin Center for Genome based Bioinformatics, BCB. The work of Wolfram Liebermeister is funded by the European Commission, grant No. 503269.

References

[1] Ashburner, M. and Ball, C.A. *et al.*, Gene Ontology: tool for the unification of biology, *Nat. Genet.*, 25:25–29, 2000.

[2] Finney, A., and Hucka, M., Systems biology markup language: Level 2 and beyond, *Biochem. Soc. Trans.*, 31:1472–1473, 2003.

[3] Funahashi, A., and Kitano, H., CellDesigner: a process diagram editor for gene-regulatory and biochemical networks, *BioSilico*, 1:159–162, 2003.

[4] Gansner, E.R., and North, S.C., An open graph visualization system and its applications to software engineering, *Softw. Pract. Exper.*, 30(11):1203–1233, 2000.

[5] Kanehisa, M. and Goto, S., KEGG: Kyoto Encyclopedia of Genes and Genomes, *Nucleic Acids Res.*, 28:27–30, 2000.

[6] Le Novère, N., Bornstein, B., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J.L., and Hucka, M., BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems, *Nucleic Acids Res.*, 34:689–691, 2006.

[7] Le Novère, N., Finney, A., Hucka, M., Bhalla U., Campagne F., Collado-Vides J., Crampin E., Halstead M., Klipp E., Mendes P., Nielsen P., Sauro H., Shapiro B., Snoep J.L., Spence H.D., and Wanner B.L., Minimum information requested in the annotation of biochemical models (MIRIAM), *Nat. Biotechnol.*, 23:1509–1515, 2005.

[8] Lloyd, C.M., Halstead, M.D., and Nielsen, P.F., CellML: its future, present and past, *Prog. Biophys. Mol. Bio.*, 85:433–450, 2004.

[9] Ratcliff, J.W., and Metzener, D., Pattern matching: The gestalt approach. *Dr. Dobb's Journal*, 1988.

[10] Snoep, J.L., Bruggemann F., Olivier B.G., and Westerhoff, H.V., Towards building the silicon cell: A modular approach, *Biosystems*, 83:207–216, 2006.

[11] Snoep, J.L., and Olivier, B.G., JWS online cellular systems modelling and microbiology, *Microbiology*, 149:3045–3047, 2003.

[12] `http://dublincore.org/`

[13] `http://pubchem.ncbi.nlm.nih.gov/`

[14] `http://python.org/`

[15] `http://sbml.org/validator/`

[16] `http://sysbio.molgen.mpg.de/sbmlmerge/sbmlmerge$\_$manual/`

[17] `http://www.3dmet.dna.affrc.go.jp/`

[18] `http://www.cas.org/`

[19] `http://www.ebi.ac.uk/chebi/`

[20] `http://www.gnu.org/copyleft/gpl.html`

[21] `http://www.w3.org/RDF/`

[22] `http://www.w3.org/RT/MathML/`

[23] `http://www.w3.org/XML/`