Representation of Biochemical Pathway Models

(Issues relating conversion of model representation from SBML to a commercial tool)

Master's Thesis

Sudhir Naswa (a03sudna@student.his.se)
School of Humanities and Informatics
University of Skövde, Box 408
S-54128, Skövde, SWEDEN

Master's dissertation, February 2005
Masters Degree in Bioinformatics

Academic Supervisor: Kim Laurio
Industrial Supervisor: Henrik Johanssson

Representation of Biochemical Pathway Models
(Issues relating conversion of model representation from SBML to a commercial tool)

Submitted by Sudhir Naswa to the University of Skövde as a dissertation towards the degree of M.Sc. by examination and dissertation in the School of Humanities and Informatics.

February 2005

I hereby certify that all material in this dissertation which is not my own work has been identified and that no material is included for which a degree has already been conferred upon me.

_____

(SUDHIR NASWA)

Acknowledgments

I would like to thank my academic supervisor, Kim Laurio at University of Skövde, and industrial supervisor Henrik Johansson at InNetics AB, Sweden for their guidance throughout the duration of this project. Their inputs and comments have made a very valuable contribution in this work.

I would also like to thank Björn Olsson at University of Skövde and Johan Gunnarsson at InNetics AB, for conceptualising this project and for their guidance in the initial stages of work.

I also wish to thank my family for their support in all my endeavors, and for always believing in me.

Representation of Biochemical Pathway Models

(Issues relating conversion of model representation from SBML to a commercial tool)

Sudhir Naswa
School of Humanities and Informatics, University of Skövde, 541 28 Skövde, Sweden
a03sudna@student.his.se

Abstract

Background: Computational simulation of complex biological networks lies at the heart of systems biology since it can confirm the conclusions drawn by experimental studies of biological networks and guide researchers to produce fresh hypotheses for further experimental validation. Since this iterative process helps in development of more realistic system models a variety of computational tools have been developed. In the absence of a common format for representation of models these tools were developed in different formats. As a result these tools became unable to exchange models amongst them, leading to development of SBML, a standard exchange format for computational models of biochemical networks. Here the formats of SBML and one of the commercial tools of systems biology are being compared to study the issues which may arise during conversion between their respective formats. A tool StoP has been developed to convert the format of SBML to the format of the selected tool.

Results: The basic format of SBML representation which is in the form of listings of various elements of a biochemical reaction system differs from the representation of the selected tool which is location oriented. In spite of this difference the various components of biochemical pathways including multiple compartments, global parameters, reactants, products, modifiers, reactions, kinetic formulas and reaction parameters could be converted from the SBML representation to the representation of the selected tool. The MathML representation of the kinetic formula in an SBML model can be converted to the string format of the selected tool. Some features of the SBML are not present in the selected tool. Similarly, the ability of the selected tool to declare parameters for locations, which are global to those locations and their children, is not present in the SBML.

Conclusions: Differences in representations of pathway models may include differences in terminologies, basic architecture, differences in capabilities of software's, and adoption of different standards for similar things. But the overall similarity of domain of pathway models enables us to interconvert these representations. The selected tool should develop support for unitdefinitions, events and rules. Development of facility for parameter declaration at compartment level by SBML and facility for function declaration by the selected tool is recommended.

Background

Biochemical pathways

Cellular life is characterized by a vast number of direct or indirect interactions of cellular components. These intricate networks of functional and physical interactions between molecular species in the cell are called biochemical pathways (Deville et al., 2003). Biochemical pathways include metabolic pathways, gene regulatory pathways and signalling pathways. Biochemical pathways may consist of a linear series of steps, may be cyclic or they may form complex networks. They can also have feedback loops which help in regulation of cellular systems and prevent them from drifting away according to varying inputs or random fluctuations of their state variables (Cinquin and Demongeot, 2002). Knowledge of biochemical pathways helps understanding metabolism (Schilling et al., 1999). They also help understanding the reasons for many diseases and ways to treat them (Thagard, 2003). Algorithms are being developed to identify the ways to treat diseases by minimal disturbance to rest of the network (Klamt and Gilles, 2004).

Computational modelling of pathways: Improved knowledge of genomics and the structure of individual proteins have resulted in increased understanding of biological systems. However, insight into functional interactions between the key components of cells, organs, and systems help us in understanding their physiology. The perturbations in these interactions lead to various diseases. Computation of these interactions has therefore become necessary to determine the characteristics of the system which change from healthy to the diseased state. The development of powerful computing hardware and algorithms, increasing number of pathway databases and models of cells, tissues and organs have made it possible to explore functionality in a mathematical manner all the way from the level of genes to the physiological function of whole organs and regulatory systems (Noble, 2002).

The simplified mathematical representation of the dynamics of a system is called modelling (Hammer et al., 2004). Modelling has become an important research area in biology and bioinformatics. Models are used for explaining the experimental observations. Hence, they can be used for testing a hypothesis about biological function. They are also used for storing experimental data on biological molecules and processes in databases so as to analyze it (Deville et al., 2003). While individual reactions are being modelled since long, the importance of modelling complex reactions, biochemical pathways and networks has only recently begun to be appreciated (Crampin and Schnell, 2004). Since experimental data on biochemical reactions is insufficient and obtaining it is difficult, expensive and time consuming, computational models of biological networks help in fulfilling this data gap. Computational models can be used both for simulation and metabolic engineering (Deville et al., 2003). By computational simulation of complex biological networks, we can not only validate the conclusions drawn by experimental studies but also propound fresh hypotheses for further experimental validation. This iterative process of experimental studies and computational simulation has helped in development of highly sophisticated and realistic models, for example, models of heart cells (Noble and Rudy, 2001).

Modelling approaches: There are a variety of modelling approaches that have different strengths, weaknesses, and domains of applicability. Three basic types of models are Boolean models, differential equation models and stochastic models (Mjolsness and Gibson, 2001, 15-40)

Boolean models are usually used for gene regulatory networks where each gene can be in one of two states i.e. expressed (“1”) and not expressed (“0”). A particular state of these models consists of a list of genes along with the status of each of them. Being Boolean models they are relatively simple and it is possible to move deterministically from one state to another. Besides ‘expressed’ and ‘not expressed’ states, in reality genes also have intermediate levels of expression, which cannot be represented in these models. Hence, they are often used as a first representation of a complex system (Mjolsness and Gibson, 2001).

Differential equation models assume that concentrations of species involved in reaction vary deterministically and continuously over time. The state here is a list of concentrations of all chemical species involved. The reaction kinetics (or enzyme kinetics) is represented in terms of ordinary differential equations (ODEs) whose variables are concentrations of various compounds like proteins, mRNA, etc. (Mjolsness and Gibson, 2001). Despite being a very popular method all biological systems can not be modelled with differential equations since inside a cell the situation is not continuous and predictable and random fluctuations drive many of the reactions (Greenwald, 1998). Some biological systems for example the gene expression may be stochastic in nature (Fiering, Whitelaw and Martin, 2000). Even in genetically identical cells having the same concentrations and states of their cellular components, the rate of expression of a particular gene varies from cell to cell because of the random microscopic events that govern which reactions occur and in what order. This inherent stochasticity or intrinsic noise fundamentally limits the precision of gene regulation (Elowitz et al., 2002). Hence, a different approach is required for modelling of such cellular processes.

Stochastic models became popular after the publication of Gillespie’s algorithm (Gillespie D.T., 1977). In these models instead of using concentrations, the number of molecules (a discrete quantity) present in the system is used. The state indicates how many molecules of each type are present in the system. Changes in state occur discretely, but which change occurs and when it occurs is probabilistic. The rate constants specify the probability per unit time of a discrete event happening (Mjolsness and Gibson, 2001).

Some modelling approaches are intermediate to the above three approaches. For example, kinetic logic models as proposed by Thomas and D’Ari (1990) and Thomas, Thieffry and Kaufman (1995) represent the state of each gene as a discrete value. But they have finer granularity than Boolean networks. The states of genes in these models can be stated as "not expressed", "expressed at a low level" ,"expressed at a medium level" or "fully expressed" etc. instead of just two states of “on” and “off” in Boolean models.

Continuous logic models lie between Boolean and differential equation models (Mestl, Pholte and Omholt. 1995). These contain discrete states,

but transition from one state to another is governed by linear differential equations with constant coefficients.

Top down and bottom up modelling: Using computer modelling we want to precisely simulate the behavior of biological systems (Katagiri, 2003). The traditional bottom up, or reductionist, approach of understanding the biological systems involves breaking up the system into smaller parts and then reconstituting a model for the whole biological system by combining the pieces. This reconstitution has proven to be difficult because of a lack of complete understanding of all the parts of the system and their dynamics.

Another approach of modelling biological systems is the top down approach. This systems level approach is followed by systems biology where the focus is on understanding structure and dynamics of a system (Kitano, 2001). In systems biology data is acquired at the level of DNA, RNA, proteins, and phenotype from experimental systems to create mathematical models of the informational networks and behaviour of cells. An iterative process of modelling and wet bench testing is used to obtain parameters controlling the metabolism. Being a holistic approach, systems biology involves simultaneous handling of a large number of parameters and requires a range of software resources. A variety of computational tools have been developed recently for systems biology. However, these tools were developed in different formats because of the lack of a common format. As a result these tools became incompatible with each other and hence unable to exchange models amongst each other. Therefore this work has focused on investigating a standard developed for representation of biochemical pathways called Systems Biology Markup Language (SBML) (Finney and Hucka, 2003a). A commercial systems biology tool has been selected for comparing its representation of pathway models with the SBML representation.

PathwayLab

PathwayLab[1] is an ODE based tool for modeling and simulation of biochemical pathways. Its graphic user interface contains various objects which can be used to construct biochemical models. Each of these objects has a precise mathematical meaning (Jirstrand, Gunnarsson, and Johansson, 2004). Hence when a model is constructed in it by a simple drag and drop method, it is internally converted into a symbolic representation. This XML-based representation is unique to this tool.

Marking up biology

Over the past 20 years, the biological sciences have experienced an ‘‘information explosion’’ (Hunter and Borg, 2003) and highly interrelated and rapidly evolving data is being generated (Achard, Vaysseix and Barillot, 2001). It is becoming increasingly difficult to manage this data. Management of these data sets has been recognized as a key component of genome/biology research (Robbins, 1994). The non-management of this information is a bottleneck in the development of the field (Reichardt, 1999). Besides being of large size the data is complex in nature, consisting of different types. It is updated frequently and new data emerge regularly which may change the perception of old data. As a result new relationships of data may appear that had previously no or only weak links (Achard, Vaysseix and Barillot 2001). To address the above problems there was a need for a flexible and standard specification for storage of this data. In recent years XML-based standards have been defined for representing data relating to various domains of biology.

XML (eXtensible Markup Language) is a metadata language, meaning that it is data about data. It is derived from the Standard Generalized Markup Language (SGML), the international standard for defining descriptions of structure and content of electronic documents. The specifications of XML are being supervised by the World Wide Web consortium since its inception in 1996. As the name suggests it is a markup language which means XML tags can be used to identify different portions of documents. The word extensible implies that there is no one single set of XML tags. The tags can be extended as per requirement. Although XML is extensible, different standard tags are being defined which have a particular meaning for a particular type of application. The use of these standard tags makes the document transportable across any type of computer platform or processing program (Martin T.A., 1999, 29-48). It has been claimed that XML may be adopted in bioinformatics as a

[1] Developed by InNetics AB, Sweden. http://innetics.com/

standard for data exchange (Achard, Vaysseix and Barillot 2001) since it is simple, flexible, Internet-oriented open framework for defining standard specifications. Recently XML is being used increasingly in bioinformatics and it is expected that in the future data interchange in bioinformatics databases will take place in XML (Bruhn and Burton, 2003). Many XML-based markup languages relating to diverse fields of life sciences are being used currently, some of which are indicated below.

BIOML (BIOpolymer Markup Language): It is used for annotation of experimental information about protein and nucleotide sequences. (Fenyö, 1999).

TML (Taxonomic Markup Language): It is used for the description of taxonomic relationships between organisms (Gilmour, 2000).

MAGE-ML (MicroArray and Gene Expression Markup Language): It is a specification language for micro-array data. It can be used not only for the data generated by gene expression experiments, but also for data from all DNA micro-array experiments and technologies (Spellman et al., 2002).

BSML (Bioinformatics Sequence Markup language): It is used for description of biological sequences (DNA, RNA protein), their features and their relationships to other kinds of biological data and resources (Blyden et al., 2002).

RNAML (RNA Markup Language): It is a standard syntax for storage and exchange of RNA information including its sequence, secondary and tertiary structures (Waugh et al., 2002).

ProML (Protein Markup Language): It is a specification language for protein sequences, structures, and families including the representation of amino acid sequences, PROSITE patterns, secondary structure elements, tertiary structure data, and geometric constraints. It has been developed as a means for structured representation of protein properties in the context of structural genomics (Hanisch, Zimmer and Lengauer 2002).

Just like XML standards for data related to other branches of biology, CellML and SBML are two XML based standards used for representation of biochemical pathways.

CellML: It is a free, open-source standard for defining mathematical models of cellular function. Scientists at the University of Auckland (in the Bioengineering Institute) and at Physiome Sciences, Inc are developing it. CellML models are represented as a network of interconnected but discrete components. A component is the smallest functional unit of a model (a single component can represent a valid CellML model). The components may correspond to real biological and/or physical entities such as an organelle, or an ion channel or pump, or they can represent a convenient modelling abstraction, such as a component used to store all the constants in a model. Each component must contain at least one variable. Variables can be passed between various components via their interfaces and connections. A component may also contain mathematical equations to describe how it behaves within the model. The equations are expressed using MathML (an XML based language for expressing mathematical formulas and equations) (Lloyd, Halstead and Nielsen, 2004). The component based architecture of CellML facilitates the reuse of models and parts of models. CellML can be used to represent many different types of models. Therefore, its basic structure is very general. Models are primarily specified by explicitly defining mathematics using MathML. It is even possible to specify a model purely in terms of mathematics, without using any of the elements. However, in some types of models, information is lost when reducing the model to pure mathematics. For instance, in biochemical pathway models it will not always be straightforward, or even possible, to unambiguously determine from the mathematical rate laws which variables represent inhibitors or activators in the reactions. Therefore, some additional elements are used in CellML to fully capture the information in biochemical pathway models (Cuellar et al., 2003). For example, a reaction element is present which stores information associated with a reaction. Associated with this reaction element are variables declaring the names and roles of the substrates involved in the reaction, such as reactants, products, catalysts and inhibitors. There is also a variable declared as rate, which often has a MathML expression associated with it to define the rate equation of the reaction. CellML also provides the ability to define the

stoichiometry, directionality, and reversibility of a reaction (Lloyd et al., 2004).

SBML (Systems Biology Markup Language): It is a standard exchange format for computational models of biochemical networks. However, unlike in CellML in SBML it is not possible to build large models out of instances of other models i.e. at present it does not allow the reuse of model components (Finney and Hucka, 2003a). CellML is more abstract and general and describes the structure and underlying mathematics of cellular models in a very general way. However, SBML is better suited for interoperability with existing simulation tools since its developers interacted with various existing simulation packages to take into consideration their differing formats (Finney and Hucka, 2003b).

Since different tools have their own format for representing the pathway models, they must be able to convert their format to a standard format and vice versa in order to import or export models to and from the other tool. However, various issues relating to differences in formats, which may affect the possibility of these conversions, may arise. To examine these issues, formats of the commercial tool PathwayLab and SBML were compared. The tool StoP (SBML to PathwayLab) was developed to convert the format of SBML to PathwayLab.

Implementation

For studying the issues, which may arise during the conversions between the SBML format and the format of commercial tools, PathwayLab was used as a representative example of such commercial tools. The representation of biochemical pathways in SBML and PathwayLab was compared. For comparison of formats a known SBML model was selected and constructed in PathwayLab. The model constructed in PathwayLab was simulated. The simulations with PathwayLab were compared with its simulations in an online tool JWS Online [2]. In JWS online various models are available. These models are already constructed and can be simulated at various parameter settings. When the simulations of the selected model were similar for both these tools the XML representation of this model was extracted from PathwayLab and it was mapped with the SBML format. Using this mapping a tool StoP was developed to convert the SBML format to PathwayLab format. The main stages of implementation are shown in Fig. 1.

Selection of a suitable known model having an SBML representation: All the models given in the SBML model repository[3] were examined and it was found that most of the models were of SBML level 1 (version 1). Only three models were of SBML level 2. The following criteria were used to select the models:

a) The model shall cover as many features of SBML (level 2 version 1) which are supported by PathwayLab as possible.

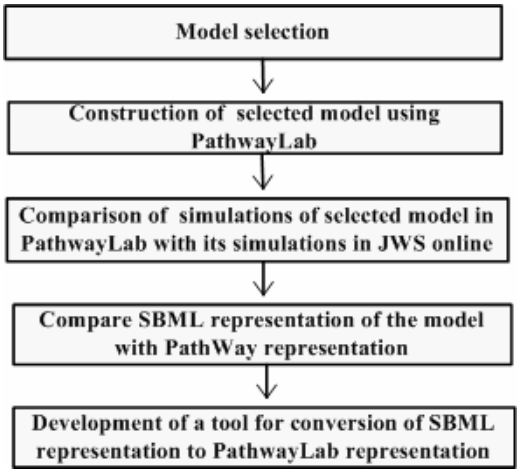


Fig. 1: Stages of implementation.

b) Some of the models provided in the SBML repository have links to the online simulation facility JWS Online. It was considered important that the selected model shall be one amongst these since it will enable us to compare the simulations cited by creators of the SBML for the selected model with the simulations of same model when constructed using PathwayLab. Since the construction of any model using any tool depends on how correctly the user develops it, for checking the correctness of construction of the selected model using PathwayLab, it was considered prudent to compare its simulations with the simulations of the same model which is already constructed in the JWS Online facility.

[2] This tool is available online at http://jjj.biochem.sun.ac.za/index.html. [Accessed on 12 Dec. 2004.]

[3] It is a collection of models in SBML format available at http://sbml.org/models/ .It contains 29 models. [Accessed on 8 Feb. 2005.]

Simulation comparison at this stage was considered vital because similar simulations by both the tools made sure that the selected model was developed correctly using PathwayLab. None of the three SBML level 2 models had this online simulation facility. Only the following five SBML level 1 (version 1) models had an online simulation facility (JWS online cellular systems modelling):

Glycolysis in *Saccharomyces cerevisiae:* This model of glycolysis in yeast was used by Teusink et al. (2000) while examining whether *in vivo* behavior of yeast glycolysis can be understood in terms of the *in vitro* kinetic properties of the constituent enzymes.

Glycolysis in *Trypanosoma brucei:* This model was developed by Helfert et al. (2001) to study the role of triosephosphate isomerase and aerobic metabolism in *Trypanosoma brucei.*

Pyruvate branches in *Lactococcus lactis:* This model of glycolysis in *Lactococcus lactis* was set up by Hoefnagel et al. (2002) while doing metabolic engineering experiments aimed at optimization of the production of di-acetyl, a by-product of glycolysis, but an important flavor component of dairy products such as butter.

Pyruvate branches in *Lactococcus lactis* - full Glycolysis: This model of glycolysis in *Lactococcus lactis* was also set up by Hoefnagel et al. (2002) while doing metabolic engineering experiments as stated in the paragraph above.

Glycogenolysis in skeletal muscle: This model was developed by Lambeth and Kushmeric (2002) to explore the consequences of the use of irreversible reaction steps in metabolic modelling .They also observed the consequences of coupling glycogenolysis to other reactions within the cell on the regulation of its pathway flux.

