Towards a Complete Covering of SBML Functionalities

Tommaso Mazza

'Magna Græcia' University of Catanzaro, Italy
t.mazza@unicz.it

Summary. The complexity of biological systems is at times made worse by the diversity of ways in which they are described: the organic evolution of the science over many years has led to a myriad of conventions. This confusion is reflected by the in-silico representation of biological models, where many different computational paradigms and formalisms are used in a variety of software tools.

The Systems Biology Markup Language (SBML) is an attempt to overcome this issue and aims to simplify the exchange of information by imposing a standardized way of representing models. The success of the idea is attested to by the fact that more than 110 software tools currently support SBML in one form or another.

This work focuses on the translation of the Cyto-Sim simulation language (based on a discrete stochastic implementation of P systems) to SBML. We consider the issues both from the point of view of the employed software architecture and from that of the mapping between the features of the Cyto-Sim language and those of SBML.

1 Introduction

Nowadays, very few common exchange formats exist. We face difficulties to exchange models among different analysis and simulation tools. Therefore, taking advantage of the different tools power and capabilities is the main issue among scientists.

To overcome this issue, in March 2001, a first step was taken. During the *First International Symposium on Computational Cell Biology*, (Massachussetts, USA), Michael Hucka presented a new simple, well-supported and with textual substrate (XML) language adding components that reflect the natural conceptual constructs used by modellers in the domain, **SBML: Systems Biology Markup Language**.

In the following November 2002, M. Hucka talked again about *The Systems Biology Markup Language* at the *I3C 4th Quarter Technical Meeting* (San Diego, CA). On that occasion, starting from the observation of the enormous proliferation of software tools in this domain, he observed that a single package able to

cover all needs does not exist. Different packages can have different *niche strengths* and their strengths are often complementary. Moreover, he highlighted that much likely, no single tool is able perform likewise in the near future because the range of capabilities needed is large and new techniques and new tools evolve all the time making simulations and results often not shareable or reusable. Finally, he remarked that SBML is intended to be a common exchange format for transferring network models among tools, even if it may not capture everything represented by every tool (lossy transformation). At the same time, SBML is not suited to represent experimental data and numerical results, even if with the addition of the metadata support it may be suitable as a storage format for models too.

In May 2003, during the *I3C May 2003 Meeting* (Cambridge, MA) with a presentation entitled *Update on the Status of the Systems Biology Markup Language (SBML)* [17], M. Hucka dealt with the *Related Efforts*, the *current status* of the features and the *software libraries* for SBML. In particular he talked about similarities with *CellML* and the current joint work with the aim to bring them together. He also discussed the adoption in SBML of some features from CellML, like the *MathML Subset* and the *Metadata specification*. Moreover, he presented *BioPAX*, specifying that it is oriented towards being a common exchange format for databases of pathways Complementary efforts and then not a competing tool. However, SBML and BioPAX teams will work together to define linkages between SBML and BioPAX representations. Finally, he presented the version 1.0.1 of *libsbml*, a library designed to help modellers to read, write, manipulate, translate, and validate SBML files and data streams. In the same year many other presentations have been given by M. Hucka and others. In fact, in June 2003 MIPNETS Meeting (Liverpool, UK), Finney presented *The Systems Biology Workbench and Systems Biology Markup Language* [9], a project funded by *Japan Science and Technology Corporation ERATO program* and started in the summer 2000. The project goal was to provide software infrastructure which (i) enables sharing of simulation/analysis software and models and (ii) enables collaboration between software developers. Focused on biochemical modelling, it is an environment that enables tools to interact using SBML to transfer models between tools and supporting resource sharing.

All 2004 long was spent to delineate the limitations of the current SBML specification. In the same year Shapiro at al. talked about *MathSBML* [39], a *Mathematica* package designed for manipulating SBML models. It converts SBML models into Mathematica data structures and provides a platform for manipulating and evaluating these models. In [16], Hucka et al. summarise the current and upcoming versions of SBML and their efforts at developing software infrastructure for supporting and broadening its use. They also provided a brief overview of the many available SBML-compatible software tools.

In the following year other interesting publications appeared. The one about *MIRIAM* [26] is related to the way to define a minimum quality standard for the encoding of biochemical models by means of a set of rules about quantitative models of biological systems. These rules define procedures for encoding and an-

notating models represented in machine-readable form. In May 2005 as well, an ideal test bed in the understanding of the cellular systems by means of computational modelling appeared in [42]. In this chapter one goes over (i) computing for modelling cells, (ii) SBML and (iii) developing the International E. coli Alliance, which has been created to tackle the whole cell problem. Again in [15] and [8], one presents SBML as an XML-based exchange format for computational models of biochemical networks, including overview, enhancements, several proposals for the language extension, model composition and multi-component chemical species. In [32], the first step forward is taken by the *P System* community for representing P systems which model biological reaction networks as SBML models.

In 2006 a very good paper about the *BioModels database* was published, *an annotated resource of quantitative models of biomedical interest. Models are carefully curated to verify their correspondence to their source articles. They are also extensively annotated, with (i) terms from controlled vocabularies, such as disease codes and Gene Ontology terms and (ii) links to other data resources, such as sequence or pathway databases. Researchers in the biomedical and life science communities can then search and retrieve models related to a particular disease, biological process or molecular complex* [25]. In February of the same year, the joint work between CellML and SBML team groups had a big result, namely the *CellML2SBML* tool [38] implemented as a suite of XSLT stylesheets that, when applied consecutively, convert models expressed in CellML into SBML without significant loss of information. One month later, Sarah Keating et al. presented another similar conversion tool: *SBMLToolbox* [22] a toolbox that facilitates importing and exporting models represented in SBML in and out of the MATLAB environment and provides functionality that enables an experienced user of either SBML or MATLAB to combine the computing power of MATLAB with the portability and exchangeability of an SBML model. Other two useful tools appeared in October of the same year, they were *SBMLSupportLayout* and *SBWAutoLayout*, supporting reading, creating, manipulating and writing layout information for biochemical models. SBMLSupportLayout can read, update, add and render model layout information. SBWAutoLayout can automatically layout models, graphically manipulate model layouts and generate layout information for models without layout information. Using them, researchers can study large or complex biochemical networks and benefit from the ability to automatically create lucid visualizations and store them in a portable and widely accepted format [5]. In the winter of the same year, Bergmann and Sauro described the current state of the *Systems Biology Workbench* talking about how users and developers can perceive SBW and then focusing on currently available SBW modules [2]. Yet other four smart tools have been presented in the current year: *SBML ODE Solver Library* [28] (SOSlib), a programming library for symbolic and numerical analysis of chemical reaction network models encoded in SBML; *SBML-PET* [43], a tool designed to enable parameter estimation for biological models including signalling pathways, gene regulation networks and metabolic pathways. It can estimate the parameters by fitting a variety of experimental data from different experimental conditions.

In the same year, Eccher and Priami presented a tool to translate SBML into pi-calculus [6] while Gheorghe presented the *P System Modelling Framework* [10], a framework able to simulate the evolution of multi-compartmental Gillespie algorithm over a hierarchy of compartment structures.

In 2007, other tools have appeared. The first one has been *SBMLR*[1], a tool able to link R to libsbml for SBML parsing and output converting SBML to R graph objects, and more. Another emerging project is *SemanticSBML*[2]: a suite of tools to facilitate merging of SBML models for systems biology starting from all elements in the SBML files described by MIRIAM-type annotations. SemanticSBML will help to insert and check such annotations. The need to build a tool to facilitate the quick creation and editing of models encoded in SBML has been growing with the number of users and the increased complexity of the language. *SBMLeditor* [36] tries to answer this need by providing a very simple, low level editor of SBML files. Users can create and remove all the necessary bits and pieces of SBML in a controlled way, that maintains the validity of the final SBML file.

As many tools have been implemented all around SBML just to highlight the trust of developers on the standardizing initiatives related to the software biological infrastructures towards commons exchange formats. In particular, it is an undeniable fact the increasing and unison consensus among developers in favour of SBML. In fact, several languages have been recently developed to overcome these kind of problems (integrations, standardizing, reuse of biological models) [27], [14], [7], [40], [41], [12], [29], [1], [19], [18]. However, only two XML-based formats are suitable for representing compartmental reaction network models with sufficient mathematical depth that the descriptions can be used as direct input to simulation software. The two are CellML [4], [13] and SBML[17]. The latter is becoming a defacto standard for a common representation supporting basic biochemical models. In fact, today, SBML is supported by over 110 software systems. As a consequence, many SBML models of gene regulatory networks and metabolic pathways that code a considerably body of biological knowledge have been accumulated in repositories. Among all databases, I recall (i) the *PANTHER* Classification System, [31], an unique resource that classifies genes by their functions, using published scientific experimental evidence and evolutionary relationships to predict function even in the absence of direct experimental evidence; (ii) KEGG [21], a knowledge base for systematic analysis of gene functions, linking genomic information with higher order functional information; (iii) JWS Online [34], a Systems Biology tool for simulation of kinetic models from a curated model database and (iv) Reactome [20], a curated resource of core pathways and reactions in human biology. The information in this database is cross-referenced with the sequence databases at NCBI, Ensembl and UniProt, the UCSC Genome Browser, HapMap, KEGG (Gene and Compound), ChEBI, PubMed and GO. In addition to curated human events, inferred orthologous events in 22 non-human species including mouse, rat, chicken, zebra fish, worm, fly, yeast, two plants and E.coli are also available.

[1] Web Site of SBMLR: http://cran.r-project.org/src/contrib/Descriptions/rsbml.html
[2] Web Site of SemanticSBML: http://sysbio.molgen.mpg.de/semanticsbml/

Therefore, with the constant focus on SBML, in this paper I am going to inspect in section 2 all the features and the internal structure of SBML, in the section 4 the software facilities employed and the software packages implemented to build a pure Java library to handle SBML documents, in the section 3 I am going to show how to use the library to encode and decode information from a SBML file to Cyto-Sim model and vice-versa. In the section 5 I am going to test the software package implemented on real SBML files taken from different data sources and in the last section I am going to delineate the future works.

2 SBML Structure

In this paper I am going to take into consideration the latest stable release of SBML highlighting differences with the previous versions. SBML is primarily oriented to allow models to be encoded using XML. Major release of SBML are termed *levels* and represent substantial changes to the composition and structure of the language. The latest release[3] is the SBML level 2. It represents an incremental evolution of the level 1, therefore a valid SBML level 1 can be mapped in a valid SBML level 2 file while only a subset of level 2 can be mapped in a level 1 file. However, both levels remain separated and distinct. Instead, minor revisions of whatever level of a SBML file will bring a new *version*. A new version carries new corrections, adjusts and refinings of the language. Some languages features of previous levels or versions can be deprecated or entirely removed in the time. A feature can be directly removed, but not recommended, although the usual path to completely remove a feature is to deprecate it before and then remove to maintain backward compatibility as much as possible.

SBML allows models of arbitrary complexity to be represented. Each type in a model is described using a specific type of data structure that organizes the relevant information and derive directly or indirectly from a single abstract type called *Sbase*. In addition to serving as the parent class for most other classes of objects in SBML, this base type is designed to allow modeller or a software package to attach arbitrary information to each major structure or list in an SBML model. Sbase contains two default elements: (i) *notes*, intended to serve as a place for storing optional information intended to be seen by humans; (ii) *annotations*, it is a container for optional software-generated content not meant to be shown to humans. All other types have at least other three parameters: (i) *id*, a mandatory field on most SBML structures used to identify a component within the model definition; (ii) *name*, an optional field, not intended to be used for cross-referencing purposes within a model but just to provide a human-readable label for the component; (iii) *sboTerm*, an optional field used to identify a term from an ontology where vocabulary describes entities and processes in computational models.

[3] at the writing time

2.1 SBML in more depth ...

The top level of an SBML model definition consists of a list of all data type (figure 1), all being optional. These elements are contained in the *model* element directly enveloped in the most outer element properly called *sbml*.