Since SBML level 2 is the latest available SBML format it was considered necessary to have SBML level 2 representation of the selected model. Hence, the above five models were converted to SBML level 2 using the online LibSBML converter [4] . The SBML level 2 representations of these five models were then compared to find the model which covers most features of SBML level 2. These comparisons (shown in Appendix 1) revealed that out of these five models the model 'Glycolysis in *Trypanosoma brucei'* covers one additional feature 'Rules' which is absent in the other four models whereas the models 'Glycogenolysis in skeletal muscle' and 'Pyruvate branches in *Lactococcus lactis*- full Glycolysis' contained modifiers which were absent in the remaining three models. Since the presence of modifiers is a basic feature of most chemical reactions, it was decided to choose one of the two models having this feature. This feature was also important because in PathwayLab these elements have a distinct representation using 'control elements' which are one of the basic elements of PathwayLab representation. Since the main idea behind the selection of model was to compare its SBML representation with its representation in PathwayLab it did not matter which of these two models was selected. The representation comparisons would be the same with either of the two. Accordingly the model 'Glycogenolysis in skeletal muscle' was selected out of these two models.

Construction of the selected model using PathwayLab: The model 'Glycogenolysis in skeletal muscle' was constructed in PathwayLab (Appendix 4). The value of all model constituents viz. concentrations of substrate, concentrations of product and parameter values were kept the same as in the selected SBML model. This would enable obtaining similar simulations of the model in PathwayLab and the JWS Online cellular system modelling software.

Comparison of simulations of selected model in PathwayLab with simulations in JWS Online: The model was simulated at different sets of parameter settings. For each set of parameter settings the simulations in both the tools produced similar graphical outputs (Appendix 5). Similar simulations indicated that the selected model has been constructed correctly in PathwayLab.

Mapping of representation of PathwayLab and SBML: After achieving similar simulation results

[4] It is provided by developers of SBML and is available online at http://sbml.org/tools/htdocs/sbmltools.php. [Accessed on 12 Dec. 2004.]

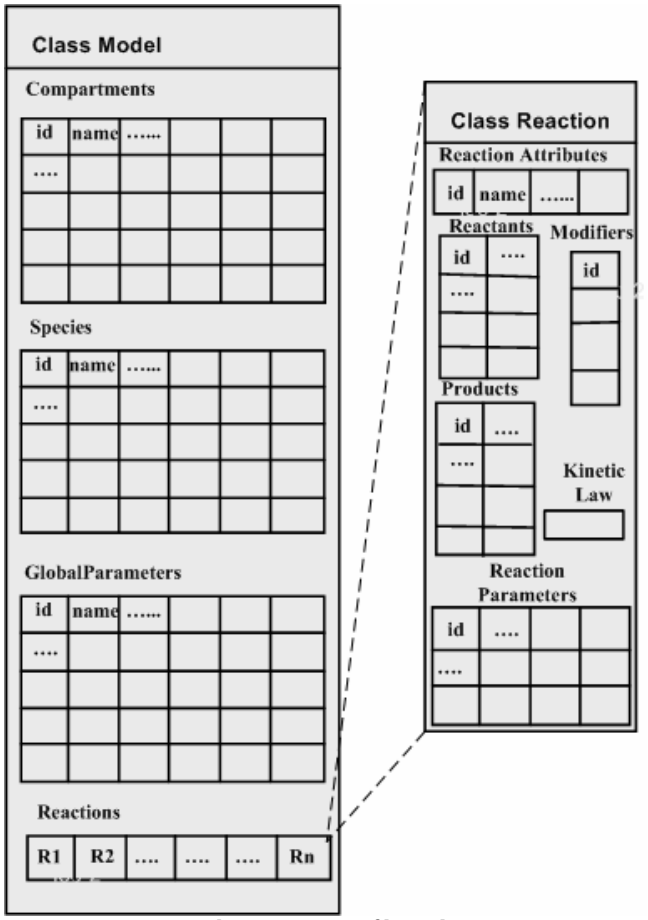


Fig. 2: Pictorial representation of basic data structure.

of the model 'Glycogenolysis in skeletal muscle' its XML representation in PathwayLab was extracted for comparisons with its SBML representation and for generating the mapping between the two representations. The corresponding elements in both formats were mapped. A tabular representation of the mapping is given in Appendices 2 and 3.

Conversion of SBML representation to PathwayLab representation (Import of SBML model to PathwayLab): Once the mappings between the representations of SBML and PathwayLab were completed a tool was implemented which could convert the SBML representation of a given model to PathwayLab representation. For implementation purposes concentration was laid on the features of the selected model i.e. 'Glycogenolysis in skeletal muscle'. The main implementation was done using C#. The free source library LibSBML was used to convert the MathML format of kinetic formulas in SBML to the string format of kinetic formula in PathwayLab. Since LibSBML does not have direct binding with C# a DLL was made in managed C++ for accessing LibSBML library (Fig. 9).

For implementing the conversion of format of the model from SBML to PathwayLab it was decided to first read it in to a data structure which is generic to the extent that in case conversions are to be done from SBML to any other tool the same data structure could be used. This was made possible by storing all the SBML features of the selected model in the data structure. Since the data structure stores all the features of the selected model, the SBML reading feature of the tool can be used without any modifications. The only difference when the implementation is done with a tool other than PathwayLab would therefore be in writing the model to the specific format of that tool. Broadly the data structure consisted of two classes (Fig. 2). One class consisted of matrices representing the compartments, various species and global parameters and a vector of objects of the second class representing the reactions. This second

class for reactions consisted of matrices representing reactants, products, modifiers and reaction parameters and a string element in which the MathML element of the kinetic law was stored. After storing the model in the data structure each of its features was read and translated to the corresponding feature of PathwayLab format as per the mapping. PathwayLab specific features were inserted at appropriate places with their default values.

It was observed that the major difference between PathwayLab and SBML representation is that whereas the SBML representation is 'component oriented' the PathwayLab representation is 'location oriented'. Hence in SBML every component of the model e.g. reactant, product, reaction, parameter etc. is given as a list of that component where its details are mentioned. In contrast in PathwayLab each component is a child of its location (equivalent to compartment of SBML) i.e. we have location(s) and within that other locations each of them with their respective reactants, products, modifiers and reactions. Moreover the PathwayLab representation is GUI (Graphic User Interface) oriented. As a result PathwayLab representation is more verbose as compared to SBML representation. Another major difference is that PathwayLab represents kinetic formulas as a string whereas SBML uses MathML for representing kinetic formulas. Some of the features of SBML representation like events, functions and unit definitions are not supported by PathwayLab. Moreover PathwayLab does not fully support rules of SBML. Out of three types of rules of SBML it only supports the assignment rules. The detailed mappings between SBML and PathwayLab are given in Appendices 2 and 3. However, major differences between the two formats and how they were dealt with are discussed below.

Compartments: In a typical biological scenario

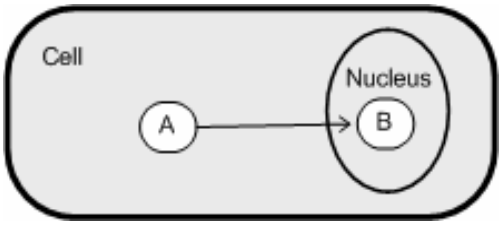


Fig. 3: A simple reaction system showing two species and two compartments. Corresponding SBML and PathwayLab representations showing the relationship between the species and compartment are shown in Fig. 4.

```
<sbml .....>
 <model id="ODEExampleModel">

  <listOfCompartments>
   <compartment id="C" name="Cell"/>
   <compartment id="N"
          name="Nucleus" outside ="C" />
  </listOfCompartments>

  <listOfSpecies>
   <species id="X0" name="A"
                  compartment="C"/>
   <species id="S1" name="B"
                  compartment="N"/>
  </listOfSpecies>
        ..............

 </model>
</sbml>
```

SBML

```
<document >
    <name>Cell </name>
     ..........
    <children>
      <entity>
        <name>A </name>
         ..........
      </entity>
      <location>
          <name>Nucleus </name>
           ..........
          <children>
            <entity> <name>B </name>
              ..........
            </entity>
          </children>
      </location>
    </children>
</document>
```

PathwayLab

Fig. 4: Representation of relationship of species (entity in PathwayLab) and compartments (called location in PathwayLab) of the reaction system shown in Fig. 3 above. The broken lines shown in PathwayLab representation is not its part. They show that the locations are included within one another and entities are located within the compartments. (SBML, portion of diagram adapted with minor variations from Finney and Hucka, 2003b).

each species must be located within a compartment. Hence, for defining a species (reactant, product, or modifier) the compartment in which that species is present must have been defined. For representing a species 'A' in a compartment 'Cell' (Fig. 3) in SBML representation the compartment cell is defined first within the tags <listOfCompartments> as indicated in Fig. 4. The species 'A' which is defined within the tags <listOfSpecies> has as its attribute the unique Id 'C' of the compartment 'Cell' implying presence of the species 'A' in the compartment with Id 'C'. Similarly, when there is a compartment within a compartment it is mentioned by indicating the 'Id' of outer compartment in the outside field of inner compartment (Finney and Hucka, 2003b). In Fig. 4 compartment 'Nucleus' has Id of the compartment cell as the value of its outside attribute. Hence, in SBML representation the compartments are represented as a list and are not arranged in a parent-child relation either with other compartments or with species.

In contrast in PathwayLab representation the compartments (locations) are arranged in a hierarchy of parent child relations (Fig. 4). A compartment may have other compartments, entities (equivalent to species in SBML),

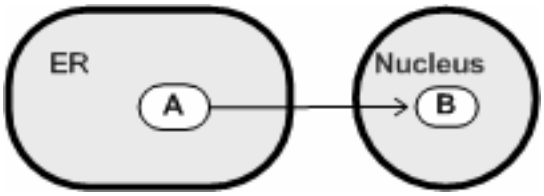


Fig 5: Two compartments in outermost position. Corresponding SBML and PathwayLab representation are shown in Fig. 6.

transformation elements (equivalent to reactions) and control elements (used for connecting modifiers with reaction) as its children. That is, the inner compartments are encompassed by the compartment in which they are present. The inner compartments may in turn have their own children. In PathwayLab the outermost location corresponding to the outermost compartment is defined with a different name called document. Since there is only one document location possible this creates a unique situation in PathwayLab representation when there is more than one compartment in the outermost position as shown in Fig. 5.

```
<sbml .....>
 <model id="ODEExampleModel">
  <listOfCompartments>
   <compartment id="E" Name="ER"/>
   <compartment id="N"
                Name="Nucleus" />
  </listOfCompartments>
  <listOfSpecies>
   <species id="X0" name="A"
                compartment="ER"/>
   <species id="S1" name="B"
                compartment="N"/>
  </listOfSpecies>
       ..............
 </model>
</sbml>
```

SBML

```
<document >
   <name>doc1 </name>
    ..........
   <children>
     <location>
        <name>Nucleus </name>
          ..........
          <children>
            <entity>  <name>A </name>
              ..........
            </entity>
          </children>
     </location>
     <location>
        <name>ER </name>
          ..........
          <children>
            <entity> <name>B </name>
              ..........
            </entity>
          </children>
     </location>
   </children>
</document>
```

PathwayLab

Fig. 6: Representation of the reaction system shown in Fig. 5. In this case the PathwayLab document tags do not represent a compartment .The two compartments 'Nucleus' and 'ER' indicated by dotted lines are outermost in position. (SBML, portion of diagram adapted with minor variations from Finney and Hucka, 2003b)

In such a case the outermost location, i.e. document, has to be ignored as location and all the outermost locations are to be placed within the document. Fig. 6 shows SBML and PathwayLab representations of the system shown in Fig. 5 having two independent compartments. In SBML the two compartments 'ER' and 'Nucleus' are shown as usual. However, in view of the fact that both of these compartments are in the outermost position the 'outside' attribute of both the compartments is not defined. On the other hand, in case of PathwayLab in such a scenario the 'document' tags are not considered as locations and the compartments 'ER' and 'Nucleus' which are within the <document> tags are considered as the outermost locations.

In PathwayLab everything is a child of location (compartment). Hence, for converting the format from SBML to PathwayLab, first of all the locations must be appropriately arranged before any other element can be placed. For arranging the locations in the PathwayLab format we shall know the relative depth of each location so that they can be placed appropriately one within or besides the other. First all the outermost compartments (locations) of SBML were assigned depth zero. Then the compartments which had one of these outermost compartments as the value of their 'outside' attribute were assigned the depth one. This way the process was continued until all the compartments were assigned a depth. For writing the model in PathwayLab format first of all the compartments (locations) with depth zero were placed. In case of a single compartment the <document> tags represented the outermost compartment whereas in case of models with more than one compartment of depth zero the <document> tag was not considered as a compartment. After placing the compartments of depth zero the remaining compartments were placed one by one within their respective parent compartments.

Reactions: In both SBML and PathwayLab a reaction represents any transformation, transport or binding process that can change the amount of one or more species.

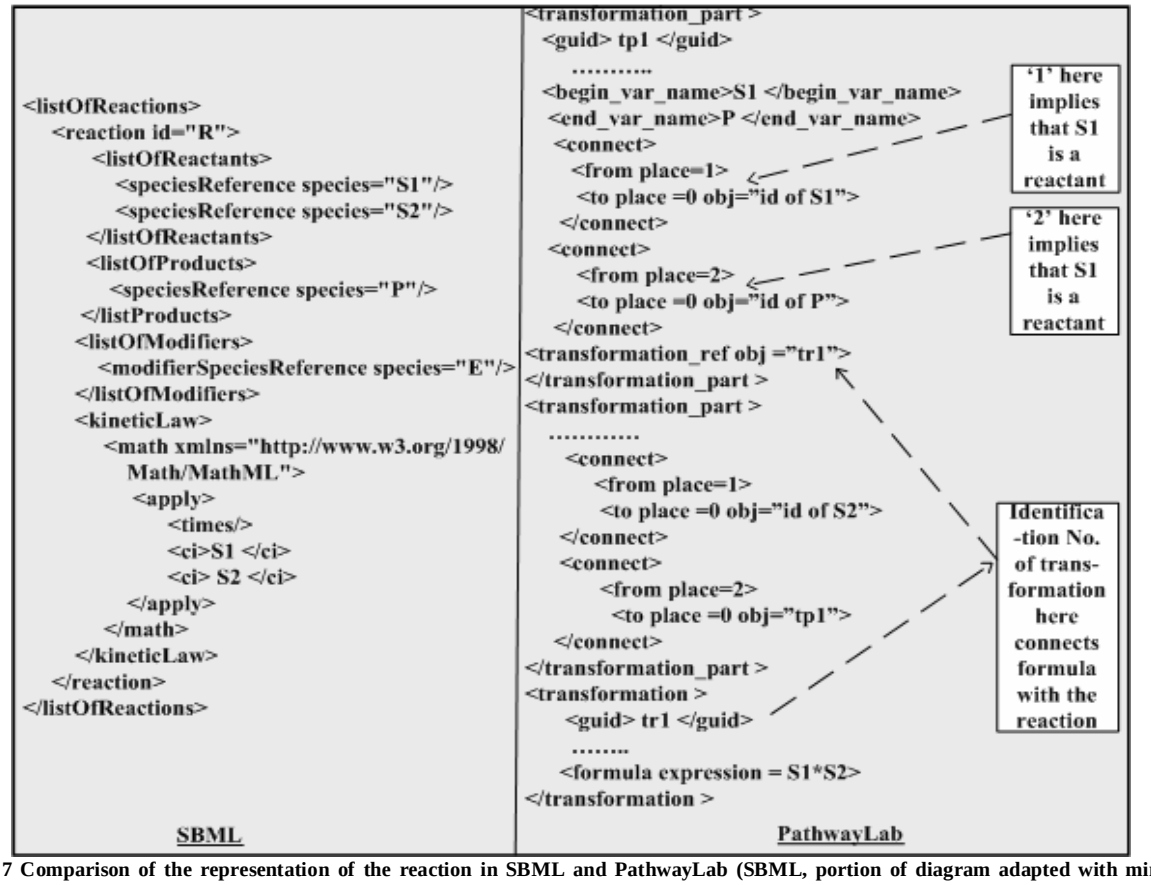


Fig. 7 Comparison of the representation of the reaction in SBML and PathwayLab (SBML, portion of diagram adapted with minor variations from Finney and Hucka, 2003b)

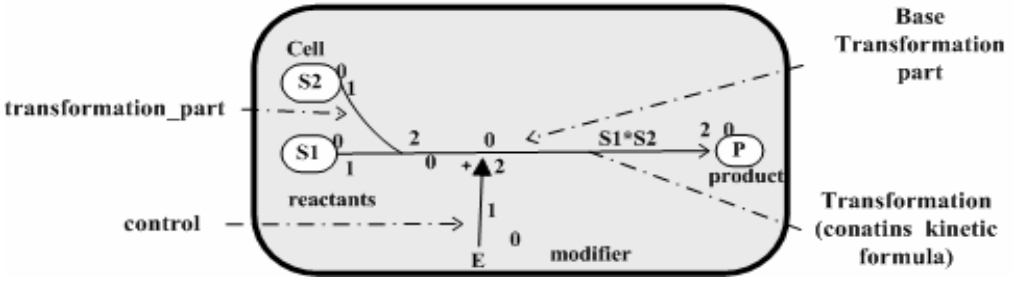


Fig. 8: Elements of reaction in PathwayLab. (Complete details not shown. Refer Appendix 2 and 3 for details).

In SBML all the reactions are grouped together in a list. Each element of this list represents a reaction and in turn contains lists for reactants, products and modifier species present in that reaction. The stoichiometry of reactant and product are mentioned as attributes of the corresponding element. An optional kinetic law describing the rate at which the reaction takes place and optional parameters entering into the kinetic law are also children of the reaction element. The kinetic formulae in SBML (level 2) are represented in terms of MathML (Finney and Hucka, 2003b) (Fig. 7).

In PathwayLab transformations (reactions) present in a particular compartment are children of that compartment. Hence, there is no overall listing of transformations at one place. Since the transformations are represented graphically they have been divided into two types of tags that is, <transformation_part> and <transformation> (Fig. 8). There can be more than one <transformation_part> but only one <transformation> element for each reaction. The number of occurrences of the former depends on the number of reactants and products involved. The <transformation_part> gives information about the status of each entity i.e. whether it is a reactant or a product. An entity present towards the tail side of the transformation_part arrow is categorized as reactant and the one towards the head side of the arrow as product. One of the transformation_parts which does not connect to other transformation parts through its head or tail side is called the 'base transformation part'. The remaining transformation_parts of the reaction join to this base transformation_part. This way all the reactants and products of the reaction are grouped together. Hence there is no need of listing reactants and products of every reaction in PathwayLab format as is done in SBML. The <transformation> tag tells mainly about the kinetic formula of the reaction. The transformation and the base transformation_part are connected together by the <transformation_ref> tag present in each base <transformation_part>. The transformation_ref of the base transformation part contains as its value the identification number of the corresponding <transformation>. This way the kinetic formula gets associated with the base transformation part and hence to all the elements of the reaction which are already connected to the base transformation part.

Modifiers are connected to the reaction through control objects. The head side of the control element joins the transformation_part of the reaction in which the modifier is participating and its tail side joins the modifier species.

Kinetic formula: Another major difference in SBML and PathwayLab representations of reactions is the way they represent the kinetic formulas. In contrast to MathML representation of SBML the formula in PathwayLab is represented as a string as value of the attribute expression within the tag <formula expression=" " unit=" "> which is present within the <transformation>... </transformation> tag of each reaction. The free source library LibSBML was used to convert the MathML format of kinetic formulae in SBML to the string format of kinetic formulae in PathwayLab. However, LibSBML does not have direct binding with C#, which is the language in which the current implementation is being done. Since LibSBML supports C++ it was decided to develop a module in C++ for the conversion of mathematical formulas from MathML to string format. This module could then be called in the C# program wherever needed. Since the module developed in C++ shall be callable from the C# implementation it was decided to use managed C++. Classes developed in managed C++ can be called in the C# program. Hence a DLL called connectLibSBML was developed in managed C++ which uses the LibSBML library for converting MathML to a string (Appendix 7).

This DLL was called in the C# program as shown in Fig. 9.

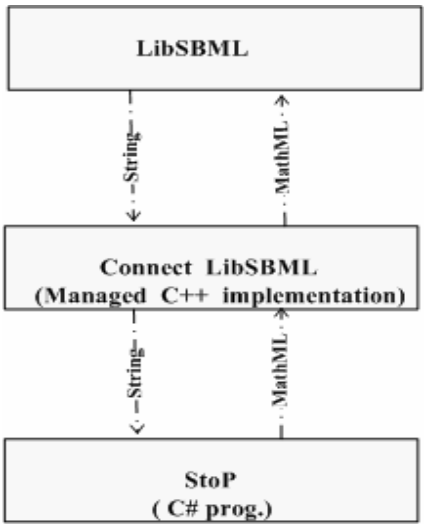


Fig. 9: ConnectLibSBML module in managed C++ connects StoP to LibSBML.