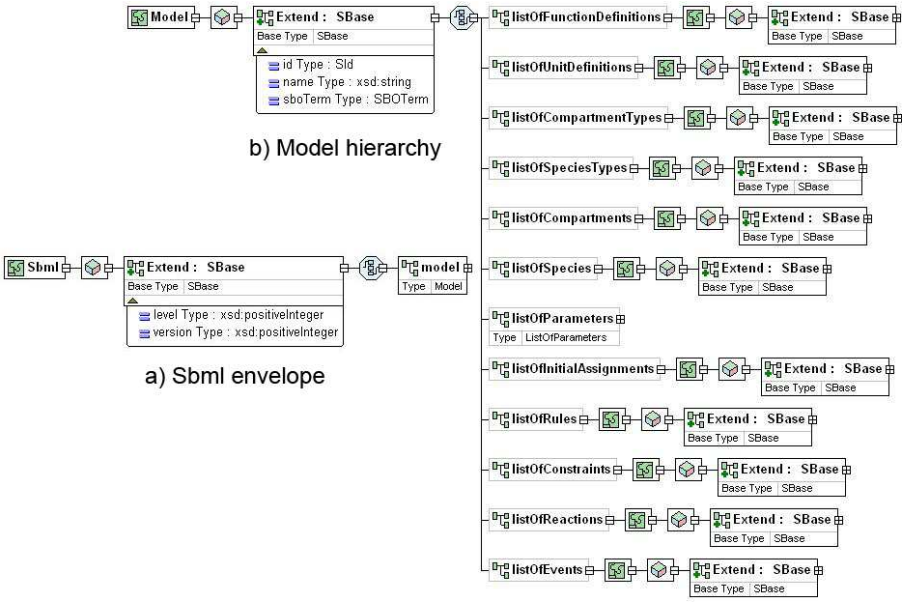

Fig. 1. List of all data types in a SBML model

SBML envelope: The root of a SBML file is an *SBML element* with three required fields: two attributes (*level* and *version*) and exactly one child node (*model*).

Model: It is the highest level construct in a SBML document. It has got three optional parameters: *id*, *name*, *sboTerm* and the hierarchy of data types just before shown and in the exact order of the listed terms.

Function definitions: A *function* (or often called *user-defined function*) is a mere textual macro containing a MathML lambda element, then its call can be implemented as a textual substitutions. It has limited capabilities because none external parameter can be referenced by a function.

Unit definitions: SBML optionally allows for the employment of units of measurements for some mathematical entities (e.g. compartment size, reaction rates, mathematical formulas, etc...). By means of two operative classes (*UnitDefinition* and *Unit*), a new unit of measurement (*UnitDefinition(newid, newname)*) can be obtained by composition of the elementary ones already defined in SBML (*Unit(id, name)*).

Compartment types: If compartments share either a common biological function or similar reactions or other underlying conceptual features, they can be grouped assigning to them the same compartment type value. After defining a compartment type with its univocal identifier, it can be referred by each compartment belonging to the category specified by that compartment type.

Species types: As compartments, if species share common characteristics, they can be grouped in a same logical set specified by a new species type. The existence of SpeciesType structures in a model has not effect on the model's numerical interpretation.

Compartments: A compartment represents a bounded space in which species are located. Compartments can be arranged in hierarchical structures and can have a static or dynamical size.

Species: A species refers to a multiset of entities of a specific species type that take part in reactions and are located in a specific compartment. With a species it is possible to specify a number of parameters to better explaining its characteristic. For example for a species one can specify the substance quantity or the concentrations, the charge, the substance unit, etc...

Parameters: As usual programming languages, they are variables that can be used in mathematical formulas. They are defined as constant values and can be global, if defined at the beginning of the model or local if defined within the scope of a reaction. Local parameters overwrite the global ones with the same name.

Initial Assignments: An initial assignment is an alternative way to set quantities of species, size of compartments and value of parameters with complex mathematical expressions at the beginning of a simulation. It overwrites eventual already existing numerical values for quantities, sizes and parameter with the mathematical expression it carries.

Rules: A rule furnishes a supplementary way to set a variable of the system that cannot be set otherwise by any reaction rules or initial assignments. There are three kind of rules: (i) *assignment*, used to express equations that set the values of the variables. As an initial assignment, it overrides eventual existing numerical values of species quantity, compartments size and parameters value with its mathematical expression. It is forbidden to have both an initial assignment and a assignment rule related to the same object; (ii) *rate*, it is used to express equations that determine the rates of change of variables. It can refer to species, compartments and parameters as before explained. In the context of a simulation, it can have effect either at $t = 0$ to obtain consistent initial conditions or at $t > 0$ during the simulation run. To avoid indetermination problems, for each object in a system, only one between assignment rules and rate rules can exist; (iii) *algebraic rule*, used to express equations that are neither assignments of model variables nor rates of changes. The only one role is to distinguish this case from the other cases.

Constraints: Constraints are mathematical expressions used to check permissible values of different quantities in a model. A constraint should be checked against

a variable at all time and triggers a message warning if the conditions expressed by the formula is not verified.

Reactions: Reactions are processes that result in the interconversion of substances that can change their quantities. The participants of a reaction are called *reactants* and *products*. They change their quantities according to their own stoichiometric coefficient and the kinetic law which rules the reaction. All participants must belong to the list of species. A reaction can be optionally set as *reversible*, *fast* and can contains modifiers, namely species acting as catalysts or inhibitors.

Events: Events defines changes of variables of a system when a triggering condition fires. In particular an event specify (i) under which mathematical condition, (ii) how and (iii) when a variable changes. The triggered events can be delayed and applied to more than one variables at $t > 0$. As usual, events have effect on species quantity, compartments size and parameters value.

2.2 Differences between SBML Level 1 and Level 2

Many significant changes characterize the new level 2. They cover the (i) identification of the objects, (ii) annotation within the model, (iii) language for kinetic expressions, (iv) dimensionality of compartments and (v) rule specification.

In particular SBML Level 2 supports the inclusion of metadata. In fact, all structures in SBML can be annotated with optional content in RDF. The top-level *Model* structure can contain an optional list of global user-defined functions expressed in *MathML* and organized in new structures of type *FunctionDefinition* and can contain an optional list of event definitions organized in structures of type *Event*. All data structures, including SBML and listOf elements, are now derived from the type SBase. This means all major structures in SBML can have separate annotations and metadata associated with them.

A new field, *id*, replaces the name field previously defined for most SBML structures to identify each part of a model. Formulas in Level 2 are expressed using MathML 2.0. The field named formula previously available on the KineticLaw and Rule structures has been replaced by a MathML element named math containing MathML content. In addition, stoichiometry numbers may now be expressed using MathML, allowing for more flexibility in defining reactions. Unlike in SBML Level 1, unit identifiers in Level 2 are in a separate namespace from the namespace used for models, functions, species, compartments, reactions and parameters and have the additional fields *multiplier* and *offset* to enable the definition of non-SI units.