Position of reaction elements in PathwayLab representation: Another important feature of the representation of reactions in PathwayLab is that the location of the transformation_part and transformations depends on the graphical representation of the reaction. The representation for the transformation_part which spans across two or more compartments graphically is located in the compartment which is deepest amongst these compartments. Since the graphical representation of the model will depend on the user they cannot be known at the stage of conversion of the format. Hence, the following method was used for determining the location of control elements, transformations and the transformation_parts. Since this method just relates to variation in position of the elements of reaction in the representation of the reaction they will not affect the simulation of the model.

First of all for determining the location of the transformation and the transformation_part the first reactant and the first product were selected from the reactants and products of the reaction. The base transformation_part was used to connect the selected reactant and product. The depth of the compartment in which the selected reactant and product were present was determined and compared. The transformation and the base transformation_part were placed in the compartment of the reactant or product depending upon which of the two is located in the deeper compartment.

In case of reactions with just reactants or just products the base transformation was placed in the compartment of the selected reactant or the product.

After placing the base transformation, the rest of the reactants and products were connected to it by using one transformation part for each. This transformation_part was placed in the compartment of the reactant or the product which is being connected by it.

The modifiers were similarly connected to the base transformation_part through control elements. The control elements were placed in the compartment of the modifier which is being connected by it.

Parameters: Parameters are used both by SBML and PathwayLab. The parameters can be assigned values and are used in mathematical expressions of the model, as for example in rates of reaction, in both SBML and PathwayLab.

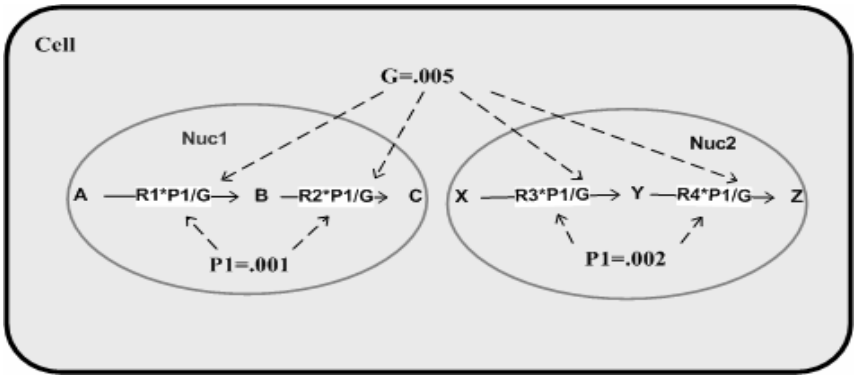


Fig. 10: Parameter declaration in PathwayLab. Parameter 'G' is global to the whole cell, P1 in Nuc1 is global to it whereas P1 in Nuc2 is Global to Nuc2. R1 to R4 are local parameters of the reactions. In SBML it is not possible to define parameters like P1 which are Global to a sub location.

PathwayLab allows the user to declare parameters in each compartment as well as in the reactions as shown in Fig. 10. The parameters declared in a compartment are global to all the compartments and reactions enclosed within it, whereas the parameters declared for a reaction are local to that reaction. Hence, parameters declared in the document compartment will be global to the whole model. Global declaration for inner compartments is useful in a scenario where all the locations within that location or all the reactions within that location have common parameter values. This flexibility in declaration of parameters is very advantageous specifically in large models where there can be a very large number of parameters. Declaration of common values for a subgroup of parameters decreases the chances of inconsistencies in their values. In contrast, in SBML the parameters can be declared as either global to the model as a whole or local for a single reaction (Finney and Hucka, 2003b). This means that SBML does not allow

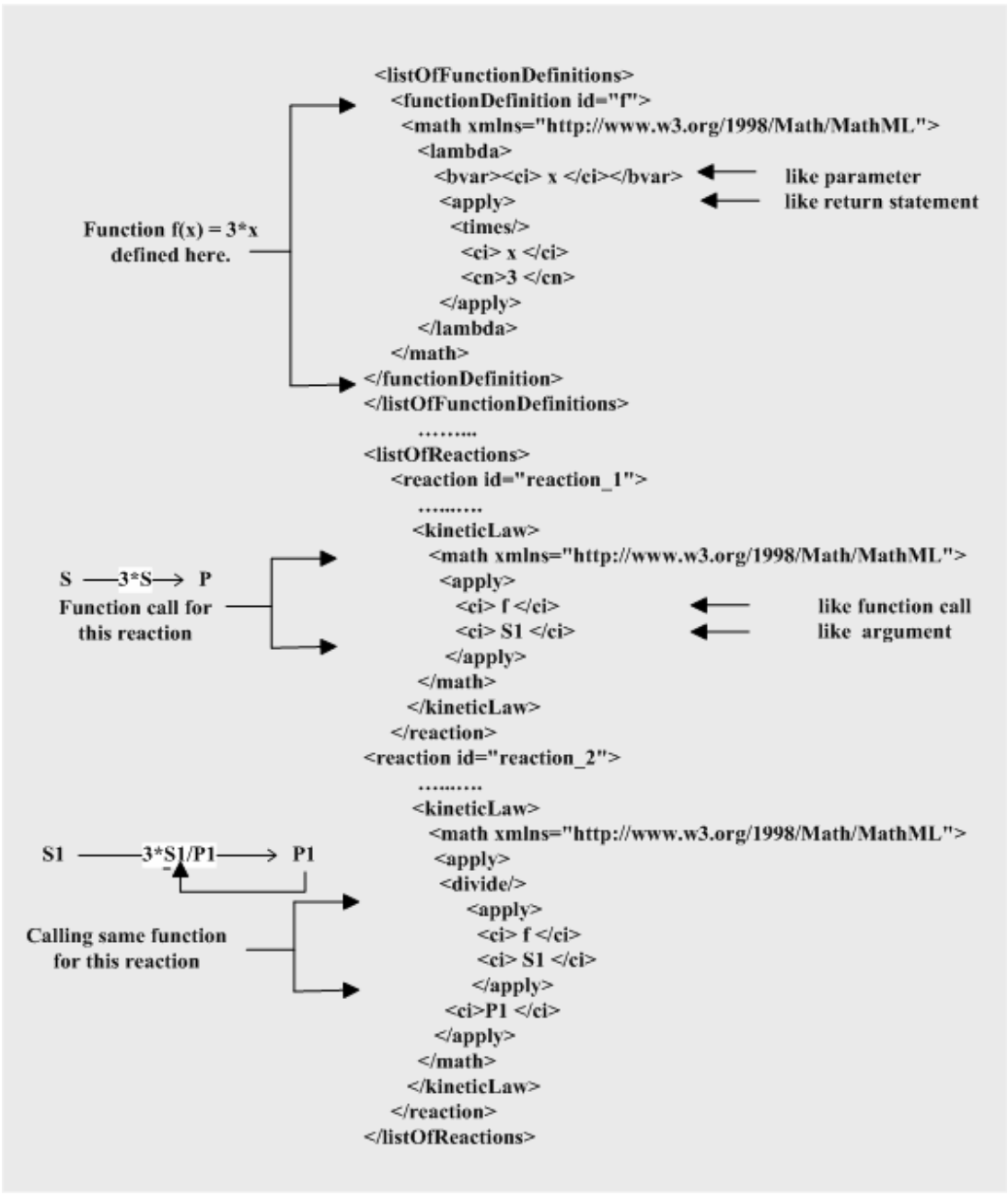


Fig. 11: Function definition and call in SBML (adapted with minor variations from Finney and Hucka, 2003b).

declaration of parameters which are global to an individual compartment only.

Function: SBML provides facilities for defining mathematical functions that may be used throughout the model. The function is declared in a special lambda element in MathML (Finney and Hucka, 2003b). The usage of functions enables SBML to save code for repetitive elements found in MathML as can be seen from Fig. 11 where a common function is defined in the lambda element of the function definition tag. This function is then being called by MathML in the kinetic equation of two reactions.

PathwayLab does not have facilities to handle functions in its representation and hence this feature has not been implemented. For implementing this aspect of SBML two alternatives were identified. One is to enhance PathwayLab so that the user is able to define functions. Since PathwayLab format has a default <document> compartment which is present in all the models and is parent to all the elements of the model it can be possible to define functions here. Just like the parameters defined in the document compartment become global to the whole model a function defined here will be accessible from all the places in the model. The second workable, but less efficient, method is to enhance the present tool StoP such that while converting the format from SBML to PathwayLab it shall store all the functions of the SBML model and make them inline at each instance of their call.

Events: Events represent instantaneous, discontinuous changes in variables of a biological system when a triggering condition is satisfied. The event tags in SBML define when the event can occur, the variables that are affected by the event, and how the variables are affected. The triggering condition in SBML representation is defined in the trigger element of the event. After satisfaction of the triggering condition the SBML events can instantaneously modify species concentration, compartment size or parameter. In Fig. 12 for example, 'Gene 2' gets activated the moment the concentration of 'P', which is the product of 'Gene 1', crosses a predefined threshold concentration set at 't'. The effect of the event can also be delayed after the occurrence of the condition which invokes it (Finney and Hucka, 2003b).

PathwayLab currently does not support events. Hence this feature of conversion from SBML to PathwayLab has not been implemented.

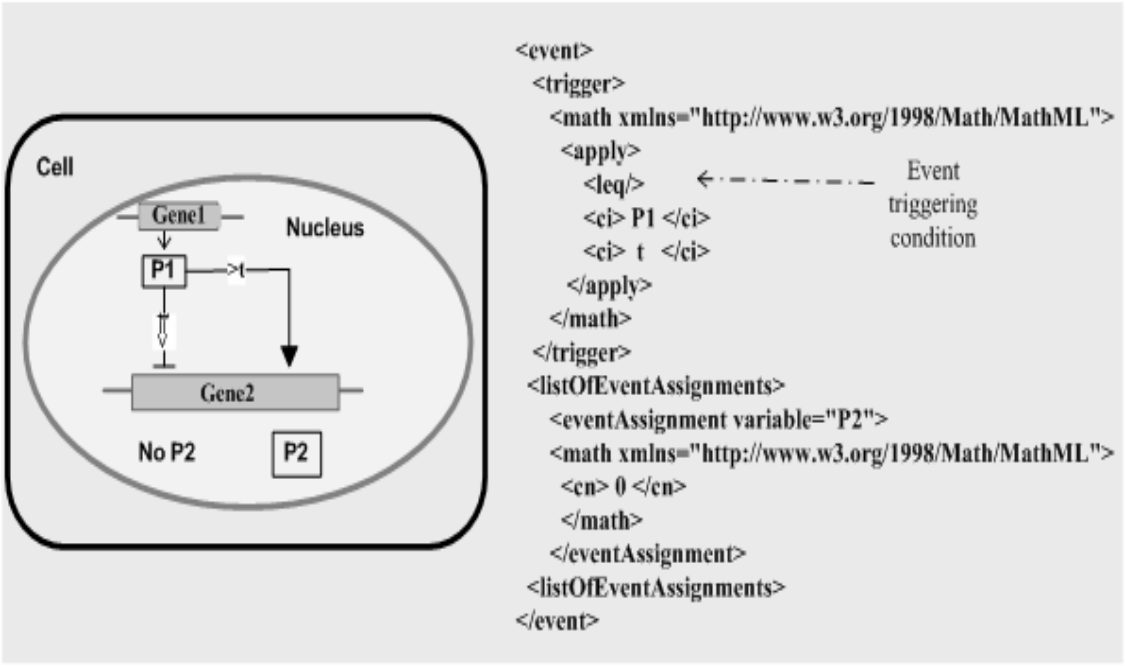


Fig. 12: Events. Diagrammatic representation (left) of an event in which the triggering condition (concentration of P1 becomes less than 't') is reached.Gene2 stops producing P2. The same event is shown in SBML representation (right). (SBML, portion of diagram adapted with minor variations from Finney and Hucka, 2003b).

Rules: The values of various variables and parameters of an SBML model can be constrained by using rules. Algebraic rules, assignment rules and rate rules are the three different kinds of rules defined in SBML. All the three rules of SBML are described within the tags <listOfRules>…</listOfRules> (Finney and Hucka, 2003b).

```
<listOfRules>
   ……...
 <algebraicRule>
   <math xmlns="http://www.w3.org/1998/
                          Math/MathML">
         <apply>
            <minus/>
             <apply>
               <times/>
               <ci> X2 </ci>
               <ci> X1 </ci>
             </apply>
             <apply>
               <times/>
               <ci> X3 </ci>
               <ci> 3 </ci>
             </apply>
         </apply>
   </math>
 </algebraicRule>
  ……..
</listOfRules>
```

Fig. 13: SBML representation of algebraic rule X1*X2-X3*3=0 (adapted with minor variations from Finney and Hucka, 2003b).

Algebraic rules: These rules are of the form

$$0 = f(W)$$

where 'W' is a vector of variables. Hence these rules constrain the values of variables to the expression consisting of various variables and constants (Finney and Hucka, 2003b). For example, there may be a set of variables X1, X2 and X3 related in such a way that the expression (X1*X2)-(X3*3) = 0 is always true. Such a constraint on the values of the variables X1, X2 and X3 can be expressed in SBML by way of an algebraic rule as shown in Fig. 13. PathwayLab does not support algebraic rules at present. Additional functionality will have to be developed in PathwayLab to handle such a situation where it is required to constrain a set of variables to an expression such that the expression is always true.

AssignmentRules:

These rules are of the form

$$X = f(V)$$

where 'X' refers to a species, parameter or compartment and 'V' is a vector of variables that does not include X.

Thus assignment rules can be used to change the concentration of a species, value of a parameter or volume of a compartment to the value of the assigned function. The value assigned initially to these variables gets overridden. The rules are evaluated in the order they appear in the model and hence their ordering is important (Finney and Hucka, 2003b). For example, a variable 'X' can be assigned a value equal to a function evaluating to (X1*X2)-(X3*3) where X1, X2 and X3 are three other variables as shown in Fig. 14.

```
<listOfRules>
   ……...
 <assignmentRule variable="X">
   <math xmlns="http://www.w3.org/1998/
                          Math/MathML">
         <apply>
            <minus/>
              <apply>
                 <times/>
                   <ci> X2 </ci>
                   <ci> X1 </ci>
              </apply>
              <apply>
                 <times/>
                   <ci> X3 </ci>
                   <ci> 3 </ci>
              </apply>
         </apply>
   </math>
 </algebraicRule>
  ……..
</listOfRules>
```

Fig 14: An assignment rule where the variable 'X' is assigned a value equal to the mathematical expression in the math element (adapted with minor variations from Finney and Hucka, 2003b).

In PathwayLab, there is no specific representation for assignment rules. However, by declaring an entity as auxiliary we can have the effect of an assignment rule, since now it can act

as a variable which can be assigned an expression of parameters and/or other variables (e.g. species). The other entities in such a case are to be connected to this auxiliary entity by way of control arrows. However, this way we can only assign values to the variables created by auxiliary entities. We cannot assign values to the volume of a compartment or species concentration and parameters in the way it is possible in SBML.

RateRules are used to define equations which determine the rates of changes of variables. These are of the form

$$dX/dt = f(W)$$

where 'X' is a variable which can refer to a species, parameter or compartment and 'W' is a vector of variables which may include X.

Thus rate rules can be used to set the rate of change of concentration of a species, volume of a compartment and the value of a parameter to the value determined by the rate rule (Finney and Hucka, 2003b). PathwayLab does not support rate rules at present.

Since PathwayLab does not fully support rules they have not been implemented.

Results and Discussion

The differences in formats explained above indicate that although some of the features of SBML models can be converted to PathwayLab format, others cannot be converted. The features which cannot be converted by the tool StoP are those which are not supported by PathwayLab at present. The tool StoP, when integrated with PathwayLab, will enable the users to import the models developed in over sixty other SBML supporting commercial tools. This facility would obviate the need to reenter the models in PathwayLab which have been made in other SBML supporting tools. However only those models can be imported which have features currently supported by PathwayLab.

Features which have been implemented: The tool StoP is capable of converting the format of various components of biochemical pathways including multiple compartments, global parameters, reactants, products, modifiers, reactions, kinetic formulas and reaction parameters from the SBML representation to the PathwayLab representation. It converts the MathML representation of the kinetic formula in the SBML model to the string representation of the formula in PathwayLab format. Besides taking care of differences in terminology and placements of various elements between the two formats it also takes care of features peculiar to PathwayLab, such as compartments and representations of reactions. The tool StoP when integrated with PathwayLab will enable it to import and simulate SBML models having above mentioned features which have been implemented.

Features not supported by PathwayLab which can be implemented without modifying it: The SBML functions are at present ignored by the tool since they are not supported by PathwayLab. When a model containing a function is encountered a suitable warning is displayed to the user indicating that the functions are not supported by PathwayLab and have been ignored. Functions encountered in an SBML model during conversion from SBML to PathwayLab can be taken care of by making them inline. This would require modification of the tool StoP. Another option is to modify PathwayLab so as to support functions.

Features not supported by PathwayLab which cannot be implemented without modifying it: Some of the features in the SBML model definition including unit definitions, rules and events cannot be converted into PathwayLab format at present since they are not supported by PathwayLab. Suitable modifications will have to be made in PathwayLab before they can be implemented. At present when these features are encountered in an SBML model the tool StoP gives warnings to the user that they are being ignored during conversion to PathwayLab format.

UnitDefinitions: SBML provides a facility to define user-defined units as well as conversion of units. At present the users have to take care of the units of various quantities as PathwayLab does not automatically convert units. Development of support for unitdefinitions will make PathwayLab more user-friendly.

Events: Events can not be implemented in the current version of PathwayLab. Suitable additions and/or modifications are required in PathwayLab to take care of events. The development of support for events will enable PathwayLab to handle models of biological

systems which involve instantaneous, discontinuous changes in variables.

Rules: Rules cannot be implemented in the present version of PathwayLab. Only a minor portion of assignment rules can be implemented by declaring an entity as auxiliary since now it can act as a variable which can be assigned an expression of parameters and/or other variables. However even in this case it is not possible at present to assign values to volume of compartment or species concentration and parameters the way it is possible in SBML. Support for rules should be developed in PathwayLab since they can be used in models where we want to constrain various variables and parameters of the model according to predefined rules.

Features supported by PathwayLab and not supported by SBML: The ability of PathwayLab to declare local parameters for locations which can be accessed by those locations as well as their children makes parameter declaration more flexible and less prone to inconsistencies in parameter values. As the parameter declaration in PathwayLab is more flexible than in SBML the conversion from SBML to PathwayLab is not a problem and has been implemented. SBML can also include parameters at the level of compartments by way of including a ListofParameters element for each compartment just as they are declared for the reactions. The definition of parameters at compartment level will make SBML more flexible in this respect since it will obviate the need to make multiple entries of parameters common to the various children of that compartment.

Conclusions

Different standards are being developed for representation of data relating to different domains of biology. Even when we confine ourselves to the domain of biochemical pathways we may have different possibilities of representations as exemplified by extremely different representations of SBML and PathwayLab. The differences in representations may include the differences in overall capabilities of the modelling softwares as exemplified by the absence of SBML's rules, events and unit definitions in PathwayLab. Even features which are common may also be represented differently. These differences may be as simple as differing terminologies for the same things. The basic architecture of representation may also be different as shown by the difference between the concise and component-oriented representation of SBML and a more verbose and location-oriented PathwayLab representation. Differences may also reflect differences in visualisation of the same things. The difference in representation of reactions between PathwayLab and SBML falls under this category. In SBML all the reactions are grouped together in a list, where each item in turn contains lists for reactants, products and modifier species present in that reaction. Hence the reactions here are visualised as listings of their components. However, in the case of PathwayLab the representation of reactions follows its graphical structure. Other differences in representation of pathway models which have been observed here include adoption of different standards for representing similar concepts. This is clearly evident from the difference in string representation of kinetic formulas in PathwayLab and the MathML representation of kinetic formulas in SBML.

PathwayLab should develop support for unitdefinitions, events and rules. Development of support for functions in PathwayLab is not essential but it will simplify model construction since functions will obviate the need of repetitive entry of similar formulae. Similarly definition of parameters at compartment level in SBML is not essential but it will obviate the need to make multiple entries of parameters common to the various children of that compartment.

In spite of the differences the overall similarity in the domain of pathway models makes it possible to convert the representation of common features from one format to the other as exemplified by conversion of SBML to PathwayLab format here. Tools like StoP may be developed for these conversions and they will be required for the existing modelling tools like PathwayLab for making them SBML compatible. However it will be advantageous for newly emerging tools to follow SBML as a standard for the representation of pathway models, in view of the fact that SBML support has already been developed by over sixty other tools, and as a result it has become a popularly supported standard.

References

- Achard F., Vaysseix G, Barillot E., (2001), XML Bioinformatics and Data Integration, *Bioinformatics*, 17(2):115-125.
- Blyden E., Dai D., Gordon D., Guo C., Seth K., Rentschler E., Roggenkamp S., Rumpf R, and Spitzner J., (2002), Bioinformatic Sequence Markup Language , *BSML Reference Manual* , Available from http://www.bsml.org/resources/ default.asp[Accessed 10 Dec., 2004].
- Bruhn R. and Burton P., (2003), Designing XML Schemas for Bioinformatics, *Journal of BioTechniques*, 34(6):1200 – 1211.
- Cinquin O. & Demongeot J., (2002), Roles of positive and negative feedback in biological systems, *Comptes Rendus Biologies* 325(11): 1085-1095.
- Crampin E. J. and Schnell S., (2004), New approaches to modelling and analysis of biochemical reactions, pathways and networks, *Progress in Biophysics & Molecular Biology*, 86(1):1–4.
- Cuellar, A.A., Nielsen, P.F., Bullivant, D.P., Nickerson, D., Hedley, W., Nelson, M., Lloyd, C.M., (2003), CellML Specification 1.1. Draft-30 September 2003 [online], Available at http://www.cellml.org/public/specification/cell ml_specification.html [Accessed 27 July, 2004].
- Deville Y., Gilbert D., Helden J.V., Wodak S.J., (2003), An overview of data models for the analysis of biochemical pathways. Brief *Bioinformatics* 4(3): 246-259.
- Elowitz, M. B., Levine, A. J., Siggia, E. D., Swain, P. S., (2002), Stochastic Gene Expression in a Single Cell. *Science* 297(5584): 1183-1186.
- Fenyö D., (1999), The Biopolymer Markup Language, *Bioinformatics*, 15(4):339-340.
- Fiering S., Whitelaw E., and Martin. D.I., (2000), To be or not to be active: The stochastic nature of enhancer action *Bioessays* 22:381-387.
- Finney A. and Hucka M., (2003 a), Systems biology markup language: Level 2 and beyond, *Biochemical Society Transactions* 31(6):1472.
- Finney A. and Hucka M., (2003 b), Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions [online], Available from http://sbml.org/ documents/ [Accessed 27 July, 2004].
- Gilmour R., (2000), Taxonomic Markup Language: applying XML to systematic data, *Bioinformatics*, 16(4):406-407.
- Gillespie D. T., (1977), Exact stochastic simulation of coupled chemical reactions, *Journal of Physical Chemistry*, 81(25):2340–2361.
- Greenwald I., (1998), LIN-12/Notch Signalling: Lessons from Worms and Flies. *Genes Dev*, 12 1751-1762
- Hammer G.L, Sinclair T. R., Chapman S. C. and Oosterom E.V., (2004), Scientific correspondence on systems thinking, systems biology and the in silico plant, *Plant Physiology*, 134: 909–911.
- Hanisch D., Zimmer R and Lengauer T., (2002), ProML - the Protein Markup Language for specification of protein sequences, structures and families, *In Silico Biology* 2(3):313-24.
- Helfert S., Estevez A.M., Bakker B., Michels P. and Clayton C., (2001),Roles of triosephosphate isomerase and aerobic metabolism in *Trypanosoma brucei*, *Biochem J*, 357:117-25.
- Hoefnagel M.H.N., Starrenburg M.J.C., Martens D.E., Hugenholtz J., Kleerebezem M., Van Swam I.I., Bongers R., Westerhoff H.V. and Snoep J.L., (2002), Metabolic engineering of lactic acid bacteria, the combined approach: Kinetic modelling, metabolic control and experimental analysis, *Microbiology*. 148: 1003-1013.
- Hunter, P.J., Borg, T.K., (2003), Integration from proteins to organs: the Physiome Project. *Nature Reviews Molecular Cell Biology* 4(3):237–243.
- Jirstrand M., Gunnarsson J., Johansson H., (2004), PathwayLab - A software for modeling and simulation of biochemical reaction networks, Reglermöte 2004, Signaling and system, Chalmers Technical University, Göteborg.
- Katagiri F. (2003), Perspectives on Systems Biology: Attacking Complex Problems with the Power of Systems Biology, *Plant Physiol*. 132:419.
- Kitano H., (2001), Foundations of Systems Biology, *MIT Press*, Cambridge, MA.
- Klamt S. and Gilles E.D., (2004), Minimal cut sets in biochemical reaction networks, *Bioinformatics* (2004), 20(2):226–234.
- Lambeth M.J. and Kushmeric, M.J., (2002), A computational model for glycogenolysis in skeletal muscle, *Annals of Biomedical Engineering* 30:808–827.
- Lloyd C.M., Halstead M.D.B., Nielsen P.F., (2004), CellML: it's future, present and past. *Progress in Biophysics & Molecular Biology*, 85:433–450.
- Martin T.A., (1999*), Project cool guide to XML for web designers*, 29-48.
- Mestl T., Pholte E. and Omholt S.W., (1995), A mathematical framework for describing and analyzing gene regulatory networks. *Journal of Theoretical Biology*, 152:429-453.

- Mjolsness E. and Gibson M.A., (2001), In: Bower and Bolouri editors. *Computational Modeling of Genetic and Biochemical Networks*, 15-40.
- Noble D., (2002), Modelling the heart - from genes to cells to the whole organ, *Science*, 295(5560).
- Noble D. and Rudy Y., (2001), Models of cardiac ventricular action potentials: iterative interaction between experiment and simulation, *Philosophical Transactions: Mathematical, Physical and Engineering Sciences, 359*:1127-1142.
- Reichardt T., (1999), It is a sink or swim as a tidal wave of data approaches. *Nature* 399:517-520.
- Robbins R.J., (1994), Report of invitational DOE workshop on genome bioinformatics, databases. *Journal of computational Biology* 1:173-190.
- Schilling C. H., Schuster S., Palsson B.O, Heinrich R., (1999), Metabolic Pathway Analysis: Basic Concepts and Scientific Applications in the Post-genomic Era, *Biotechnology Progress*, 15:296-303.
- Spellman P.T. , Miller M., Stewart J., Troup C., Sarkans U., Chervitz S., Bernhart D., Sherlock G., Bal C., Lepage M., Swiatek M., Marks W. L., Goncalves J., Markel S., Iordan D, Shojatalab M, Pizarro A., White J, Hubley R, Deutsch E., Senger M., AronowB J., Robinson A., Bassett D., Stoeckert C J. and Brazma A., (2002), Design and implementation of microarray gene expression markup language (MAGE-ML) , *Genome Biology*, 3(9):001-0046.
- Teusink B. Passarge J. Reijenga C.A. Esgalhado E., van der Weijden C.C., Schepper M., Walsh M.C., Bakker B.M., van Dam K., Westerhoff H.V. and Snoep J., (2000), Can yeast glycolysis be understood in terms of in vitro kinetics of the constituent enzymes? Testing biochemistry, *Eur. J. Biochem*. 267, 5313-5329.
- Thagard P., (2003), Pathways to Biomedical Discovery, *Philosophy of Science*, 70 (April): 235–254.
- Thomas, R. and D'Ari, R., (1990), *Biological Feedback*. CRC Press, Boca Raton, Fla. 37
- Thomas R., Thieffry D. and Kaufman M., (1995), Dynamic behavior of Biological regulatory networks-I, Biological role of feedback loops and practical use of the concept of the loop-characteristic state, Bull *Mathematical Biology*, 57(2): 257-276.
- Waugh A, Gendron P, Altman R, Brown JW, Case D, Gautheret D, Harvey SC, Leontis N, Westbrook J, Westhof E, Zuker M, Major F., (2002), RNAML: a standard syntax for exchanging RNA information, *RNA*. Jun; 8(6):707-17.

Appendix 1: Differences between five SBML Models which had online simulation facility

Differences between five SBML Models which had online simulation facility						
SBML Level 2 (Version 1)		***T. Brucei***	***S.Cerevisae***	***Skeletal Muscle***	***L. lacti***	***L. lacti***
		1	2	3	4	5
Comp-onent	**Field**	***T. Brucei* model with minor changes**	***S.Cerevisae* model with minor changes**	**Skeletal Muscle model with minor changes**	**Pyruvate Branches**	**Pyruvate Branches Full Glycolysis**
Function		N	N	N	N	N
Unit Defi-nition		N	N	N	N	N
Compartment						
	id	P	P	P	P	P
	name	A	A	A	A	A
	units	A	A	A	A	A
	outside	**Ab**	**Ab**	**Ab**	**Ab**	**Ab**
	spatial dimension	D	D	D	D	D
	size	P	P	P	P	P
	constant	D	D	D	D	D
Spec-ies						
	id	P	P	P	P	P
	name	A	A	A	A	A
	compartment	P	P	P	P	P
	initial amount	M	M	M	M	M
	boundary condition	D	D	D	D	D
	charge	N	N	N	N	N
	initial concentration	M	M	M	M	M
	substanceunits	N	N	N	N	N
	SpatialSizeUnits	N	N	N	N	N
	HasOnlySustanceUnits	N	N	N	N	N
	constant	D	D	D	D	D
Para-meters		R	R	R	R	R
	Name	P	P	P	P	P
	Value	P	P	P	P	P
	Units	**Ab**	**Ab**	**Ab**	**Ab**	**Ab**

Rules							
	Assign-ment						
		MathML	P	**Ab**	**Ab**	**Ab**	**Ab**
		Variable	P	**Ab**	**Ab**	**Ab**	**Ab**
	Rate		N	N	N	N	N
		MathML					
		Variable					
	Algeb-raic		N	N	N	N	N
		MathML					
Reac-tions							
		id	P	P	P	P	P
		name	A	A	A	A	A
		reversible	D	D	D	D	D
		fast	N	N	N	N	N
	Reac-tant						
		species	P	P	P	P	P
		stoichiometry	P	P	P	P	P
		stoichiometrymath	N	N	N	N	N
	Prod-uct						
		species	P	P	P	P	P
		stoichiometry	P	P	P	P	P
		stoichiometrymath	N	N	N	N	N
	Modi-fier						
		species	**Ab**	**Ab**	P	**Ab**	P
	Kinetic law						
		math	P	P	P	P	P
		parameter	P	P	P	P	P
		timeunits	N	N	N	N	N
		substanceunits	N	N	N	N	N
Event			N	N	N	N	N

Leg-end :

p	**Present**	N	**Not Supported by PathwayLab**	A	**Add**
Ab	**Absent**	D	**Default value present**	M	**Mutually exclusive with each other**
R	**Some local parameter of reaction can be used globally**				

Appendix 2: Description of Mapping between SBML and PathwayLab

SBML Level 2 (Version 1)		SBML Level 2 (Version 1)	PathwayLab
SBML Tags	**Attributes**	**What is it**	
model			
name			
listOfFunctionDefinitions			
Function		Function definition.	Function not defined here.
listOfUnitDefinitions			
Unit Definition		Present	No such definition. In fact PathwayLab does not take into account units.
listOfCompartments			
Compartment		Every species in an SBML model must be located in a compartment. Therefore for defining a species a compartment must be defined. Id of the compartment is mentioned as a property of the species to mention the location of species i.e. species are not children of compartment. Similarly a compartment within a compartment is mentioned by indicating the id of outer compartment in the outside field of inner compartment	Corresponds to <location>… …….</location> tags. Here location is at the top of hierarchy of parent children relation. Entities (Species), reactions (transforma-tions and control) and inner locations are children of a location. i.e. inner location is encompassed by the outer location. The inner location in turn has its own children. The outermost location corresponding to outer most compartment is however defined with <document>.....</document> tags.
	id	Uniquely identifies a compartment.	Content between <guid>..</guid> tags in location correspond to this and uniquely identifies it.
	name	Mentions name of location	Content between <name>.. </name> tags in location correspond to this.
	units	Mentions appropriate unit according to dimensions of the compartment.	Unit mentioned as attribute of formula tags present in signal tags of location. However units are not taken into consideration while evaluation.
	outside	Mentions the Id of the compartment present outside to the current compartment.	Neither present nor needed since inner compartment (location) is children of outer. So the hierarchy decides the location of the compartment.
	spatial dimension	Mentions number of dimensions 0, 1, 2 or 3. Default is 3.	Compartment is considered 3 Dimensional only.
	size	Indicates size of compartment.	Corresponds to the value of the attribute expression of <formula> tag which is present in <signal> tags of location.
	constant	Default True i.e. size not variable since in the most common modelling scenarios present compartment sizes remain constant.	PathwayLab supports time-varying volumes, but it can not be changed as in SBML with their event mechanism.

listOfSpecies		Substance or entity that takes part in a reaction. Here all the species of the model are listed. Their id is used in reactions at appropriate place (Reactant, Product or Modifier) wherever the species is required.	Entity is equivalent to species here. No such overall listing here. Entities are children of location in which they are present. Hence entities of a particular location are listed in the children block of that location.
Species		Substance or entity that takes part in a reaction.	Corresponds to entity here.
	id	Uniquely identifies a species.	Content between <guid>..</guid> tags in entity correspond to this. In case of duplicates unique id is unique for each duplicate.
	name	Mentions name of species.	Content between < name>.. </name> tags in entity correspond to this.
	Compart-ment	No default value. Is used to identify the compartment in which the species is located.	Not required since entities are children of location in which they are located.
	initial-amount	Used to set the initial amount of species in the compartment.	In PathwayLab initial quantities of species are given in concentration-units.
	boundary condition	If true indicates that a given species' concentration is not determined by the set of reactions even when that species occurs as a product or reactant.	Can declare the entity as Auxiliary / signal by assigning the value true to 'use' attribute of signal tag within concentration tag of entity. <Concentration value =" " min=" " max=" " use_min =" " use_max=" "> <signal use=true> <formula expression=" " unit=" "/> </signal> </concentration>
	charge	An integer indicating the charge on the species	No correspondence. Charges not taken into account.
	initial-concen-tration	Mutually exclusive to initialamount .It must not have a value if the species' compartment has a spatialDimensions value of "0" or if the value of the species' hasOnlySubstanceUnits field is "true".	Corresponds to value attribute in the concentration tag present within entity tags. <Concentration value=" " min=" " max=" " use_min=" " use_max=" "> </concentration>.
	substance units	Relate to Units under different conditions.	PathwayLab does not evaluate units hence no correspondence.
	Spatial SizeUnits	-do-	-do-
	HasOnly Substan-ce Units	-do-	-do-
	constant	Indicates whether the concentration of species can vary during simulation. Default is false	As in boundary condition above. PathwayLab does not make distinction between boundary condition and constant.
listOfParameters		Parameters listed here are global to the whole model.	Here parameters are listed within <parameter> tags present within <formula> tag within each location as well as reaction. Each parameter here is represented by a separate parameter tag having the attributes 'local', 'name', 'value' and 'unit'. Those listed in the outermost location (i.e. "document") are global to whole model.

Parameters			
	id	A unique identifier for parameter.	No correspondence since not required. Here parameters can be declared in each location. Name itself is unique for that location.
	name	Mentions name of parameter.	Within <parameter> tag a parameter is assigned name as a value of 'name' attribute. The parameter tag is present in the <formula> tag which in turn is present in <signal tag>. Name is unique to the location in which it is present. In case of overlap with the name of outer location the name which is local to the location prevails.
	value	Indicates value assigned to the parameter.	Corresponds to attribute value in the <parameter name = " " value=" " unit=" "> tag.
	units	.Determines units associated with the value of the parameter	Corresponds to unit in the <parameter name=" " value=" " unit=" "> tag. However no unit conversions are performed here.
	constant	Indicates whether the parameter's value can vary during a simulation. Parameters local to a reaction cannot be changed and are implicitly always constant. Hence they should not have their constant field set. Default value is true.	Parameters do not vary during simulation.
listOfRules			
Rules		Three types of rules i.e. rate rule, algebraic rule and assignment rules can be defined.	No specific representation for Rules. However by declaring a species as auxiliary we can have the effect of assignment rule since now it can act as a variable which can be assigned an expression of parameters and/or other variables (e.g. species).
listOfReactions		All the reactions of the model are listed within it.	Reactions called transformation present in a particular location are listed within that location. Therefore no overall listing of reactions for the model is done at one place.
Reactions		Corresponds to each chemical reaction.	Correspond to transformation. Since the transformations are represented graphically therefore it has been divided into two types of tags i.e. <transformation_part> and <transformation>.These two are present in the children block of each location. There can be more than one <transformation_part> but only one <transformation> for each reaction. The number of former depends on the number of entities involved. The <transformation_part> gives information about the status of each entity i.e. whether it is reactant or product. The <transformation> tells mainly about the kinetic formula of the reaction. The two are connected together by the <transformation_ref obj> tag

			present in each <transformation_part> of reaction which refers to the guid of the corresponding <transformation>.
	id	Uniquely identifies a reaction.	In view of information in above paragraph there is no single id for a reaction.
	name	Mentions name of reaction.	Corresponds to the content within tag <name>.</name> present both in <transformation> and <transformation_part> tags.
	reversible	Indicates whether the reaction is reversible or not. Default is true.	No separate tag for reversibility here. It is determined by the formula specified in rate law only.
	fast	True indicates reaction is fast. No default value. Missing value indicates the modeller does not know.	No corresponding match.
Reactant			No separate listing of reactants is given. An entity present towards the tail side of the transformation arrow is categorized as reactant and the one towards the head side of arrow as product. The head side of transformation arrow is given value "2" and the tail side is given value "1".
	species	Refers to id of species in species list.	Corresponds to guid of the reactant. In the connect element of the transformation part within the tag <to place="0" obj={ }/> the value within the braces of attribute "obj" corresponds to guid of reactant if this tag is preceded by the tag <from place="1"/>
	stoichio-metry	Indicates the stoichiometry of reactant. Should contain values greater than 0.Default value is 1.	Corresponds to the value within the tags <begin_stoich_coeff>.. </begin_stoich_coeff> in corresponding <transformation_part>
	stoichio-metrymath	Is implemented as an element containing a MathML.	No correspondence since stoichiometry not mentioned as formula.
Product			As against Reactant above.
	species	Same as Reactant	Corresponds to guid of the Product. In the connect element of the transformation part within the tag <to place="0" obj={ }/> the value within the braces of attribute "obj" corresponds to guid of reactant if this tag is preceded by the tag <from place="2"/>.
	stoichio-metry	Same as Reactant	Corresponds to the value within the tags <end_stoich_coeff> ... </end_stoich_coeff> in corresponding <transformation_part>.
	stoichio-metrymath	Same as Reactant	No correspondence since stoichiometry not mentioned as formula.
Modifier		Indicates catalyst or inhibitor. Neither created nor destroyed.	Modifiers are connected to the reaction through control objects.
	species	Refers to id of species in species list.	Corresponds to guid of the Modifier. In the connect element of the <control> tags is present a tag <from place="1">. It is followed by the tag <to place="0" obj={ }/>.The value in the braces of field "obj"

			referred above corresponds to guid of modifier.
Kineticlaw		Describes the rate at which the reaction takes place.	
	math	MathML expression for rate of the reaction.	Expressed as a string as value of the attribute expression within the tag <formula expression=" " unit=" "> which is present within the <transformation>...</transformation> tag of each reaction.
	parameter	Local in scope. Takes precedence over that global parameter. The parameters in the kinetic law are declared within the <Parameters> tags as the global parameters described above. Here also they are all listed together within the tags <listOfParameters>...<listOfParameters>.	Parameters local to a reaction are present in the <parameter> tag present in the <formula> tag which in turn is present in the tag <transformation>.Each parameter is described by separate parameter tag.
	substance-units and timeunits	SubstanceUnits and timeUnits determine the units of substance and time for the reaction respectively	PathwayLab does not take into consideration the units during calculation of reaction kinetics.
listOfEvents			
Event		Describe the time and form of explicit instantaneous discontinuous state changes in the model.	PathwayLab does not handle events.