The Compartment structure has a new field, *spatialDimensions*, whose value is a positive integer specifying the number of dimensions in space the compartment possesses. This enables the definition of such things as two-dimensional membranes. All fields representing initial conditions or parameter values, including compartment sizes and species concentrations, are optional in Level 2. A missing value for one of these fields implies that the value is either unknown, not required for analysis, or should be obtained from an external source. The Compartment,

Species and Parameter structures each have a new boolean field named constant. This field specifies whether the variables represented by these structures can be changed by rules and reactions. In particular, the Species structure has a new field, *initialConcentration*, for setting the initial value of a species in terms of its concentration. This is in addition to the ability, carried over from Level 1, to set the values in terms of amounts.

There is no longer a type field on Rule, and new structures *AssignmentRule* and *RateRule* replace SBML Level 1's *ParameterRule*, *SpeciesConcentrationRule* and *CompartmentVolumeRule*. The Reaction structure has a new list of modifiers in addition to the list of reactants and products. The *listOfModifiers* enumerates species that affect a reaction but are neither created nor destroyed by the reaction.

3 SBML ⇔ Cyto-Sim

As deeply shown, SBML is a powerful and well defined language for modelling biological interactive systems in standard way. The aim of my work has been to make Cyto-Sim able to *speak* and *understand* SBML.

Cyto-Sim [3] is a stochastic simulator of biochemical processes in hierarchical compartments which may be isolated or may communicate via peripheral and integral membrane proteins. It is available online as a Java applet [33] and as standalone application. For security issue, although the functionalities of the applet has been reduced, it fully and correctly works. By means of it, it is possible to model: (i) interacting species; (ii) compartmental hierarchies; (iii) species localizations inside compartments and membranes and (iv) rules and their and correlated velocity formulas which govern the dynamics of the system to be simulated, as chemical equations.

Some real biological systems have already been successfully simulated in the past by means of Cyto-Sim. Now I am going to try to explain at first how to translate a Cyto-Sim model into SBML (and vice-versa) and later I will test the quality of the translation comparing the simulations available in literature against those obtained by Cyto-Sim about the same models.

3.1 Speaking SBML

The conversion process from the Cyto-Sim syntax to the SBML one is quite straightforward. In Cyto-Sim, users must declare the species present into the system writing something like this:

```
/* Object Declaration */
object speciesA, speciesB, speciesC
```

This line of code corresponds to the following SBML chunk of code:

```
<listOfSpecies>
   <species id="compartmentA_0_speciesA" name="speciesA"
```

```
        compartment="compartmentA" initialAmount="0.0"/>
    <species id="compartmentB_0_speciesB" name="speciesB"
        compartment="compartmentB" initialAmount="1.0"/>
    <species id="compartmentC_2_speciesC" name="speciesC"
        compartment="compartmentB" initialAmount="2.0"/>
</listOfSpecies>
```

Not all the information in this XML code can be retrieved by the previous objects specification[4]. In fact, the compartment, membrane and initial amount related to one species are reached both from the following code:

```
/* Compartments Declarations */
compartment compartmentA [ruleA]
compartment compartmentB [compartmentA, ruleB, ruleC,
   speciesB, 100 speciesB@7000 : |2 speciesC|]
system compartmentB
```

and from this:

```
/* Rules Declarations */
rule ruleA {
   speciesA k1-> *
   || + speciesA k2-> speciesA + ||
}
rule ruleB speciesB k3-> speciesC
rule ruleC |speciesC| k4-> || + speciesC
```

From the code related to the compartments we take information about (i) the compartment hierarchy, (ii) which rule happens and in what compartment, (iii) the declared initial quantities (species not declared in this context will not still exist as default at the beginning of the simulation) and (iv) eventual re-feeding events at specified evolution times. Considering now that a reaction happening in a compartment acts only on the species within it, looking the localization of a reaction we can infer the localization of its reactant species. Moreover it is possible to notice that the rule *ruleC* acts on the species *speciesC* inside the membrane[5] (membrane number 2) of the compartment *compartmentB*.

In SBML each compartment is quadruplicated to easily handle membranes.

```
<listOfCompartments>
    <compartment id="compartmentA_0" compartmentType="compartmentA"
        outside="compartmentA_1"/>
```

[4] The figure between the compartment and the species names within the string assigned to each species id corresponds to the membrane in which a species sits. For more information about the syntax, look at [3]

[5] Recall that in this context a compartment is surrounded by a membrane with a not negligible thickness, therefore a compartment is logically divided into the internal (membrane 0), internal and superficial (membrane 1), intra (membrane 2), external and superficial (membrane 3) and external (membrane 4) membranes.

```
    <compartment id="compartmentA_1" compartmentType="compartmentA"
        outside="compartmentA_2"/>
    <compartment id="compartmentA_2" compartmentType="compartmentA"
        outside="compartmentA_3"/>
    <compartment id="compartmentA_3" compartmentType="compartmentA"
        outside="compartmentB_0"/>
    <compartment id="compartmentB_0" compartmentType="compartmentB"
        outside="compartmentB_1"/>
    <compartment id="compartmentB_1" compartmentType="compartmentB"
        outside="compartmentB_2"/>
    <compartment id="compartmentB_2" compartmentType="compartmentB"
        outside="compartmentB_3"/>
    <compartment id="compartmentB_3" compartmentType="compartmentB"
        outside="system_0"/>
    <compartment id="system_0" compartmentType="system"
        outside="system_1"/>
    <compartment id="system_1" compartmentType="system"
        outside="system_2"/>
    <compartment id="system_2" compartmentType="system"
        outside="system_3"/>
    <compartment id="system_3" compartmentType="system"/>
</listOfCompartments>
```

Then a single compartment generates four independent concentric compartments, as a matrioska doll toy, related to the same compartment but enclosing different spatial areas and then species. To keep conceptually linked these compartments, a compartment type specification is provided.

```
<listOfCompartmentTypes>
    <compartmentType id="compartmentA"/>
    <compartmentType id="compartmentB"/>
    <compartmentType id="system"/>
</listOfCompartmentTypes>
```

The previously seen reactions are easily translated into SBML differenting the names of the grouped rules (e.g. the ruleA group contains two reactions. Their names will become: ruleA.0 and ruleA.1). Moreover, the kinetic formulas just touched (k1, k2, etc) before are expressed by MathML expressions inside <kineticLaw> tags.

```
<listOfReactions>
    [...]
    <reaction id="ruleA.1" name="compartmentA_0_ruleA.1">
        <listOfReactants>
             <speciesReference species="compartmentA_0_speciesA"
                stoichiometry="1.0"/>
        </listOfReactants>
        <!--listOfProducts>No Products</listOfProducts-->
        <kineticLaw>
```

```
            <math xmlns="http://www.w3.org/1998/Math/MathML">
               <apply>
                  <times/>
                  <cn>k1_value</cn>
                  <ci>compartmentA_0_speciesA</ci>
               </apply>
            </math>
         </kineticLaw>
      </reaction>
   </listOfReactions>
```

Cyto-Sim also requires the specification of a range of evolution times and of the species whose quantities have to be plotted on the screen.

```
evolve 0 - 1000
plot compartmentA[speciesA], compartmentB[speciesB:|speciesC|]
```

This information can be encoded in SBML by the use of an annotation which is auto-explicative.

```
<annotation>
    <Cyto-Sim xmlns:cytosim="http://www.sbml.org/2001/ns/cytosim">
        <plot>
            <species>compartmentA_0_speciesA</species>
            <species>compartmentB_0_speciesB</species>
            <species>compartmentB_2_speciesC</species>
        </plot>
        <evolve>
            <from>0</from>
            <to>1000</to>
        </evolve>
    </Cyto-Sim>
</annotation>
```

3.2 Understanding SBML

The process to make an existing SBML file comprehensible to Cyto-Sim is more complex than the opposite step. Keeping in mind the correspondences among structures before shown, during this kind of translation we have to check some restrictions and to guarantee some constraints which are now explained.

Parameters: SBML optionally carries global parameters, visible everywhere in the file and local ones with more restricted scope. During the parsing time of an SBML file, Cyto-Sim loads all global parameters putting them into a global HashMap. In the case of local parameters inside kineticLaw of reactions, Cyto-Sim considers local and global parameters together taking care to overwrite eventual global parameters with the same name of local ones.

Species Quantities: SBML provides optional size for compartments. Cyto-Sim handles quantities and not concentration for species, then each concentration (if any) has to be converted into quantity. To do that, Cyto-Sim requires the size specification for each compartment if there are any specification of the species concentrations inside it.

Assignments: Cyto-Sim handles assignment rules at the moment of parsing and use them to replace eventual existing fixed values specified for species quantity, compartment size or parameters value. Up to now, it does not understand initial assignments, rate rules and algebraic rules. These features will be made available soon.

Functions: Cyto-Sim does not still handle λ-functions.

Units & Constraints: Cyto-Sim does not still make use of units of measurements and costraints.

4 Binding to the SBML Schema

After having conceptually explained how Cyto-Sim converts SBML in its own language and vice-versa, now I am going to show which software architecture gives it the possibility to do that. I used two well known tools for this aim: the Java Architecture for XML Binding (JAXB) package and the XML DOM parser, both build-in the latest release of Java (Java Mustang).

JAXB [11] simplifies access to an XML document from a Java program by presenting the XML document to the program in a Java format. The first step in this process is to bind the schema for the XML document into a set of Java classes that represents the schema. Binding a schema means generating a set of Java classes that represents the schema. All JAXB implementations provide a tool called *binding compiler* in order to bind a schema. In response, the binding compiler generates a set of interfaces and a set of classes that implement the interface. I obtained Java classes for each available XML levels and versions. I mean SBML level 1 version 1, level 1 version 2, level 2 version 1 and level 2 version 2. Later, I compiled and packaged them into just one package. The second step is to unmarshal an SBML document. Unmurshalling means creating a tree of content objects that represents the content and the organization of the document. The content tree is not a DOM-based tree. In fact, content trees produced through JAXB can be more efficient in terms of memory use than DOM-based trees. The content objects are instances of the classes produced by the binding compiler. In addition to providing a binding compiler, JAXB provides runtime APIs for JAXB-related operations such as marshalling. It is possible to validate source data against an associated schema as part of the unmarshalling operation. If the data is found to be invalid (that is, it doesn't conform to the schema) the JAXB implementation can report it and might take further action. JAXB providers have a lot of flexibility here. The JAXB specification mandates that all provider implementations report validation errors when the errors are encountered, but the implementation does

not have to stop processing the data. Some provider implementations might stop processing when the first error is found, others might stop even if many errors are found. In other words, it is possible for a JAXB implementation to successfully unmarshal an invalid XML document, and build a Java content tree. However, the result will not be valid. The main requirement is that all JAXB implementations must be able to unmarshal valid documents. I unmarshal and validate each SBML file at runtime.

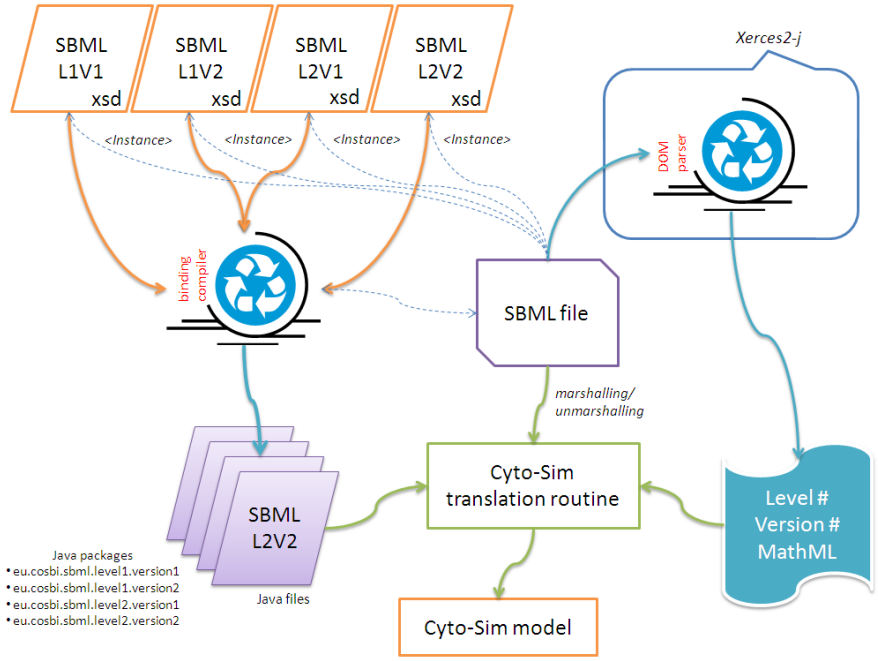

Fig. 2. Software Architecture for SBML Binding

The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document. The XML DOM is the tool to define (i) a standard set of objects for XML, (ii) a standard way to access XML documents; (iii) a standard way to manipulate XML documents. Cyto-Sim uses the DOM parser contained into xerces2-j [35] built into the Java Mustang release. The DOM parser is used to:

Check Levels and Versions: Cyto-Sim preliminary opens SBML files and checks levels and versions (delegating validation and comprehension to JAXB). It acquires knowledge about which JAXB context instantiating or, more clearly, which SBML schema considering for binding, unmarshalling and validation;

Parse MathML expression: Due to intrinsic limitations of JAXB to handle recursively nested xml tags, Cyto-Sim makes use of DOM to explore MathML expressions and parse their components.