Appendix 3: Mapping between SBML and PathwayLab

Basic formats: -

SBML Basic Format	PatwayLab Basic Format
<sbml> <model id="My_Model"> <listOfFunctionDefinitions></listOfFunctionDefintions> <listOfUnitDefinitions></listOfUnitDefinitions> <listOfCompartments></listOfCompartments> <listOfSpecies></listOfSpecies> <listOfParameters></listOfParameters> <listOfRules></listOfRules> <listOfReactions></listOfReactions> <listOfEvents></listOfEvents> </model> </sbml>	<document> ... miscellaneous properties ... <children> <entity>...</entity> <transformation>..</transformation> <control>...</control> <location> <children>...</children> </location> </children> </document>

Details : -

SBML	**PatwayLab**
<sbml> <model id="My_Model">	
<listOfFunctionDefinitions>	Functions not defined in PathwayLab
</listOfFunctionDefintions>	
<listOfUnitDefinitions>	Units not taken into consideration in calculations in PathwayLab.
</listOfUnitDefinitions>	
<listOfCompartments>	No such separate list of compartments is given. Outermost compartment is <document version=" ">… </document> Inner compartments are <location>.....</location>
<compartment	
id =" "	Content within **<guid>..</guid>** with in location.
name =" "	Content within **< name>..</name>** within in location.
spatialDimensions =" "	No corresponding field.
size =" "	Value of the attribute expression of the tag formula shown below. <signal> <formula **expression=" "** unit=" "> <parameter local="true/false" name=" " value=" "/> .. .. <parameter local="true/false" name=" " value=" "/> </formula expression> </signal>
units =" "	Value of the attribute unit of the tag formula shown above
outside =" "	Neither present nor needed since inner compartment (location) is children of outer. So the hierarchy decides the location of the compartment.

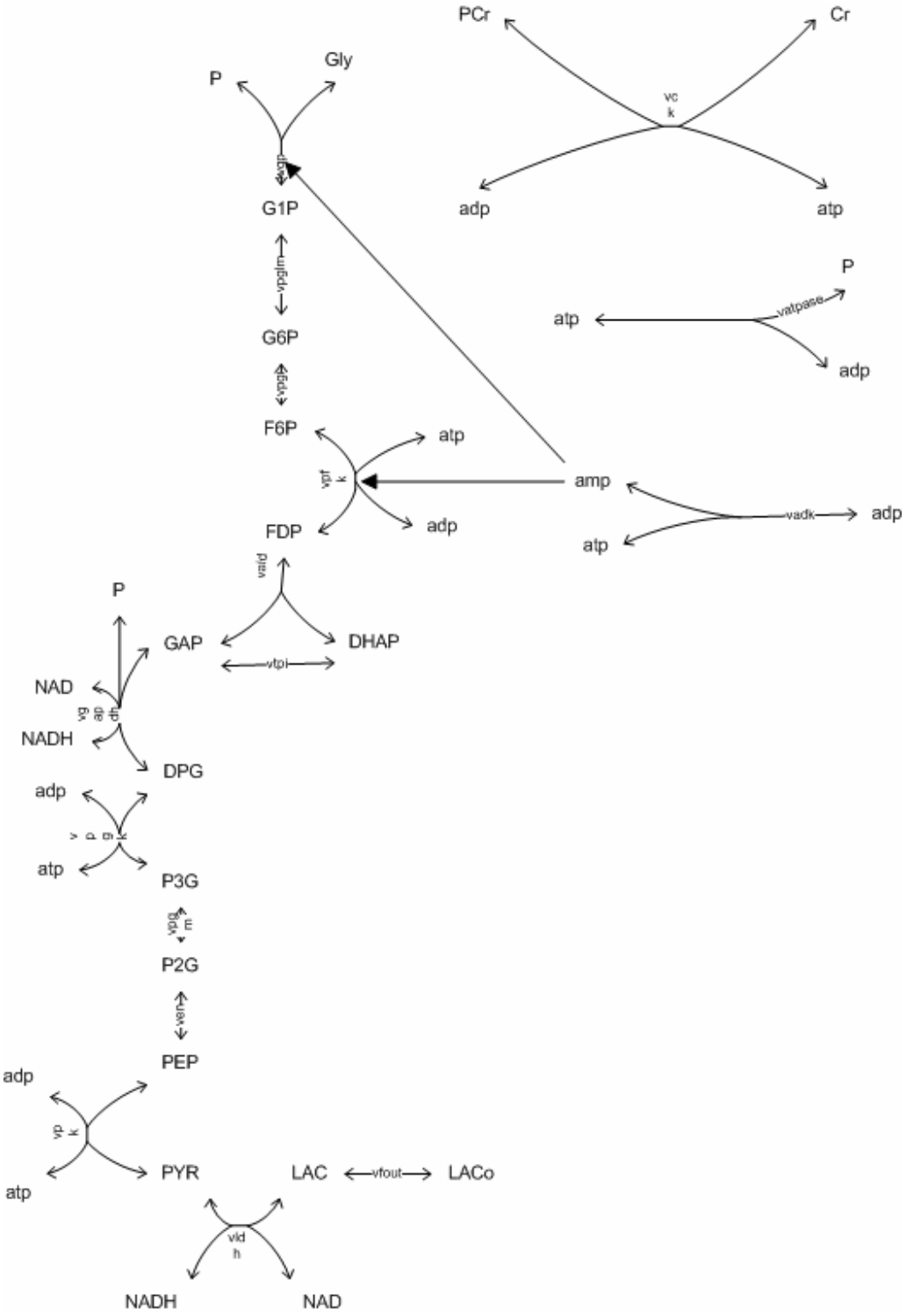
constant =" "	PathwayLab supports time-varying volumes, but it can not be changed as in SBML with their event mechanism.
/>	
<listOfCompartments>	
<listOfSpecies>	Entity is equivalent to species here. No overall listing of entities for the complete model. Since entities are children of location in which they are present hence entities of a particular location are listed in the children block of that location.
<Species	Tag **<entity version=" "> ..</entity>** corresponds to it
id =" "	Value within tag **<guid>.. </guid>** within <entity> tag corresponds to it.
name =" "	Value within tag **<name>.. </name>** within <entity> tag corresponds to it.
compartment =" "	No need since entity is children of the location in which it is present.
initialAmount =" "	In PathwayLab initial quantities of species are given in concentration-units.
initialConcentration =" "	Corresponds to value of the attribute 'value' in the tag <Concentration **value=" "** min=" " max=" " use_min=" " use_max=" "> ... </concentration> .
substanceUnits =" "	No correspondence since PathwayLab does not evaluate units.
spatialSizeUnits =" "	-do-
hasOnlySubstanceUnits =" "	-do-
constant =" "	There is no specific attribute for it. However by declaring a species as Auxiliary/signal i.e. by making the attribute 'use' of tag signal as true and specifying a constant(or expression of constants) in formula same effect can be achieved. <Concentration value=" " min=" " max=" " use_min=" " use_max=" "> <signal **use=true**> <formula expression=" " unit=" "/> </signal> </concentration>
charge =" "	No correspondence.Charges not taken into account.
boundaryCondition =" "	As in constant above. PathwayLab does not make distinction between boundary condition and constant.
/>	</entity>
</listOfSpecies>	
<listOfParameters>	Here parameters are listed within <parameter> tags present within <formula> tag within each location as well as reaction. Those listed in the outermost location i.e. "document" are global to whole model.

	Corresponding to global parameters declared in SBML the parameters declared in the outermost location become automatically global here. They are declared as follows in the PathwayLab.
<parameter	<document version==" "> <signal> <formula expression=" "> **<parameter** local =" " name =" " value=" " unit=" "**/>** </formula> </signal>
id= " "	No correspondence
name =" "	corresponds to attribute 'name' in <parameter> tag above
value =" "	corresponds to attribute 'value' in <parameter> tag above
units =" "	corresponds to attribute 'unit' in <parameter> tag above
constant =" "	No correspondence here.
/>. . . . . . **</listOfParameters>**	
<listOfReactions>	No corresponding entry since reactions are grouped as per their location.
<reaction	**<transformation_part>** and **<transformation>** together correspond to it. One to many of former may be their and one of the later.
id=" "	In view of information in above para there is no single id for a reaction.
name=" "	Value within **< name> </name>** tag which is present in both <transformation> and <transformation_part> tags corresponds to this.
reversible=" "	No corresponding flag since it is determined by the rate law.
fast=" ">	No corresponding entry.
<listOfReactants>	The following block is present in each <transformation_part> tag for recognising the reactants. If in the block value for the attribute ' place' in tag <form place= /> is "1" then the entity whose guid is mentioned within braces in obj="{ } " is reactant. This way all the reactants of the reaction have to be recognised by looking in all transformation parts of the transformation. <transformation> ……. <connect> <from **place="1"**/> <to place="0" **obj="{…….}"**/> </connect> ……. </transformation>
<speciesReference **species**=" "	Corresponds to guid value which is mentioned against obj in the block referred in previous para.
stoichiometry= " "/>	Corresponds to value within **<begin_soich_coeff>.. </begin_soich_coeff>** tag in corresponding <transformation_part> tag.
</listOfReactants>	

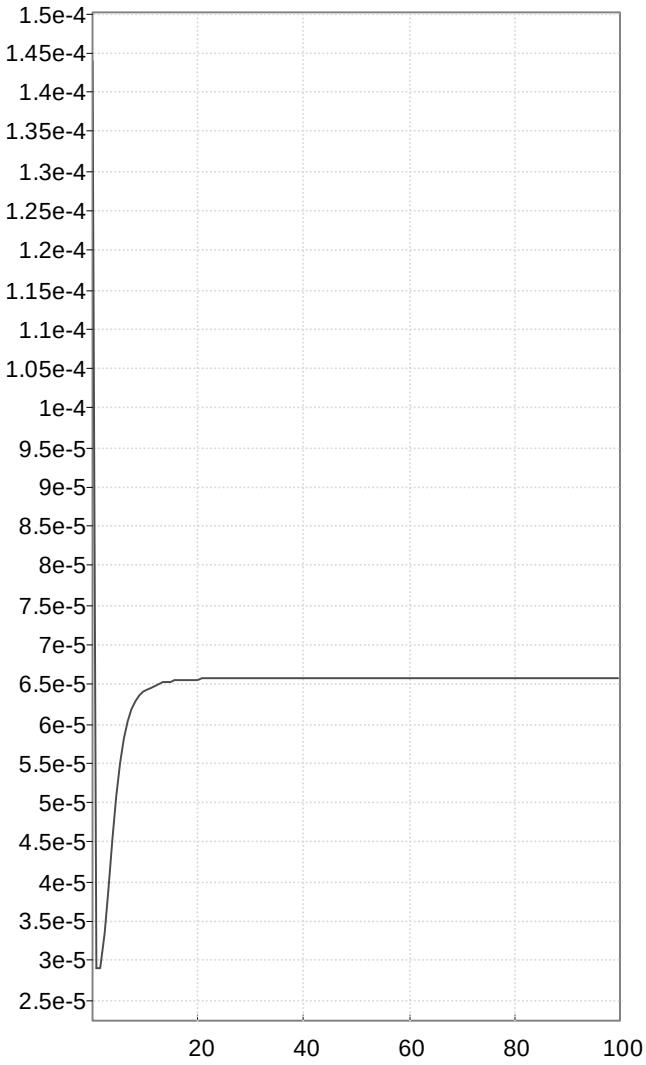
<listOfProducts>	A block similar to the one for reactant can be used for recognising the product. The variation being in case of product the value for the attribute 'place' in the tag **<from place ="2 "/>** is "2". The entity whose guid is mentioned within braces in **obj="{...} "** is product. This way all the products of the reaction have to be recognised by looking in all transformation parts of the transformation.
<speciesReference **species=" "**	Corresponds to guid value which is mentioned against obj in the block referred in previous para.
stoichiometry=" "/>	Corresponds to value within **<end_soich_coeff>... </end_soich_coeff>** tag in corresponding <transformation_part>.
</listOfProducts>	
<listOfModifiers>	Modifiers are connected to the reaction through control tags. The **<control>....</control>** tags are also present in the children block of each location if control element are present.
	In the connect element of the <control> tags is present a tag <from place="1">. It is followed by the tag <to place="0" obj ={ }/>.The value in the braces of field "obj" referred above corresponds to guid of modifier. The corresponding connect block will be like
<modifierSpeciesReference **species**=" "/>	<control> ..... <connect> <from place="1"/> <to place="0" **obj="{.....}"**/> </connect> ..... <control>
</listOfModifiers>	
<kineticLaw>	
	Expressed as a string as the value of the attribute 'expression' in the tag <formula expression=" " unit=" "> which is present in the <transformation> tag of each reaction.
<math xmlns="http://www.w3.org/1998/Math/MathML"> **<apply>** **<times/>** **<ci> k1 </ci>** **<ci> X0 </ci>** **</apply>** </math>	<transformation> ..... <formula **expression="k1*X0 "** unit=" "> **<**parameter local="true" name=" " value=" "/> .... <parameter local="true" name=" " value=" /> </formula> ..... </transformation>
<listOfParameters>	Parameters local to a reaction are present in the <parameter> tag present in the <formula> tag which in turn is present in the tag <transformation>.Attribute 'local' has value 'true' in the parameter tag.
<parameter id= " "	No correspondence
name =" "	Corresponds to attribute 'name' in <parameter> tag above
value =" "	Corresponds to attribute 'value' in <parameter> tag above

units =" "	Corresponds to attribute 'unit' in <parameter> tag above
constant =" "	No correspondence here.
/>.	
</listOfParameters>	
</kineticLaw>	
</reaction> …….	
</listOfReactions>	
<listOfEvents></listOfEvents>	Events are not defined in PathwayLab.
</model>	
</sbml>	

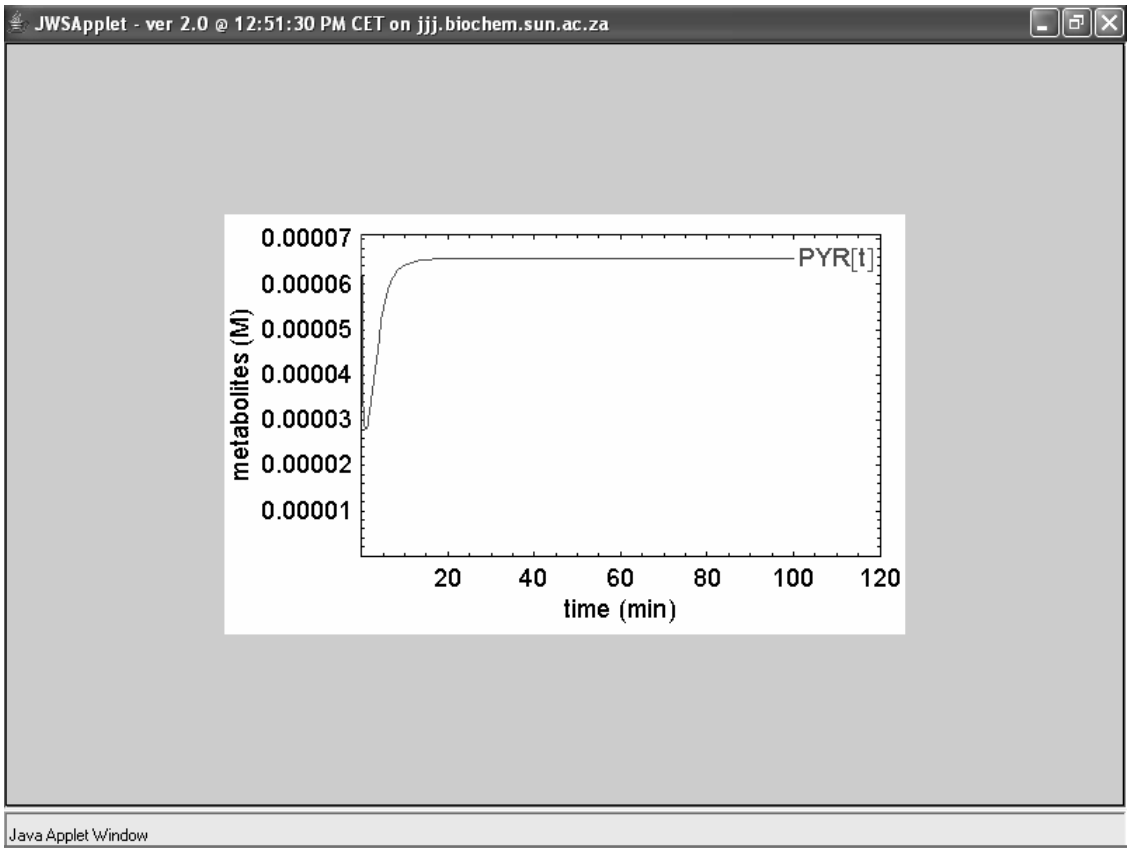
Appendix 4: Glycogenolysis in Skeletal muscle constructed in PathwayLab



Appendix 5: Simulation of the model "Glycogenolysis in Skeletal muscle" in PathwayLab. It shows variation in concentration of pyruvate (Y-axis) over time (X-axis) when simulated for 100 Seconds at default settings of parameters.



Appendix 5(Contd.): Simulation of the model "Glycogenolysis in Skeletal muscle" in JWS Online. It shows variation in concentration of pyruvate (Y-axis) over time (X-axis) when simulated for 100 seconds at default settings of parameters.