5 Experimental Tests

The capability of Cyto-Sim to understand all currently existing SBML levels and versions has been tested on almost all official existing SBML files available on the web. I successfully imported all SBML files generated by Gepasi [30], a software package for modelling biochemical systems and the most part of the models stored into the BioModels database [25]. Gepasi makes available 9 SBML level 1 version 1 files [6] while BioModels has 70 curated and 43 not curated models exported as SBML level 2 version 1 files. I have also tested models from the PANTHER (130 SBML level 1 version 2 files) Classification System, and from KEGG (77 SBML level 2 version 1 files). All SBML files were converted from KEGG by using a conversion script *kegg2sbml*. Moreover, I retrieved some interesting models among all 238 CellML models and tested them. To do that, I had to manually convert from the CellML format to SBML by means of CellML2SBML [38] and later import and simulate them with Cyto-Sim. All imported files have been successfully parsed by Cyto-Sim. This testifies the quality of the conversion routines and of the architecture employed. Summarising, I retrieved 567 models from the most known and famous biological model containers available in SBML (or in formats having reference to SBML), and tested them. *I obtained a successful test, when Cyto-Sim had been able to correctly parse the inferred model.* In particular, now I am going to show a couple of examples which Cyto-Sim has been able not only to correctly parse, but also to simulate and get the same results shown in literature.

The first test is related to the model *BIOMD0000000010* picked up from the BioModels database. It concerns the functional organization of signal transduction into protein phosphorylation cascades and in particular the mitogen-activated protein kinase (MAPK) cascades. It greatly enhances the sensitivity of cellular targets to external stimuli [23]. In this paper it is demonstrated that a negative feedback loop combined with intrinsic ultrasensitivity of the MAPK cascade can bring about sustained oscillations in MAPK phosphorylation. The conversion of the SBML file produces the following model with 1 compartment, 8 species and 10 reactions.

```
object MKKK, MKKK_P, MKK, MKK_P, MKK_PP, MAPK, MAPK_P, MAPK_PP

rule J0 MKKK ((1.0*2.5*MKKK)/((1+((MAPK_PP/9.0)^1.0))*(10.0+MKKK)))-> MKKK_P
rule J1 MKKK_P ((1.0*0.25*MKKK_P)/(8.0+MKKK_P))-> MKKK
rule J2 MKK ((1.0*0.025*MKKK_P*MKK)/(15.0+MKK))-> MKK_P
rule J3 MKK_P ((1.0*0.025*MKKK_P*MKK_P)/(15.0+MKK_P))-> MKK_PP
rule J4 MKK_PP ((1.0*0.75*MKK_PP)/(15.0+MKK_PP))-> MKK_P
```

[6] among all, a very large model representing a set of 100 yeast cells in a liquid culture whose dynamics is represented by means of 2000 reactions

```
rule J5 MKK_P ((1.0*0.75*MKK_P)/(15.0+MKK_P))-> MKK
rule J6 MAPK ((1.0*0.025*MKK_PP*MAPK)/(15.0+MAPK))-> MAPK_P
rule J7 MAPK_P ((1.0*0.025*MKK_PP*MAPK_P)/(15.0+MAPK_P))-> MAPK_PP
rule J8 MAPK_PP ((1.0*0.5*MAPK_PP)/(15.0+MAPK_PP))-> MAPK_P
rule J9 MAPK_P ((1.0*0.5*MAPK_P)/(15.0+MAPK_P))-> MAPK

compartment uVol[J0, J1, J2, J3, J4, J5, J6, J7, J8, J9, 280.0 MAPK,
    10.0 MKK_P, 10.0 MKK_PP, 10.0 MKKK_P, 10.0 MAPK_PP, 280.0 MKK,
    10.0 MAPK_P, 90.0 MKKK]
system uVol

evolve 0-33000
plot uVol[MAPK,MAPK_PP]
```

In the figure 3, on the left is shown the simulation result coming from the literature and on the right that one obtained with Cyto-Sim. The graphs are identical.