Appendix 6: Program

```
//*************************************************************************************************//
Name of the Program        :        StoP
Functionality              :        Convert SBML format to PathwayLab format.
Developer                  :        Sudhir Naswa
Guides                     :        Kim Laurio and Henrik Johansson
Last Modified on       :
//*************************************************************************************************//
#define CONSOLEOUTPUT
#undef CONSOLEOUTPUT
#define USETESTFILE
#undef USETESTFILE
using System;
using System.Xml;
using System.IO;
using System.Globalization;

namespace StoP
{
        public class reaction
/*class representing reactions.Each object of class will store details of single reaction*/
        {
                public int numofreactants;
                public int numofproducts;
                public int numofmodifiers;
                public int numofreactionparameters;
                public string[] reactionattributearray;
                //0=id,1=name,2=reversible,3=fast,4=no_ractants,
                //5=no_products,6=n   o_modifier,7=no_reactionparmeter
                public string[,] reactantmatrix;
            //0=speciesreference,1=stoichiometry
                public string[,] productmatrix;
                //0=speciesreference,1=stoichiometry
                public string[,] modifiermatrix;
                //0=speciesreference
                public string[,] reactionparametermatrix;
                //0=id,1=name,3=value,4=units,5=constant
                public string kineticformula;
                public void makereactionattributearray(int numofelements)
                {
                reactionattributearray = new string[numofelements];
                }

        }

        public class Class1
        {
        #if USETESTFILE
                public string filename=@"E:\project\csharp\smalltest.xml";
        #else
                public string filename;
        #endif

                private int objectid=0;
                public string[,] compartmentmatrix;
                //stores details of compartment from listofcompartments
                public string[,] speciesmatrix;
                //stores the details of various listofspecies
                public string[,] globalparametermatrix;
                //stores parameters declared global in
                public reaction[] reactionobject;
```

```
//array of reaction objects
//sbml under listofparameters
public int numofcompartment=0;
public int maxdepthofcompartment=0;
public int numofoutermostcompartment=0;
public int numofspecies;
public int numofglobalparameter;
public int numofreactions=0;
public int calledonce=0;

public System.Xml.XmlDocument xmldoc =new System.Xml.XmlDocument();
public XmlNamespaceManager nsmgr;
XmlElement[] compartmentelement;
//declaring array of comapartment objects
XmlElement[] childelement;
XmlElement          elementguid,elementidentifier,elementdescription, //declaring elements
elementhypothetical,elementshow_description,elementnotes,elementshow_name,
elementsignal,elementunique_id,elementglobalparameter,documentcompartment,
childdocumentcompartment,elementformula,elementconcentration,
elementtransformation,elementlocalparameter,elementtransformationpart,
element_bsc,element_esc,element_bt,element_et,element_bvn,element_evn,
element_tr,elementconnect,elementfrom,elementto,elementcontrol;

public void loadxmldoc()       //function to load input sbml file
{
        xmldoc.Load(filename);
}

public void creatematrix(string matrixname,int numrows )
        // creates matrices for storing information of compartment,species,globalparameters.
{
        int numberofrows=numrows;
        switch(matrixname)          //matrix created on the basis of matrixname.
        {
        case "listOfCompartments":
                 compartmentmatrix = new string[numberofrows,10];
                /*three extra columns then no. of attributes respectively for storing depth(at index 7),
                corresponding new id generated for the compartment(at index 8) & childflag(at index 9)*/
                numofcompartment =numberofrows;
                for(int i=0;i<numberofrows;i++)
                //fill default SBML values in empty compartmentmtrix
                {
                        compartmentmatrix[i,2]="3";
                        compartmentmatrix[i,6]="true";
                }
                break;
        case "listOfSpecies":
                speciesmatrix = new string[numberofrows,12];
                numofspecies =numberofrows;
                for(int i=0;i<numberofrows;i++)
                //fill default SBML values in empty speciesmatrix
                {
                        speciesmatrix[i,7]="false";
                        speciesmatrix[i,8]="false";
                        speciesmatrix[i,10]="false";
                }
                break;
        case "listOfParameters":
                globalparametermatrix = new string[numberofrows,5];
                numofglobalparameter =numberofrows;
                for(int i=0;i<numberofrows;i++)
                //fill default SBML values in empty globalparametermatrix
```

```
                    {
                    globalparametermatrix[i,4]="true";
                    }
                    break;
                    default:
                    Console.WriteLine("No such attribute of Compartment" + matrixname);
                    break;
            }
}

public void fillmatrixcompartment(int rownum,string argname,string argvalue)
/*each compartment assigned a row.Present arguments stored serially in column(id,name,
 spatialDimensions,size,units,outside,constant)*/
{
        switch(argname)
        {
                case "id":
                        compartmentmatrix[rownum,0]=argvalue;
                        break;
                case "name":
                        compartmentmatrix[rownum,1]=argvalue;
                        break;
                case "spatialDimensions":
                        compartmentmatrix[rownum,2]=argvalue;
                        break;
                case "size":
                        compartmentmatrix[rownum,3]=argvalue;
                        break;
                case "units":
                        compartmentmatrix[rownum,4]=argvalue;
                        break;
                case "outside":
                        compartmentmatrix[rownum,5]=argvalue;
                        break;
                case "constant":
                        compartmentmatrix[rownum,6]=argvalue;
                        break;

        }

}

public void fillmatrixspecies(int rownum,string argname,string argvalue)
        /*each species assigned a row.Present arguments stored serially in columns(id,name,
        compartment,initialAmount,initialConcentration,substanceUnits,spatialSizeUnits,
        hasOnlySubstanceUnits,boundaryCondition,charge,constant )*/
{
        switch(argname)
        {
                case "id":
                        speciesmatrix[rownum,0]=argvalue;
                        break;
                case "name":
                        speciesmatrix[rownum,1]=argvalue;
                        break;
                case "compartment":
                        speciesmatrix[rownum,2]=argvalue;
                        break;
                case "initialAmount":
                        speciesmatrix[rownum,3]=argvalue;
                        break;
                case "initialConcentration":
```

```
                        speciesmatrix[rownum,4]=argvalue;
                        break;
                case "substanceUnits":
                        speciesmatrix[rownum,5]=argvalue;
                        break;
                case "spatialSizeUnits":
                        speciesmatrix[rownum,6]=argvalue;
                        break;
                case "hasOnlySubstanceUnits":
                        speciesmatrix[rownum,7]=argvalue;
                        break;
                case "boundaryCondition":
                        speciesmatrix[rownum,8]=argvalue;
                        break;
                case "charge":
                        speciesmatrix[rownum,9]=argvalue;
                        break;
                case "constant":
                        speciesmatrix[rownum,10]=argvalue;
                        break;
        }

}

public void fillmatrixglobalparameter(int rownum,string argname,string argvalue)
        /*each globalparameter assigned a row.Present arguments stored serially in
          column(id,name,value,units,constant)*/
{
        switch(argname)
        {
                case "id":
                        globalparametermatrix[rownum,0]=argvalue;
                        break;
                case "name":
                        globalparametermatrix[rownum,1]=argvalue;
                        break;
                case "value":
                        globalparametermatrix[rownum,2]=argvalue;
                        break;
                case "units":
                        globalparametermatrix[rownum,3]=argvalue;
                        break;
                case "constant":
                        globalparametermatrix[rownum,4]=argvalue;
                        break;
        }

}

public void fillreactionattributearray(string argname,string argvalue,int reaction_no)
        /*each reaction assigned an array in which its arguments are stored serially
         id,name,reversible,fast)*/
{
        switch(argname)
        {
                case "id":
                        reactionobject[reaction_no].reactionattributearray[0]=argvalue;
                        break;
                case "name":
                        reactionobject[reaction_no].reactionattributearray[1]=argvalue;
                        break;
                case "reversible":
```

```
                                        reactionobject[reaction_no].reactionattributearray[2]=argvalue;
                                        break;
                                case "fast":
                                        reactionobject[reaction_no].reactionattributearray[3]=argvalue;
                                        break;
                        }

                }


public void fillreac_prodmatrix(string argname,string argvalue,int reaction_no,int species_no,string matrixname)
   /*each species assigned a row in which its arguments are stored serially  as species and stoichiometry)*/
        {
        string fillwhichmatrix = matrixname;
                switch(argname)
                        {
                        case "species":
                                if(fillwhichmatrix=="reactantmatrix")
                                {
                                reactionobject[reaction_no].reactantmatrix[species_no,0]=argvalue;
                                }
                                else
                                {
                                        if(fillwhichmatrix=="productmatrix")
                                        {
                                        reactionobject[reaction_no].productmatrix[species_no,0]=argvalue;
                                        }
                                        else
                                        {
                                        reactionobject[reaction_no].modifiermatrix[species_no,0]=argvalue;
                                        }
                                }
                        break;
                        case "stoichiometry":
                                if(fillwhichmatrix=="reactantmatrix")
                                {
                                reactionobject[reaction_no].reactantmatrix[species_no,1]=argvalue;
                                }
                                else
                                {
                                reactionobject[reaction_no].productmatrix[species_no,1]=argvalue;
                                }
                        break;
                        }
        }


public void fillreactionparametermatrix(string argname,string argvalue,int reaction_no,int para_no,string matrixname)
                //each parameter assigned a row in which its arguments are stored serially
                {
                        switch(argname)
                        {
                        case "id":
                                reactionobject[reaction_no].reactionparametermatrix[para_no,0]=argvalue;
                        break;
                        case "name":
                                reactionobject[reaction_no].reactionparametermatrix[para_no,1]=argvalue;
                        break;
                        case "value":
                                reactionobject[reaction_no].reactionparametermatrix[para_no,2]=argvalue;
                        break;
                        case "units":
```

```
                                        reactionobject[reaction_no].reactionparametermatrix[para_no,3]=argvalue;
                                break;
                                case "constant":
                                        reactionobject[reaction_no].reactionparametermatrix[para_no,4]=argvalue;
                                break;
                                }
                        }


                        public void filldepthofcompartment()
                                /*fills the overall depth of each compartment in the compartmentmatrix*/
                        {
                        for(int i=0;i<numofcompartment;i++)
                                //Fills depth as zero for compartments with argument outside as null
                                {
                                if(compartmentmatrix[i,5]==null)
                                        {
                                        //compartmentmatrix[i,5]="0"; //corresponds to oustide field of compartment
                                        compartmentmatrix[i,7]="0"; //corresponds to depth of compartment
                                        numofoutermostcompartment++;
                                        }
                                }
                                int pendingcompartments=numofcompartment-numofoutermostcompartment;
                                while(pendingcompartments>0)
                                //iterates till all the compartments have depth filled.Depth filled at column index 7.
                                {
                                for(int i=0;i<numofcompartment;i++)
                                  {
                                    if(compartmentmatrix[i,7]==null) //depth is still blank
                                      {
                                        for(int j=0;j<numofcompartment;j++)
                                            {
                                            if(compartmentmatrix[i,5]==compartmentmatrix[j,0])
                                            //match id in outside argument with id of the surrounding compartment
                                                    {
                                                    if(compartmentmatrix[j,7]!= null)
                                                   //for calculating depth of a compartment depth of
                                                  // its outer compartment must be known
                                                    {
                                            compartmentmatrix[i,7]=(Int32.Parse(compartmentmatrix[j,7])+1).ToString();
                                            //depth of a compartment is one more than the depth of its outer compartment
                                                    if(Int32.Parse(compartmentmatrix[i,7])>maxdepthofcompartment)
                                                    {
                                                    //if depth of current compartment is more then assign it to a variable.
                                                    maxdepthofcompartment=Int32.Parse(compartmentmatrix[i,7]);
                                                    }
                                            }
                                      }
                                  }
                                }
                        }
                }
        }
}

public void checkchild_fillcompmatrix()
//checks for compartment's child(compartment/species).Sets its index 9 as "Y" if yes.
{
        for(int i=0;i<numofcompartment;i++)
        {
                if(compartmentmatrix[i,9] !="Y")// if labelled no children.
                {
                        for(int j=0;j<numofcompartment;j++)
```

```
                        {
                                if(compartmentmatrix[i,0]==compartmentmatrix[j,5])
                                    //if it is present in outside field of other compartment set its index 9 as "Y".
                                {
                                compartmentmatrix[i,9] ="Y";
                                break;
                                }
                        }
                }
        }
        for(int i=0;i<numofcompartment;i++)
        {
                if(compartmentmatrix[i,9] !="Y")
                {
                        for(int j=0;j<numofspecies;j++)
                        {
                        if(compartmentmatrix[i,0]==speciesmatrix[j,2])
                        //if it is present in compartment field of a species set its index 9 as "Y".
                                {
                                        compartmentmatrix[i,9] ="Y";
                                        break;
                                }
                        }
                }
        }
}

#if CONSOLEOUTPUT
public void showmatrix()
{
/*optional function showing the matrices containing SBML model values on console*/
Console.WriteLine("***** compartmentmatrix entries *****");
Console.WriteLine("id"+"\t"+"name"+"\t"+"spDim"+"\t"+"size"+"\t"+"units"+"\t"+"outside"+"\t"+"const"+"\t"+"depth"+"\t"+"id
"+"\t"+"child");
Console.WriteLine("----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----
");
        for(int m=0;m<numofcompartment;m++)
        {
                for(int n=0;n<10;n++)
                {
                        Console.Write(compartmentmatrix[m,n]);
                        Console.Write("\t");
                }
                Console.WriteLine(" ");
        }
        Console.WriteLine("***** speciesmatrix entries *****");
        Console.WriteLine("id"+"\t"+"name"+"\t"+"compa"+"\t"+"iAmt"+"\t"+"iConc"+"\t"+"subUni"+"\t"+"sSUni"+"\t"+"hO
SUni"+"\t"+"bCond"+"\t"+"charge"+"\t"+"constant"+"\t"+"id");
        Console.WriteLine("----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----
"+"\t"+"----"+"\t"+"----"+"\t"+"----");

        for(int m=0;m<numofspecies;m++)
        {
                for(int n=0;n<12;n++)
                {
                Console.Write(speciesmatrix[m,n]);
                Console.Write("\t");
                }
                Console.WriteLine(" ");
        }
        Console.WriteLine("***** globalparametermatrix entries *****");
        Console.WriteLine("id"+"\t"+"name"+"\t"+"value"+"\t"+"units"+"\t"+"constant");
```

```
	Console.WriteLine("----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t");
	for(int m=0;m<numofglobalparameter;m++)
	{
		for(int n=0;n<5;n++)
		{
			Console.Write(globalparametermatrix[m,n]);
			Console.Write("\t");
		}
		Console.WriteLine(" ");
	}
Console.WriteLine(" ");
Console.WriteLine("**********REACTIONS**********");
Console.WriteLine("Total number of reactions = "+ numofreactions.ToString());
	for(int m=0;m<numofreactions;m++)
	{
	Console.WriteLine(" ");
	Console.WriteLine("********** REACTION "+(m+1).ToString()+ " :");
	Console.WriteLine("id"+"\t"+"name"+"\t\t"+"rev"+"\t"+"fast"+"\t"+"no_reac"+"\t"+"no_prod"+"\t"+"no_mod"+"\t"+"n
o_rparam");
	Console.WriteLine("----"+"\t"+"----"+"\t\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t");
		for(int n=0;n<8;n++)
		{
			Console.Write(reactionobject[m].reactionattributearray[n]);
			Console.Write("\t");
		}
		Console.WriteLine(" ");
		Console.WriteLine("**reactants :");
		Console.WriteLine("speciesreference"+"\t"+"stoichiometry");
		Console.WriteLine("----------"+"\t"+"----------");
		for(int n=0;n<reactionobject[m].numofreactants;n++)
		{
			for(int p=0;p<2;p++)
			{
				Console.Write(reactionobject[m].reactantmatrix[n,p]);
				Console.Write("\t\t\t");
			}
		Console.WriteLine(" ");
		}
		Console.WriteLine("**products :");
		Console.WriteLine("speciesreference"+"\t"+"stoichiometry");
		Console.WriteLine("----------"+"\t"+"----------");
		for(int n=0;n<reactionobject[m].numofproducts;n++)
		{
			for(int p=0;p<2;p++)
			{
			Console.Write(reactionobject[m].productmatrix[n,p]);
			Console.Write("\t\t\t");
			}
		Console.WriteLine(" ");
		}
		Console.WriteLine("**modifiers :");
		Console.WriteLine("speciesreference");
		Console.WriteLine("----------");
		for(int n=0;n<reactionobject[m].numofmodifiers;n++)
		{
			for(int p=0;p<1;p++)
			{
				Console.Write(reactionobject[m].modifiermatrix[n,p]);
				Console.Write("\t");
			}
		Console.WriteLine(" ");
		}
```

```
                Console.WriteLine("**parameters :");
                Console.WriteLine("id"+"\t"+"name"+"\t"+"value"+"\t"+"units"+"\t"+"constant");
                Console.WriteLine("----"+"\t"+"----"+"\t"+"----"+"\t"+"----"+"\t"+"----");
                for(int n=0;n<reactionobject[m].numofreactionparameters;n++)
                {
                        for(int p=0;p<5;p++)
                        {
                Console.Write(reactionobject[m].reactionparametermatrix[n,p]);
                Console.Write("\t");
                        }
                Console.WriteLine(" ");
                }
                Console.WriteLine("**Kinetic formula:");
                Console.WriteLine("______________________________");
                Console.Write(reactionobject[m].kineticformula);
                Console.WriteLine(" ");
        }
}
#endif
        public void createobjectarray(int numberofcomp) //an array of compartment objects
        {
        compartmentelement = new XmlElement[numberofcomp];
        }
        public void createchildarray(int numberofcomp) //an array of child objects
        {
        childelement = new XmlElement[numberofcomp];
        }
        public void getmodelvalues(string listofelementname,string elementname)
        {
                XmlNode modelnode = xmldoc.SelectSingleNode("//urlname:model", nsmgr);
        if(modelnode.HasChildNodes)//if model is not empty
        {
          if(calledonce==0)
         //To execute this block once since function called repeatedly
          {
            for (int i=0;i<modelnode.ChildNodes.Count;i++)
           //Go through names of childnodes of model.Produce warning if
          //feature not supported by PathwayLab found
            {
            XmlNode modelchildnode=modelnode.ChildNodes[i];
              if(modelchildnode.LocalName=="listOfFunctionDefinitions")
              {
        System.Console.WriteLine("Warning!Functions are not supported by PathwayLab and are ignored");
              }
        if(modelchildnode.LocalName=="listOfUnitDefinitions")
        {
        System.Console.WriteLine("Warning!Units are not supported by PathwayLab and are ignored");
        }
        if(modelchildnode.LocalName=="listOfRules")
        {
        System.Console.WriteLine("Warning!Rules are not fully supported by PathwayLab and are ignored");
        }
                if(modelchildnode.LocalName=="listOfEvents")
                {
                System.Console.WriteLine("Warning!Events are not supported by PathwayLab and are ignored");
                }
        }
        calledonce++;
}

                for (int i=0;i<modelnode.ChildNodes.Count;i++)
                {
```

```
                    XmlNode modelchildnode=modelnode.ChildNodes[i];
                        if(modelchildnode.LocalName==listofelementname)
                        {
                        creatematrix(listofelementname,modelchildnode.ChildNodes.Count);
                        //creation of matrices for childs of model
                            if(listofelementname=="listOfCompartments")
                            {
                            createobjectarray(modelchildnode.ChildNodes.Count);
                            createchildarray(modelchildnode.ChildNodes.Count);
                            //creates these arrays.Length eq to no. of compartments.
                            //Used while writing PathwayLab.
                            }

                            for (int j=0;j<modelchildnode.ChildNodes.Count;j++)
                            //iterate over childs of modelchildnode(e.g. each compartment
                                //for listOfCompartments)
                                {
                                XmlNode modelgrandchildnode=modelchildnode.ChildNodes[j];
                                if(modelgrandchildnode.LocalName==elementname)
                                    {
                            XmlNodeReader reader1 = new XmlNodeReader(modelgrandchildnode);
                            reader1.MoveToContent();
                            if (reader1.HasAttributes)
                              //if the current nodes has attributes(e.g. compartment has attributes)
                                {
                                    for (int k=0; k<reader1.AttributeCount; k++)
                                //iterate over each attribute(e.g. each attribute of compartment)
                                    {
                                    reader1.MoveToAttribute(k);
                                    switch(elementname)
                            //depending upon the elementname fill the corresponding matrix
                                    {
                                    case "compartment":
                                    fillmatrixcompartment(j,reader1.Name,reader1.Value);
                                    break;
                                    case "species":
                                    fillmatrixspecies(j,reader1.Name,reader1.Value);
                                    break;
                                    case "parameter":
                                    fillmatrixglobalparameter(j,reader1.Name,reader1.Value);
                                    break;
                                    }
                            }
                            }

                            reader1.MoveToElement();
                        }
                    }
            }

      }
   }
      else
            {
            Console.WriteLine("Model is empty !!!");
            }
}


                    public string convertmathtostring(XmlNode mathnode)
                    {
                        XmlNodeReader mathreader = new XmlNodeReader(mathnode);
```

```
		string  totalxml="";
		while (mathreader.Read())
		{
			switch (mathreader.NodeType)
			{
				case XmlNodeType.Element:
					if (mathreader.IsEmptyElement)
					{
						totalxml=totalxml+"<"+mathreader.Name+"/>";
					}
					else
					{
						if(mathreader.Name=="math")
						{
totalxml="<"+totalxml+mathreader.Name+" xmnls=\"http:www.w3.org/1998/Math/MathML\">";
						}
						else
						{
				totalxml=totalxml+"<"+mathreader.Name+">";
						}

					}
				break;
				case XmlNodeType.Text:
				totalxml=totalxml+mathreader.Value;
				break;
				case XmlNodeType.EndElement:
				if(mathreader.Name=="math")
				{
				totalxml=totalxml+"</"+mathreader.Name+">";
				}
				else
				{
				totalxml=totalxml+"</"+mathreader.Name+">";
				}
				break;
			}
		}
		connectlibsbml.MyClass jsr = new connectlibsbml.MyClass();
		string formulastring = jsr.callconvertMathMLToString(totalxml);
		return formulastring;
}

public void getspeciesdetails(XmlNode reaction_elementnode,int num_ofreactionelementchild,int
num_reaction,string matrixtype)
{
for(int norec=0;norec<num_ofreactionelementchild;norec++)
//iterate over each reactionelement child(e.g. recatant in case of listOfReactions)
  {
XmlNodeReader recreader=null;
if(matrixtype=="reactantmatrix"||matrixtype=="productmatrix"||matrixtype=="modifiermatrix")
   {
XmlNode reactionelementchildnode=reaction_elementnode.ChildNodes[norec];
//point to childs(e.g. reactant) one bye one.
recreader = new XmlNodeReader(reactionelementchildnode);
   }
  else
{
if(matrixtype=="reactionparametermatrix")
{
		XmlNode parameternode=reaction_elementnode.ChildNodes[norec];
		//point to childs(e.g. parameters) one bye one.
```

```
                        recreader = new XmlNodeReader(parameternode);
                }
        }
                recreader.MoveToContent();
                if (recreader.HasAttributes)
                        {
                        for (int numatt=0; numatt<recreader.AttributeCount; numatt++)
                        //iterate over all attributes
                        {
                        recreader.MoveToAttribute(numatt);
                if(matrixtype=="reactantmatrix"||matrixtype=="productmatrix"||matrixtype=="modifiermatrix")
                                {
                        fillreac_prodmatrix(recreader.Name,recreader.Value,num_reaction,norec,matrixtype);
                        //send attribute name,value,reaction no.,no.of child(e.g.reactant),matrixname for storage
                                }
                                else
                                {
                                        if(matrixtype=="reactionparametermatrix")
                                        {
                fillreactionparametermatrix(recreader.Name,recreader.Value,num_reaction,norec,matrixtype);
                                }
                        }
                }
        }
                recreader.MoveToElement();
}
}

public void getreactiondetails()
{
XmlNode listofreactionsnode = xmldoc.SelectSingleNode("//urlname:listOfReactions", nsmgr);
//Pointing to  the node listOfReactions
if(listofreactionsnode.HasChildNodes)//look for presence of reactions
        {
        numofreactions=listofreactionsnode.ChildNodes.Count;//storing number of reactions
        reactionobject= new reaction[numofreactions];
        //creating array of reactionobjects.The Length of array equals number of reactions
        for(int numreaction=0;numreaction<numofreactions;numreaction++)//iterate over all reactions
                {
                        reactionobject[numreaction]=new reaction();
                        //initialise each reaction object pointed by array
                        XmlNode reactionnode=listofreactionsnode.ChildNodes[numreaction];
                        //create a reactionnode pointing to current reaction.
                        if(reactionnode.LocalName=="reaction")
                        {
                                XmlNodeReader reader = new XmlNodeReader(reactionnode);
                                //create reader for current reaction
                                reader.MoveToContent();
                                if (reader.HasAttributes)
                                //check presence of attributes in current reaction
                                {
                                reactionobject[numreaction].reactionattributearray=new string[8];
                        //initialise array for current reaction attributes.Array pointed by current reactionobject
                                reactionobject[numreaction].reactionattributearray[2]="true";
                                //fill default value of attribute reversible as true
                                for (int k=0; k<reader.AttributeCount; k++)
                                {
                                reader.MoveToAttribute(k);

        fillreactionattributearray(reader.Name,reader.Value,numreaction);//to put attribute values in array
                                }
                                }
```

```
                    reader.MoveToElement();
                    if(reactionnode.HasChildNodes)//check for childs of current reaction.
                            {
                    int numofreactionelements=reactionnode.ChildNodes.Count;
                    //store no.of child(e.g.listOfReactants,listOfProducts) of current reaction
            for (int numreactelem=0;numreactelem<numofreactionelements;numreactelem++)
            //iterate over each type of child
            {
            XmlNode reactionelementnode=reactionnode.ChildNodes[numreactelem];
            //points to nodes of lists within reaction.e.g.listOfReactants
                    int numofreactionelementchild=reactionelementnode.ChildNodes.Count;
                    //stores e.g. number of reactants in case of listOfReactants
                    switch(reactionelementnode.LocalName)
                    {
                    case "listOfReactants":
                    reactionobject[numreaction].numofreactants=numofreactionelementchild;
                    reactionobject[numreaction].reactantmatrix=new string[numofreactionelementchild,2];
                    for(int numsp=0;numsp<numofreactionelementchild;numsp++)
                    //fill default stoichiometry value for all reactants
                            {
                    reactionobject[numreaction].reactantmatrix[numsp,1]="1";
                            }
                            //create current reactions reactantmatrix

getspeciesdetails(reactionelementnode,numofreactionelementchild,numreaction,"reactantmatrix");
//passing  currentnode,no.of rectants,no.of current reaction,matrix name
//in which to store ractant attributes.
                    break;
            case "listOfProducts":
            reactionobject[numreaction].numofproducts=numofreactionelementchild;
            reactionobject[numreaction].productmatrix=new string[numofreactionelementchild,2];
            for(int numsp=0;numsp<numofreactionelementchild;numsp++)
            //fill default stoichiometry value for all products
                    {
                    reactionobject[numreaction].productmatrix[numsp,1]="1";
                    }
            getspeciesdetails(reactionelementnode,numofreactionelementchild,numreaction,"productmatrix");
            break;
            case "listOfModifiers":
            reactionobject[numreaction].numofmodifiers=numofreactionelementchild;
            reactionobject[numreaction].modifiermatrix=new string[numofreactionelementchild,1];
            getspeciesdetails(reactionelementnode,numofreactionelementchild,numreaction,"modifiermatrix");
            break;
            case "kineticLaw":
                    for (int numofkinch=0;numofkinch<numofreactionelementchild;numofkinch++)
                    //iterate over both of kin element childs(math and listofparameters)
                    {
                    XmlNode kinchnode =reactionelementnode.ChildNodes[numofkinch];
                    //points to nodes within kinetic element(i.e. listofparameters and math)
                    int numofchofkinch=kinchnode.ChildNodes.Count;
                    //stores e.g. no.of child of kinelement child(parameters in list of parameters)
            switch(kinchnode.LocalName)
            {
            case "listOfParameters":
            reactionobject[numreaction].numofreactionparameters=numofchofkinch;
            reactionobject[numreaction].reactionparametermatrix=new string[numofchofkinch,5];


getspeciesdetails(kinchnode,numofchofkinch,numreaction,"reactionparametermatrix");
break;
case "math":
reactionobject[numreaction].kineticformula=convertmathtostring(kinchnode);
```

```
		break;
			}
		}
		break;
		default:
		Console.WriteLine("Unknown element found in reaction" + "elementname");
		break;
				}
			}
		}
			//recationdetail ends
		}
	}
  }
}

/*#######################!!!!!Writing to PathwayLab begins here!!!!!#######################*/

			public string createid() //generates an id for the various guid tags
			{
				objectid++;
				return ("0000000000"+objectid.ToString());
			}

			XmlDocument xmlwritedoc = new XmlDocument(); //creating an xmldocument object

			public void writebasicframe() //writes basic repetitive elements of PathwayLab
			{
				elementguid = xmlwritedoc.CreateElement("guid");
				string objectidentification=createid();
				elementguid.AppendChild(xmlwritedoc.CreateTextNode(objectidentification));
				elementidentifier=xmlwritedoc.CreateElement("identifier");
				elementdescription=xmlwritedoc.CreateElement("description");
				elementhypothetical=xmlwritedoc.CreateElement("hypothetical");
				elementhypothetical.AppendChild(xmlwritedoc.CreateTextNode("false"));
				elementshow_description=xmlwritedoc.CreateElement("show_description");
				elementshow_description.AppendChild(xmlwritedoc.CreateTextNode("false"));
				elementnotes=xmlwritedoc.CreateElement("notes");
				elementshow_name=xmlwritedoc.CreateElement("show_name");
				elementshow_name.AppendChild(xmlwritedoc.CreateTextNode("true"));
				elementsignal=xmlwritedoc.CreateElement("signal");
				elementunique_id=xmlwritedoc.CreateElement("unique_id");
			}

			public void insertcompartmentdetail(int compartmentrow)
			//inserts attributes of compartment
			{
				XmlElement elementcompartmentname = xmlwritedoc.CreateElement("name");
				//here name Document is
				if(compartmentmatrix[compartmentrow,1]!=null)
					//if compartment has name then assign it
				{
elementcompartmentname.AppendChild(xmlwritedoc.CreateTextNode(compartmentmatrix[compartmentrow,1]));
				}
			else
			//else assign id of compartment as name
			{
elementcompartmentname.AppendChild(xmlwritedoc.CreateTextNode(compartmentmatrix[compartmentrow,0]));
			}
compartmentelement[compartmentrow].AppendChild(elementcompartmentname);
/* new functionailty added here*/
		elementsignal=xmlwritedoc.CreateElement("signal");
```

```
compartmentelement[compartmentrow].AppendChild(elementsignal);
XmlElement elementformula = xmlwritedoc.CreateElement("formula");
elementsignal.AppendChild(elementformula);
if(compartmentmatrix[compartmentrow,3]!=null)
{
elementformula.SetAttribute("expression",compartmentmatrix[compartmentrow,3]);
}
else
{
elementformula.SetAttribute("expression","1");
}
if(compartmentmatrix[compartmentrow,4]!=null)
{
elementformula.SetAttribute("unit",compartmentmatrix[compartmentrow,4]);
}
else
{
elementformula.SetAttribute("unit","");
}
elementformula.SetAttribute("show_data_files_in_plot","false");
}

public void insertentities()
{
int speciescompartmentrow=0;
for(int i=0;i<numofspecies;i++)
        {
        int countcompartments=0;//initiate no.of compartments in which sp is present
        string species_compartmentid=speciesmatrix[i,2];
        for (int j=0;j<numofcompartment;j++)
        //iterates over all compartments and finds row of the compartment outside it.
        {
        if(speciesmatrix[i,2]==compartmentmatrix[j,0])
                {
                        speciescompartmentrow=j;
                        countcompartments++;
                }
        }
        if(countcompartments!=1)
        //every species(by its id) must be assigned a compartment(one compartment only)
        {
                System.Console.WriteLine(speciesmatrix[i,0]+" must be assigned a comparment");
        }
        else
        {
        XmlElement entityelement = xmlwritedoc.CreateElement("entity");
        childelement[speciescompartmentrow].AppendChild(entityelement);
        entityelement.SetAttribute("version","0002");
        XmlElement entityname = xmlwritedoc.CreateElement("name");
        entityelement.AppendChild(entityname);
        if(speciesmatrix[i,1]!=null)       //if entity has name then assign it
                {
                entityname.AppendChild(xmlwritedoc.CreateTextNode(speciesmatrix[i,1]));
                }
        else
        //else assign id of entity as name
                                        {

entityname.AppendChild(xmlwritedoc.CreateTextNode(speciesmatrix[i,0]));
        }
        writebasicframe();
        speciesmatrix[i,11]="0000000000"+objectid.ToString();
```

```
                    //guid created in in writebasic frame for this entity stored
                    entityelement.AppendChild(elementguid);  //elements appended to current compartment
                    entityelement.AppendChild(elementidentifier);
                    entityelement.AppendChild(elementdescription);
                    entityelement.AppendChild(elementhypothetical);
                    entityelement.AppendChild(elementshow_description);
                    entityelement.AppendChild(elementnotes);
                    entityelement.AppendChild(elementshow_name);
                    //this block sets conc. elements and its child of the entity
                    elementconcentration = xmlwritedoc.CreateElement("concentration");
                    elementconcentration.SetAttribute("value",speciesmatrix[i,4]);
                    elementconcentration.SetAttribute("min","0.0");
                    elementconcentration.SetAttribute("max","Infinity");
                    elementconcentration.SetAttribute("use_min","true");
                    elementconcentration.SetAttribute("use_max","false");
                    entityelement.AppendChild(elementconcentration);
                    elementsignal.SetAttribute("use","false");
                    elementconcentration.AppendChild(elementsignal);
                    elementformula = xmlwritedoc.CreateElement("formula");
                    elementsignal.AppendChild(elementformula);
                    elementformula.SetAttribute("expression","");
                    elementformula.SetAttribute("unit","");
                    if(speciesmatrix[i,8]=="true"||speciesmatrix[i,10]=="true")
                    //if constant/boundary condition is true  reset following attributes
                            {
                            elementconcentration.SetAttribute("value","1.0");
                            elementsignal.SetAttribute("use","true");
                            elementformula.SetAttribute("expression",speciesmatrix[i,4]);
                            }
                    entityelement.AppendChild(elementunique_id);
            }
            }
}



                    public string specieslocation(string spid)
                    //returns id of the compartment where the input species is located
                    {
                            int locationcount=0;
                            string splocation=null;
                            for(int i=0;i<numofspecies;i++)
                            {
                                    if(speciesmatrix[i,0]==spid)
                                    {
                                            locationcount++;
                                            splocation=speciesmatrix[i,2];
                                    }
                            }

                            if(locationcount>1)
                            {
                            Console.WriteLine("A species(same id) can't be present at two/more locations");
                            return null;
                            }
                            else
                            {
                                    if(locationcount==0)
                                    {
                                            Console.WriteLine("A species  must be present at some locations");
                                            return null;
                                    }
```

```
                else
                {
                        return splocation;
                }
        }
}

public string getguid(string spid)
        //returns the guid of the species whose id is passed
{
        string spguid =null;
        for(int i=0;i<numofspecies;i++)
        {
                if(speciesmatrix[i,0]==spid)
                {
                        spguid= speciesmatrix[i,11];
                }
        }
        return spguid;

}


public void createcontrol(string c_location,
        reaction currentreaction,string start_guid,string end_guid,int num_control)
{
        int compartmentrownum=-1;
        for(int i=0;i<numofcompartment;i++)
                //finding the row number of compartment to which the reaction belongs
        {
                if(c_location==compartmentmatrix[i,0])
                {
                        compartmentrownum=i;
                }
        }
elementcontrol = xmlwritedoc.CreateElement("control");
childelement[compartmentrownum].AppendChild(elementcontrol);
elementcontrol.SetAttribute("version","0001");
XmlElement controlname = xmlwritedoc.CreateElement("name");
elementcontrol.AppendChild(controlname);

controlname.AppendChild(xmlwritedoc.CreateTextNode("control_"+num_control.ToString()));
//unique name for each control created since no such correspoding name in SBML
writebasicframe();
elementcontrol.AppendChild(elementguid);  //elements appended to current compartment
elementcontrol.AppendChild(elementidentifier);

element_bsc=xmlwritedoc.CreateElement("begin_stoich_coeff");
elementcontrol.AppendChild(element_bsc);
element_bt=xmlwritedoc.CreateElement("begin_type");
elementcontrol.AppendChild(element_bt);
element_bt.AppendChild(xmlwritedoc.CreateTextNode("2"));
element_esc=xmlwritedoc.CreateElement("end_stoich_coeff");
elementcontrol.AppendChild(element_esc);
element_et=xmlwritedoc.CreateElement("end_type");
elementcontrol.AppendChild(element_et);
elementcontrol.AppendChild(elementhypothetical);
elementcontrol.AppendChild(elementnotes);
elementcontrol.AppendChild(elementshow_name);
element_bvn=xmlwritedoc.CreateElement("begin_varname");
elementcontrol.AppendChild(element_bvn);
elementcontrol.AppendChild(elementdescription);
```

```
element_evn=xmlwritedoc.CreateElement("end_varname");
elementcontrol.AppendChild(element_evn);

elementcontrol.AppendChild(elementshow_description);
        for(int connectrepeat=1;connectrepeat<3;connectrepeat++)
        {
                elementconnect=xmlwritedoc.CreateElement("connect");
                elementcontrol.AppendChild(elementconnect);
                elementfrom=xmlwritedoc.CreateElement("from");
                elementconnect.AppendChild(elementfrom);
                elementfrom.SetAttribute("place",connectrepeat.ToString());
                elementto=xmlwritedoc.CreateElement("to");
                elementto.SetAttribute("place","0");
                elementconnect.AppendChild(elementto);
                if(connectrepeat==1)
                {
                        elementto.SetAttribute("obj",start_guid);
                }
                else
                {
                        elementto.SetAttribute("obj",end_guid);
                }
        }
}


public string createtransformationpart(string trans_partlocation,string transformation_guid,
        reaction currentreaction,string start_guid,string end_guid,string rstoi,string pstoi,
        string basetrans_part,string react_id,string prod_id)
{
        int compartmentrownum=-1;
        for(int i=0;i<numofcompartment;i++)
                //finding the row number of compartment to which the reaction belongs
        {
                if(trans_partlocation==compartmentmatrix[i,0])
                {
                        compartmentrownum=i;
                }
        }
        elementtransformationpart = xmlwritedoc.CreateElement("transformation_part");
        childelement[compartmentrownum].AppendChild(elementtransformationpart);
        elementtransformationpart.SetAttribute("version","0001");
        XmlElement reactionname = xmlwritedoc.CreateElement("name");
        elementtransformationpart.AppendChild(reactionname);
        if((currentreaction.reactionattributearray[1])!=null) //if reaction has name then assign it
        {

reactionname.AppendChild(xmlwritedoc.CreateTextNode(currentreaction.reactionattributearray[1]));
        }
        else
 //else assign id of reaction as name
        {
reactionname.AppendChild(xmlwritedoc.CreateTextNode(currentreaction.reactionattributearray[0]));
        }
        writebasicframe();
elementtransformationpart.AppendChild(elementguid);  //elements appended to current compartment
elementtransformationpart.AppendChild(elementidentifier);

        element_bsc=xmlwritedoc.CreateElement("begin_stoich_coeff");
        elementtransformationpart.AppendChild(element_bsc);
        element_bsc.AppendChild(xmlwritedoc.CreateTextNode(rstoi));
        element_bt=xmlwritedoc.CreateElement("begin_type");
```

```
elementtransformationpart.AppendChild(element_bt);
element_bt.AppendChild(xmlwritedoc.CreateTextNode("0"));
element_esc=xmlwritedoc.CreateElement("end_stoich_coeff");
elementtransformationpart.AppendChild(element_esc);
element_esc.AppendChild(xmlwritedoc.CreateTextNode(pstoi));
element_et=xmlwritedoc.CreateElement("end_type");
elementtransformationpart.AppendChild(element_et);
element_et.AppendChild(xmlwritedoc.CreateTextNode("1"));
elementtransformationpart.AppendChild(elementhypothetical);
elementtransformationpart.AppendChild(elementnotes);
elementtransformationpart.AppendChild(elementshow_name);

element_bvn=xmlwritedoc.CreateElement("begin_varname");
elementtransformationpart.AppendChild(element_bvn);
if(react_id!=null)
        {
        element_bvn.AppendChild(xmlwritedoc.CreateTextNode(react_id));
        }
elementtransformationpart.AppendChild(elementdescription);
element_evn=xmlwritedoc.CreateElement("end_varname");
elementtransformationpart.AppendChild(element_evn);
if(prod_id!=null)
        {
        element_evn.AppendChild(xmlwritedoc.CreateTextNode(prod_id));
        }
elementtransformationpart.AppendChild(elementshow_description);

if(basetrans_part=="y")
//insert this tag only in base transformation.
{
        element_tr=xmlwritedoc.CreateElement("transformation_ref");
        elementtransformationpart.AppendChild(element_tr);
        element_tr.SetAttribute("obj",transformation_guid);
}
elementconnect=xmlwritedoc.CreateElement("connect");
elementtransformationpart.AppendChild(elementconnect);
elementfrom=xmlwritedoc.CreateElement("from");
elementconnect.AppendChild(elementfrom);
elementto=xmlwritedoc.CreateElement("to");
elementto.SetAttribute("place","0");
elementconnect.AppendChild(elementto);

if(start_guid==null)
{
        elementfrom.SetAttribute("place","2");
        elementto.SetAttribute("obj",end_guid);
}
else
{
        if(end_guid==null)
        {
                elementfrom.SetAttribute("place","1");
                elementto.SetAttribute("obj",start_guid);
        }
        else
        {
                elementfrom.SetAttribute("place","1");
                elementto.SetAttribute("obj",start_guid);
                elementconnect=xmlwritedoc.CreateElement("connect");
                elementtransformationpart.AppendChild(elementconnect);
                elementfrom=xmlwritedoc.CreateElement("from");
                elementconnect.AppendChild(elementfrom);
```

```
                                        elementto=xmlwritedoc.CreateElement("to");
                                        elementto.SetAttribute("place","0");
                                        elementconnect.AppendChild(elementto);
                                        elementfrom.SetAttribute("place","2");
                                        elementto.SetAttribute("obj",end_guid);
                                }
                        }

                        return("0000000000"+objectid.ToString());
                        //returning the guid of the transformation created
                }

                public string createtransformation(string transformationlocation,reaction currentreaction)
                {
                        int compartmentrownum=-1;
                        for(int i=0;i<numofcompartment;i++)
                                //finding the row number of compartment to which the reaction belongs
                        {
                                if(transformationlocation==compartmentmatrix[i,0])
                                {
                                        compartmentrownum=i;
                                }
                        }
                        elementtransformation = xmlwritedoc.CreateElement("transformation");
                        childelement[compartmentrownum].AppendChild(elementtransformation);
                        elementtransformation.SetAttribute("version","0002");
                        XmlElement reactionname = xmlwritedoc.CreateElement("name");
                        elementtransformation.AppendChild(reactionname);
                        if((currentreaction.reactionattributearray[1])!=null) //if reaction has name then assign it
                        {

reactionname.AppendChild(xmlwritedoc.CreateTextNode(currentreaction.reactionattributearray[1]));
                        }
                        else
//else assign id of reaction as name
                        {
reactionname.AppendChild(xmlwritedoc.CreateTextNode(currentreaction.reactionattributearray[0]));
                        }
                        writebasicframe();
                elementtransformation.AppendChild(elementguid);  //elements appended to current compartment
                        elementtransformation.AppendChild(elementidentifier);
                        elementtransformation.AppendChild(elementdescription);
                        elementtransformation.AppendChild(elementhypothetical);
                        elementtransformation.AppendChild(elementshow_description);
                        elementtransformation.AppendChild(elementnotes);
                        elementtransformation.AppendChild(elementshow_name);

                        XmlElement elementreaction=xmlwritedoc.CreateElement("reaction");
                        elementtransformation.AppendChild(elementreaction);
                        elementreaction.SetAttribute("absolute","false");
                        elementformula = xmlwritedoc.CreateElement("formula");
                        elementreaction.AppendChild(elementformula);
                        elementformula.SetAttribute("expression",currentreaction.kineticformula);

                        for(int n=0;n<currentreaction.numofreactionparameters;n++)
                        {
                                elementlocalparameter = xmlwritedoc.CreateElement("parameter");
elementformula.AppendChild(elementlocalparameter);
elementlocalparameter.SetAttribute("local","true");
elementlocalparameter.SetAttribute("name",currentreaction.reactionparametermatrix[n,0]);
elementlocalparameter.SetAttribute("value",currentreaction.reactionparametermatrix[n,2]);
elementlocalparameter.SetAttribute("unit",currentreaction.reactionparametermatrix[n,3]);
```

```
		}
return("0000000000"+objectid.ToString());
//returning the guid of the transformation created
}

		public void insertreaction()
		{
			int numofcontrols=0;
			for(int i=0;i<numofreactions;i++)
			{
int numoftransformationparts=reactionobject[i].numofreactants + reactionobject[i].numofproducts-1;
			string firstreactantid=null;
			string firstproductid=null;
			string transformationlocation=null;
			string firstreactantguid=null;
			string firstproductguid=null;
			string transformationguid=null;
			string firsttransformationpartguid=null;
			string firstreactantstoi=null;
			string firstproductstoi =null;
			string basetransformation="y";/*flag for base transformation default "y"
			this will be assigned "n" in rest transformation parts.Required because base
			transformation_part has additional tag transformation_ref*/
			if(reactionobject[i].numofreactants>0 && reactionobject[i].numofproducts>0)
			/*i)creation of base transformation part
			 ii)if both reactant and product present transformation present in deeper
			amongst the compartments of reactant or product*/
			{
			   firstreactantid=reactionobject[i].reactantmatrix[0,0];
			firstreactantstoi=reactionobject[i].reactantmatrix[0,1];
			firstproductid =reactionobject[i].productmatrix[0,0];
			firstproductstoi =reactionobject[i].productmatrix[0,1];
			string firstreactantlocation=specieslocation(firstreactantid);
			//get compartment id of the firstreactant
			string firstproductlocation=specieslocation(firstproductid);
			//get compartment id of the firstproduct
			string firstreactantdepth=null;
			string firstproductdepth=null;
			for(int j=0;j<numofcompartment;j++)
				{
					if(firstreactantlocation==compartmentmatrix[j,0])
					{
							firstreactantdepth=compartmentmatrix[j,7];
					}
					if(firstproductlocation==compartmentmatrix[j,0])
					{
							firstproductdepth=compartmentmatrix[j,7];
					}
					}
					if(Int32.Parse(firstproductdepth)>Int32.Parse(firstreactantdepth))
					{
						transformationlocation=firstproductlocation;
					}
					else
					{
						transformationlocation=firstreactantlocation;
					}
		firstreactantguid=getguid(firstreactantid);
		firstproductguid =getguid(firstproductid);
		transformationguid=createtransformation(transformationlocation,reactionobject[i]);
		firsttransformationpartguid=createtransformationpart(transformationlocation,transformationguid,
reactionobject[i],firstreactantguid,firstproductguid,firstreactantstoi,firstproductstoi,basetransformation,
```

```
firstreactantid,firstproductid);
}
        else
        {
        if(reactionobject[i].numofreactants>0)
                //if only reactant present transformation located in  reactant compartment
                {
                firstreactantid  = reactionobject[i].reactantmatrix[0,0];
                firstreactantstoi=reactionobject[i].reactantmatrix[0,1];
                firstreactantguid=getguid(firstreactantid);
                transformationlocation=specieslocation(firstreactantid);
transformationguid=createtransformation(transformationlocation,reactionobject[i]);
firsttransformationpartguid=createtransformationpart(transformationlocation,transformationguid,
reactionobject[i],firstreactantguid,null,firstreactantstoi,"0",basetransformation,
firstreactantid,null);
                }
        else
        {
        if(reactionobject[i].numofproducts>0)
        //if only product present transformation located in  product compartment
                {
                firstproductid =reactionobject[i].productmatrix[0,0];
                firstproductstoi =reactionobject[i].productmatrix[0,1];
                firstproductguid =getguid(firstproductid);
                transformationlocation=specieslocation(firstproductid);
transformationguid=createtransformation(transformationlocation,reactionobject[i]);
firsttransformationpartguid=createtransformationpart(transformationlocation,transformationguid,
reactionobject[i],null,firstproductguid,"0",firstproductstoi,basetransformation,
                                                                null,firstproductid);
                }
                else
                {
                Console.WriteLine("A reaction must have atleast one reactant/product");
                }
                        }
                }

for(int k=1;k<reactionobject[i].numofreactants;k++)
/*Create transformation parts for each reactant.K starts from 1 since
 1st transformation part using the first reactant already created above*/
{
basetransformation="n";
string reactantid  = reactionobject[i].reactantmatrix[k,0];
string reactantstoi=reactionobject[i].reactantmatrix[k,1];
string reactantguid=getguid(reactantid);
string tp_location=specieslocation(reactantid);
string transformationpartguid=createtransformationpart(tp_location,transformationguid,
reactionobject[i],reactantguid,firsttransformationpartguid,reactantstoi,"1",basetransformation,
reactantid,null);
}
for(int k=1;k<reactionobject[i].numofproducts;k++)
/*Create transformation parts for each product.K starts from 1 since
 1st transformation part using the first product already created above*/
        {
        basetransformation="n";
        string productid  = reactionobject[i].productmatrix[k,0];
string productstoi =reactionobject[i].productmatrix[k,1];
string productguid=getguid(productid);
string tp_location=specieslocation(productid);
string transformationpartguid=createtransformationpart(tp_location,transformationguid,reactionobject[i],
firsttransformationpartguid,productguid,"1",productstoi,basetransformation,
null,productid);
```

```
}
if(reactionobject[i].numofmodifiers>0)
/*Create control element for each modifier*/
{
for(int k=0;k<reactionobject[i].numofmodifiers;k++)
        {
        numofcontrols++;
        string modifierid  = reactionobject[i].modifiermatrix[k,0];
        string modifierguid=getguid(modifierid);
        string c_location=specieslocation(modifierid);
        createcontrol(c_location,reactionobject[i],modifierguid,
        firsttransformationpartguid,numofcontrols);

                }
        }
        }
}


public void arrangecompartment()
//arranges the various compartment to PathwayLab format.
{
/*The following loop first arranges outermost compartments*/
   for(int i=0;i<numofcompartment;i++)
        //iterates over all compartments
        {
                if(compartmentmatrix[i,7]=="0")
                //if depth of compartment is zero(outermost compartment)
                {
                        if(numofoutermostcompartment==1)
                        //Incase of just one outermost compartment
                        {
                                compartmentelement[i]=documentcompartment;
                                //first compartment refers to document
                                childelement[i]=childdocumentcompartment;
                                insertcompartmentdetail(i);
                                if(numofglobalparameter>0)
                                //filling global parameters in outermost compartment
                          //if only one outermost compartment.
                                {
                                        fillglobalparameters();
                                }
                        }
                        else
                        {
                        compartmentelement[i]= xmlwritedoc.CreateElement("location");
                        compartmentelement[i].SetAttribute("version","0002");
                        if(compartmentmatrix[i,9]=="Y") //Y implies comp has children
                        {
                        childelement[i]= xmlwritedoc.CreateElement("children");
                        compartmentelement[i].AppendChild(childelement[i]);
                        }

                                childdocumentcompartment.AppendChild(compartmentelement[i]);
        //appending current compartment element to child of document compartment.
                                writebasicframe();
                                compartmentelement[i].AppendChild(elementguid);//append basic
elements of each compartment
        compartmentelement[i].AppendChild(elementidentifier);
        compartmentelement[i].AppendChild(elementdescription);
        compartmentelement[i].AppendChild(elementhypothetical);
        compartmentelement[i].AppendChild(elementshow_description);
```

```
compartmentelement[i].AppendChild(elementnotes);
compartmentelement[i].AppendChild(elementshow_name);
compartmentelement[i].AppendChild(elementunique_id);
insertcompartmentdetail(i);
        }
    }
}
    /*After arranging compartments of zero depth above the following loop searches
     for compartments starting from depth one and going upto maximum depth. The
     compartment at each depth are then inserted in its parent compartment*/
    for(int j=1;j<=maxdepthofcompartment;j++)
    /'/iterate over depth(>=1) to select compartments of incresing depth one by one
        {
            for(int m=0;m<numofcompartment;m++)
              //iterate over all compartments & look for compartments with current depth
                {
                    if(compartmentmatrix[m,7]==j.ToString())
                    //if depth of this compartment equal to value of current depth(j) store
                    //id of compartment outside it.
                    {
                        string idofoutercompartment=compartmentmatrix[m,5];
                    {
                        int rownumofoutercomp;
                        for(int o=0;o<numofcompartment;o++)
                    //iterates over compartmentmatrix to select id of outer compartment
                    {
                if(compartmentmatrix[o,0]==idofoutercompartment)
                        // o corresponds to row number of parent compartment
                        {
                        rownumofoutercomp=o;
                        compartmentelement[m]= xmlwritedoc.CreateElement("location");
                        compartmentelement[m].SetAttribute("version","0002");
                        childelement[o].AppendChild(compartmentelement[m]);
                        //compartment at row m appended to its parent compartment(row o)
                        writebasicframe();
                        compartmentelement[m].AppendChild(elementguid);
                         //append elements to current comp.
                        compartmentelement[m].AppendChild(elementidentifier);
                        compartmentelement[m].AppendChild(elementdescription);
                        compartmentelement[m].AppendChild(elementhypothetical);
                        compartmentelement[m].AppendChild(elementshow_description);
                        compartmentelement[m].AppendChild(elementnotes);
                        compartmentelement[m].AppendChild(elementshow_name);
                        compartmentelement[m].AppendChild(elementunique_id);
                            insertcompartmentdetail(m);
                            if(compartmentmatrix[m,9]=="Y")
                            //if this colmn. is 'Y' the current compartment has children
                            {
                            childelement[m]= xmlwritedoc.CreateElement("children");
                        compartmentelement[m].AppendChild(childelement[m]);
                            }
                        }
                }
            }
        }                                                   }
    }
}
        }

        public void fillglobalparameters()
        {
            //elementglobalparameter = xmlwritedoc.CreateElement("parameter");
            for(int n=0;n<numofglobalparameter;n++)
```

```
		{
		elementglobalparameter = xmlwritedoc.CreateElement("parameter");
		elementformula.AppendChild(elementglobalparameter);
		elementglobalparameter.SetAttribute("local","true");
		elementglobalparameter.SetAttribute("name",globalparametermatrix[n,0]);//case "name":
		elementglobalparameter.SetAttribute("value",globalparametermatrix[n,2]);//case "value":
elementglobalparameter.SetAttribute("unit",globalparametermatrix[n,3]);//case "units":
		}
}


public void writePathwayLab()
{
		documentcompartment=xmlwritedoc.CreateElement("document");
		xmlwritedoc.AppendChild(documentcompartment);
		documentcompartment.SetAttribute("version","0002");

		childdocumentcompartment= xmlwritedoc.CreateElement("children");
		documentcompartment.AppendChild(childdocumentcompartment);

		writebasicframe();
		documentcompartment.AppendChild(elementguid);
		documentcompartment.AppendChild(elementidentifier);
		documentcompartment.AppendChild(elementdescription);
		documentcompartment.AppendChild(elementhypothetical);
		documentcompartment.AppendChild(elementshow_description);
		documentcompartment.AppendChild(elementnotes);
		documentcompartment.AppendChild(elementshow_name);
		documentcompartment.AppendChild(elementunique_id);

		XmlElement elementsimulation = xmlwritedoc.CreateElement("simulation");
		documentcompartment.AppendChild(elementsimulation);
		XmlElement elementsettings = xmlwritedoc.CreateElement("settings");
		elementsettings.SetAttribute("absolute_tolerance","1e-8");
		elementsettings.SetAttribute("relative_tolerance","1e-4");
		elementsettings.SetAttribute("stationary_values_absolute_tolerance","1e-6");
		elementsettings.SetAttribute("stationary_values_relative_tolerance","1e-6");
		elementsettings.SetAttribute("mca_perturbation","0.01");
		elementsettings.SetAttribute("start_time","0");
		elementsettings.SetAttribute("end_time","20");
		elementsettings.SetAttribute("communication_points","501");
		elementsettings.SetAttribute("do_dynamic_simulation","true");
		elementsettings.SetAttribute("do_mca_analysis","false");
		elementsettings.SetAttribute("do_stationary_values","false");
		elementsettings.SetAttribute("use_stationary_values_as_initial_values","false");
		elementsettings.SetAttribute("force_minmax_validation","false");
		elementsettings.SetAttribute("log_minmax_violation","true");
		elementsimulation.AppendChild(elementsettings);

if(numofoutermostcompartment>1)
{
XmlElement documentname = xmlwritedoc.CreateElement("name");//here name Document is
documentcompartment.AppendChild(documentname);		//assigned to outermost compartment
documentname.AppendChild(xmlwritedoc.CreateTextNode("Document"));
documentcompartment.AppendChild(elementsignal);
elementformula = xmlwritedoc.CreateElement("formula");
elementsignal.AppendChild(elementformula);
elementformula.SetAttribute("expression","1");
elementformula.SetAttribute("unit","");
elementformula.SetAttribute("show_data_files_in_plot","false");
		if(numofglobalparameter>0)
//filling global parameters in document compartment if more then one outer compartments
```

```
                                        {
                                        fillglobalparameters();
                                        }

                                }
                                arrangecompartment();
                                insertentities();
                                insertreaction();

                                XmlTextWriter xmlTextWriter = new XmlTextWriter("Pathway.xml",null);
                                xmlTextWriter.Formatting = Formatting.Indented;
                                xmlwritedoc.WriteTo(xmlTextWriter);
                                xmlTextWriter.Flush();
                                Console.WriteLine("PathwayLab writing complete.");
                                Console.WriteLine("");
                        }
                }

                /*######################!!!!!Class conatining the main function!!!!!######################*/
                class class2

                {
                static void Main(string[] args)
                {
                #if !USETESTFILE
                Console.Write("Enter SBML File Name(with complete path) : ");
                string inputfile = Console.ReadLine();
                #endif
                Class1 obj1 = new Class1();
                   #if !USETESTFILE
                obj1.filename=inputfile;
                     #endif
                try
                {
                System.IO.FileInfo objFile=new System.IO.FileInfo(obj1.filename);
                if (objFile.Exists)
                {
                obj1.loadxmldoc();
                obj1.nsmgr = new XmlNamespaceManager(obj1.xmldoc.NameTable);
                                                        obj1.nsmgr.AddNamespace("urlname","http://www.sbml.org/sbml/level2");
                obj1.getmodelvalues("listOfCompartments","compartment");
                obj1.filldepthofcompartment();
                obj1.getmodelvalues("listOfSpecies","species");
                obj1.getmodelvalues("listOfParameters","parameter");
                obj1.checkchild_fillcompmatrix();
                obj1.getreactiondetails();
                obj1.writePathwayLab();
#if CONSOLEOUTPUT
Console.WriteLine("The following matrices show values in SBML model :");
Console.WriteLine("\t_____");
obj1.showmatrix();
Console.WriteLine("\t_____");
#endif
                Console.WriteLine("\nThe input file converted to PathwayLab file Pathway.xml");
                System.Console.Read();
                                }
                                else
                                {
                                        System.Console.WriteLine("Input SBML file "+obj1.filename+ " not found.");
                                        throw new Exception();
                                        }
                                }
```

```
                catch(ArgumentException)
                {
                        System.Console.WriteLine("No file name entered");
                        //System.Console.Read();
                }
                catch(Exception)
                {
                        System.Console.WriteLine("Error Occurred while reading XML File");
                        System.Console.Read();
                }
                finally
                {
                        System.Console.WriteLine("Thanks! for using SBML converter.Press Enter to exit.");
                        System.Console.Read();
                        System.Console.Read();
                }
        }
    }
}
```