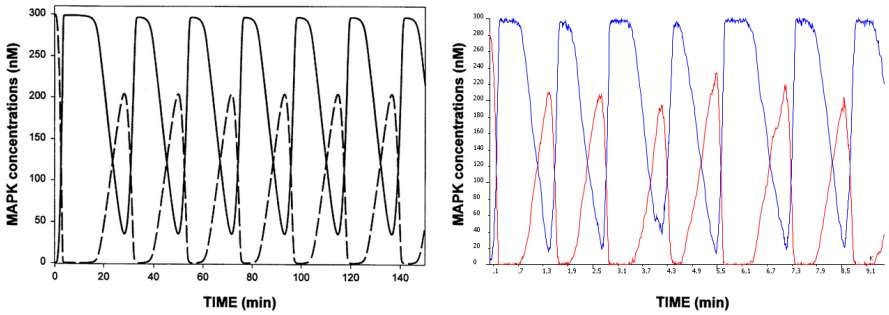

Fig. 3. Sustained oscillations in MAPK cascade

The second test is related to the glucose transport by the Bacterial Phosphoenolpyruvate [37] whose model has been found in JWS Online. The resulting model has 1 compartment, 17 species and 10 reactions.

```
object EI, PyrPI, EIP, HPr, EIPHPr, HPrP, EIIA, HPrPIIA, EIIAP, EIICB,
    EIIAPIICB, EIICBP, EIICBPGlc, PEP, Pyr, GlcP, Glc

rule v1 PEP + EI ((1960.0*PEP*EI)-(480000.0*PyrPI))-> PyrPI
rule v2 PyrPI ((108000.0*PyrPI)-(294.0*Pyr*EIP))-> EIP + Pyr
rule v3 HPr + EIP ((14000.0*EIP*HPr)-(14000.0*EIPHPr))-> EIPHPr
rule v4 EIPHPr ((84000.0*EIPHPr)-(3360.0*EI*HPrP))-> HPrP + EI
rule v5 HPrP + EIIA ((21960.0*HPrP*EIIA)-(21960.0*HPrPIIA))-> HPrPIIA
rule v6 HPrPIIA ((4392.0*HPrPIIA)-(3384.0*HPr*EIIAP))-> EIIAP + HPr
rule v7 EIICB + EIIAP ((880.0*EIIAP*EIICB)-(880.0*EIIAPIICB))-> EIIAPIICB
rule v8 EIIAPIICB ((2640.0*EIIAPIICB)-(960.0*EIIA*EIICBP))-> EIICBP + EIIA
rule v9 EIICBP + Glc ((260.0*EIICBP*Glc)-(389.0*EIICBPGlc))-> EIICBPGlc
rule v10 EIICBPGlc ((4800.0*EIICBPGlc)-(0.0054*EIICB*GlcP))-> EIICB + GlcP
```

```
compartment compartment_cyto_sim[v1, v2, v3, v4, v5, v6, v7, v8, v9, v10,
    0.0 EIICBPGlc, 5.0 EIICBP, 25.0 HPrP, 2.0 EIP, 20.0 EIIA, 5.0 EIICB,
    25.0 HPr, 2800.0 PEP, 0.0 PyrPI, 0.0 EIPHPr, 50.0 GlcP, 900.0 Pyr,
    0.0 HPrPIIA, 20.0 EIIAP, 500.0 Glc, 0.0 EIIAPIICB, 3.0 EI]
system compartment_cyto_sim

evolve 0-10000
plot compartment_cyto_sim[HPrP,EIIAPIICB,HPrPIIA]
```

In the figure 4 it is possible to notice that both graphs represent the same behaviour. The differences are due to the deterministic (on the left) or stochastic (on the right) nature of the simulations.

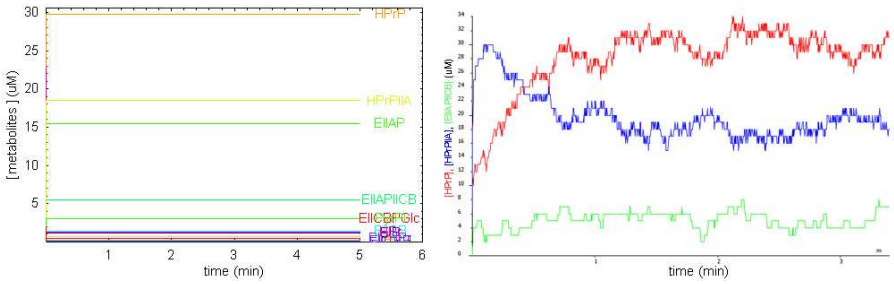

Fig. 4. Glucose Transport by the Bacterial Phosphoenolpyruvate

At the end, I tested the whole human reactome derived from Reactome. The actual release of the human reactome I used is an SBML file containing 28 compartments (even including internal membranes of the same compartment), 3054 species (in all their forms) and 1979 interactions represented by means of reactions. Cyto-Sim is able to parse and even to simulate it, although at the moment it cannot have meaning because of the lack of quantitative parameters (reaction rates and initial species quantities).

6 Conclusion

kosmopolitês (citizen of the world), has been used to describe a wide variety of important views in moral and socio-political philosophy. The nebulous core shared by all cosmopolitan views is the idea that all human beings, regardless of their political affiliation, do (or at least can) belong to a single community, and that this community should be cultivated. Different versions of cosmopolitanism envision this community in different ways, some focusing on political institutions, others on moral norms or relationships, and still others focusing on shared markets or forms of cultural expression [24].

In the context of the present work, a citizen of the world is anyone who speaks and understands a common language, who can travel to the ends of the earth without worrying about misunderstanding or being misunderstood. Limited comprehension of language is the greatest barrier for people who need to spread information and ideas. This is exactly the case for scientists who wish to share their results and models with the widest possible audience.

In this paper I have presented an extended overview of the SBML story, continually remarking on the increasing interest of scientists both to support and write their biological models in SBML. I talked about the internal structure of SBML, in order to focus on its expressive potentialities, and later I presented a possible software architectural arrangement to allow simple binding to SBML schemas and correct unmarshalling of SBML files. Finally, I presented tests performed on two models coming from separate databases. I demonstrated the correctness of the translation routines and highlighted the similarities of the obtained simulation results.

Today there are more than 600 models written in SBML, ready to be more accurately studied, confirmed or refuted. Challenging existing knowledge is the means to increase understanding and therefore to *grow* knowledge. The best way to achieve this is to maximize the number of people that speak the same language, in this case SBML. My work sits perfectly in this context and my hope is that it has wide application.

References

1. Brown et al. Altman, R.B. Ribonucleic acid markup language. http://www.smi.stanford.edu/projects/helix/riboml/, 2002.
2. F. T. Bergmann and H. M. Sauro. Sbw - a modular framework for systems biology. In *Proceedings of the 37th conference on Winter simulation*, pages 1637 – 1645. Winter Simulation Conference, 2006.
3. M. Cavaliere and S. Sedwards. Modelling cellular processes using membrane systems with peripheral and integral proteins. *Computational Methods in Systems Biology, Lecture Notes in Computer Science series*, 4210/2006:108–126, 2006.
4. A.A. Cuellar, C.M. Lloyd, P.F. Nielsen, D.P. Bullivant, D.P. Nickerson, and P.J. Hunter. An overview of cellml 1.1, a biological model description language. *Simulation*, 79(12):740–747, 2003.
5. A. Deckard, F. T. Bergmann, and H. M. Sauro. Supporting the sbml layout extension. *Bioinformatics*, 22(23):2966–2967, October 2006.
6. C. Eccher and C. Priami. Design and implementation of a tool for translating sbml into the biochemical stochastic pi-calculus. *Bioinformatics*, 22(24):3075–308, October 2006.
7. D. Fenyo. The biopolymer markup language. *Bioinformatics*, 15(4):339–340, 1999.
8. A. Finney. *Developing SBML Beyond Level 2: Proposals for Development*, volume 3082/2005, pages 242–247. Springer Berlin / Heidelberg, April 2005.
9. A.M. Finney and M. Hucka. Systems biology markup language: Level 2 and beyond. *Biochem Soc Trans*, 31:1472–1473, 2003.

10. M. Gheorghe. P system modelling framework. *http : //www.dcs.shef.ac.uk/ marian/PSimulatorWeb/P_Systems_applications.htm*, 2006.
11. Project GlassFish. The jaxb project. https://jaxb.dev.java.net/.
12. D. Hanisch, R. Zimmer, and T. Lengauer. Proml - the protein markup language for specification of protein sequences, structures and families. *In Silico Biology*, 2(3):313–324, 2002.
13. W.J. Hedley, M.R. Nelson, D.P. Bullivant, and P.F. Nielson. A short introduction to cellml. *Philos. Trans. R. Soc. Lond. A*, 359:1073–1089, 2001.
14. H. Hermjakob and et al. Montecchi-Palazzi. The hupopsis molecular interaction format - a community standard for the representation of protein interaction data. *Nature Biotechnol.*, 22(2):177–183, 2004.
15. M. Hucka and A. Finney. Escalating model sizes and complexities call for standardized forms of representation. *Molecular Systems Biology*, 1(2005.0011):Published online, May 2005.
16. M. Hucka, A. Finney, B.J. Bornstein, S.M. Keating, B.E. Shapiro, J. Matthews, B.L. Kovitz, M.J. Schilstra, A. Funahashi, J.C. Doyle, and H. Kitano. Evolving a lingua franca and associated software infrastructure for computational systems biology: the systems biology markup language (sbml) project. *Systems Biology*, 1(1):41–53, June 2004.
17. M. Hucka and A. et al. Finney. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.
18. Doubletwist Inc. Agave: architecture for genomic annotation, visualization and exchange. http://www.agavexml.org, 2001.
19. LabBook Inc. Bsml (bioinformatics sequence markup language) 2.2. http://www.labbook.com/products/xmlbsml.asp, 2002.
20. G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, GR Gopinath, GR Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein. Reactome: a knowledgebase of biological pathways. *Nucleic Acids Res.*, 33:D428–D432, January 2005.
21. M. Kanehisa and S. Goto. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.
22. S. M. Keating, B. J. Bornstein, A. Finney, and M. Hucka. Sbmltoolbox: an sbml toolbox for matlab users. *Bioinformatics*, 22(10):1275–1277, 2006.
23. B. N. Kholodenko. Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. *Eur. J. Biochem*, 267:1583–1588, 2000.
24. P. Kleingeld and E. Brown. Cosmopolitanism. In *The Stanford Encyclopedia of Philosophy*. Edward N. Zalta, Winter 2006.
25. N. Le Novre, B Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. Sauro, M. Schilstra, B. Shapiro, J. L. Snoep, and Hucka M. Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34:D689–D691, 2006.
26. N. Le Novre, A. Finney, and Hucka M. et. al. Minimum information requested in the annotation of biochemical models (miriam). *Nature Biotechnology*, 23:1509–1515, 2005.
27. Y.M. Liao and H. Ghanadan. The chemical markup language. *Anal. Chem.*, 74(13):389A–390A, 2002.

28. R. Machn, A. Finney, S. Mller, J. Lu, S. Widder, and C. Flamm. The sbml ode solver library: a native api for symbolic and fast numerical analysis of reaction networks. *Bioinformatics*, 22(11):1406–1407, March 2006.
29. D.C. McArthur. An extensible xml schema definition for automated exchange of protein data: Proximl (protein extensible markup language). http://www.cse.ucsc.edu/douglas/proximl/, 2001.
30. P. Mendes. Gepasi: a software package for modeling the dynamics, steady states, and control of biochemical and other systems. *Comput. Applic. Biosci.*, 9:563–571, 1993.
31. H. Mi, B. Lazareva-Ulitsky, A. Loo, R. amd Kejariwal, J. Vandergriff, S. Rabkin, N. Guo, A. Muruganujan, O. Doremieux, and M.J. et al. Campbell. The panther database of protein families, subfamilies, functions and pathways. *Nucleic Acids Res*, 33:D284–D288, 2005.
32. I. Nepomuceno, J.A. Nepomuceno, and F.J. Romero-Campero. A tool for using the sbml format to represent p systems which model biological reaction networks. In *Third Brainstorming Week on Membrane Computing*, pages 219–228, Jan 2005.
33. The Microsoft Research University of Trento. Centre for Computational and Systems Biology. Web page of cyto-sim. $http://www.cosbi.eu/Rpty_Soft_CytoSim.php$, 2006.
34. B.G. Olivier and J.L. Snoep. Web-based modelling using jws online. *Bioinformatics*, 20:2143–2144, 2004.
35. Apache XML project. Xerces2 java parser 2.9.0. http://xml.apache.org/xerces2-j/, 2004.
36. N. Rodriguez, M. Donizelli, and N. Le Novre. Sbmleditor: effective creation of models in the systems biology markup language (sbml). *Bioinformatics*, 8(79):Published online, March 2007.
37. J. M. Rohwer, N. D. Meadowi, S. Rosemani, H. V. Westerhoff, and P. W. Postma. Understanding glucose transport by the bacterial phosphoenolpyruvate:glycose phosphotransferase system on the basis of kinetic measurements in vitro. *The Journal of Biological Chemistry*, 275(45):34909–34921, March 2000.
38. M. J. Schilstra, L. Li, J. Matthews, A. Finney, M. Hucka, and N. Le Novre. Cellml2sbml: conversion of cellml into sbml. *Bioinformatics*, 22(8):1018–1020, February 2006.
39. B. E. Shapiro, M. Hucka, A. Finney, and Doyle J. Mathsbml: a package for manipulating sbml-based biological models. *Bioinformatics*, 20(16):2829–2831, November 2004.
40. P.T. Spellman and M. et al. Miller. Design and implementation of microarray gene expression markup language (mage-ml). *Genome Biol.*, 3(9):0046.0041–0046.0049, 2002.
41. C.F. Taylor and N.W. et al. Paton. A systematic approach to modeling, capturing, and disseminating proteomics experimental data. *Nature Biotechnol.*, 21:247–254, 2003.
42. B. L. Wanner, Finney A., and M. Hucka. *Modeling the E. coli cell: The need for computing, cooperation, and consortia*, volume 13 of *Topics in Current Genetics*, pages 163–189. Springer Berlin / Heidelberg, May 2005.
43. Z. Zhike and E. Klipp. Sbml-pet: a systems biology markup language-based parameter estimation tool. *Bioinformatics*, 22(21):2704–2705, 2006.