Appendix 7: managed c++ program for converting MAthML to string formula

```
using namespace System;
using namespace System::Xml;
using namespace System::Data;
using namespace System::Runtime::InteropServices;

#include <stdio.h>
#include "stdafx.h"
#using <System.dll>
#using <mscorlib.dll>
#using <System.Xml.dll>
#using <system.data.dll>
#include "sbml/SBMLTypes.h"

namespace connectlibsbml
{
       public __gc class MyClass
       {
       public:
       char * convertMathMLToString (const char *xml);
      //function declaration
       public: String* callconvertMathMLToString(String* stringsSet);
      //function declaration
       };
}

char * connectlibsbml::MyClass::convertMathMLToString (const char *xml)
//function accepts address of string containing mathml & returns formula as string
{
MathMLDocument_t *mdoc;
//mdoc is pointer of type MathMLDocument_t
const ASTNode_t *mtree_root;
//mtree_root s pointer of type ASTNode_t
char *mchar;
//mchar is pointer of type char
mdoc = readMathMLFromString (xml);
//takes address of string as arg and returns MathMldoc
mtree_root = (ASTNode_t *) MathMLDocument_getMath (mdoc);
//Takes MathMLdoc and returns AStNode
mchar = SBML_formulaToString (mtree_root);
//takes ASnode and returns string
return mchar;
//final return
}

String* connectlibsbml::MyClass::callconvertMathMLToString(String* stringsSet)
//accepts input MAthML string from the c sharp file
{
//Console::Write(stringsSet);
char* str2 = (char*)(void*)Marshal::StringToHGlobalAnsi(stringsSet);
//converts string to char* since the libsbml function accepts
char *returnmathstring;
returnmathstring=convertMathMLToString(str2);
// call to function calling libsbml
Marshal::FreeHGlobal(str2);
// releasing the memory of char* since no longer needed
String* str = returnmathstring;
return str;
}
```