UNIVERSITY OF CALIFORNIA,
IRVINE

# Schematic Representation and Database Population Strategies for Sigmoid, a Biochemical Network Modeling System

DISSERTATION

Submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Information and Computer Science
Concentration in Informatics for Biology and Medicine

by

Behnam Compani-Tabrizi

2012

# DEDICATION

To

My wonderful parents

Behrooz and Elizabeth

To whom I owe everything I have achieved in my life.

To my sister

Roya

Whom I love with all my heart.

And to my grandfather

Frank Doan Streightoff

For his unwavering support.

.

# Contents

## LIST OF FIGURES

# List of Tables

# ACKNOWLEDGMENTS

Having reached this point in my academic life and looking back, I find it hard to put all the names I need to thank together and in one page. I need to first and foremost thank my advisor, Professor Eric Mjolsness. Working with him has been an honor and I have benefitted greatly from his insights, direction and patience. I'd like to thank Professors Lee Bardwell and Pierre Baldi for all of their insights, feedback and contributions as PIs on the Sigmoid project. With regard to Sigmoid, they put together a highly talented group of students working on what was a very interesting project. I learned a lot through my interactions with both of them, as they were open, nice, active, extremely knowledgeable and smart. I highly value the experience and sincerely thank them for that. I'd like to sincerely thank Gopi Meenakshisundaram for his valuable input and taking the time and effort to serve on my committees. There were many members of the Sigmoid project when it was going full force and I've benefitted from all of them in one way or another. Of note and distinction; Trent Su, Ivan Chang, Tom Whisnamt, Kandarp Shah, Kirill Petrov, and Chinran Yang. Lucas Sharenbroich deserves a good portion of my gratitude, he was instrumental in providing me with a really decent primer on Sigmoid and its internal functioning. I'd also like to thank Dr. Tarek Najdi for all of his collaborative support and involvement over the years.

I'd like to thank Professor Suzanne Sandmeyer for being by BIT co-advisor and the experiences I had while rotating through her lab. Also, I'd like to thank Fang Fang for her assistance with the laboratory protocols. I'd like to thank the Bioinformatics Training Program (BIT) for what has been a great experience and opportunity to relate to a wonderful group of people. Kudos must go to Professor Pierre Baldi for his role managing such a strong and diverse set of BIT researchers here at UCI.

I'd like to thank my grantors for this research and all the wonderful people I was able to meet at the National Institute of Health. Personal support was received through the NIH grant GM069013 for the Sigmoid project. Support was also received through the NIH-NLM Biomedical Informatics Training grant LM-07443-01 for three years. In addition, I was supported by NIH grant NIH NIGMS R01 GM086883 fall quarter 2011.

Portions of previously published work is reproduced here with kind permission from © Springer Science + Business Media: H.R. Arabnia (ed.), Advances in Computational Biology, Advances in Experimental Medicine and Biology, "A scalable and Integrative System for Pathway Bioinformatics and Systems Biology", 680, 2010, pg. 523-534, Compani B, Su T, Chang I, Cheng J, Shah KH, Whisenant T, Dou Y, Bergmann A, Cheong R, Wold B, Bardwell L, Levchenko A, Baldi P, Mjolsness E. The portions appear as: Chapter 1 and Sections 2.7 (Third paragraph), 4.1.3, 4.2.4 (first paragraph) and 5.1.

The last but not the least, I would like to thank everyone who made it possible for me to focus on my work throughout my PhD life. Their friendship is invaluable to me and I always appreciate it. The list is extremely long but in specific I want to thank my family, my friends at UCI, all the nice staff at Palo Verde Housing, the wonderful people in ICS graduate student affairs, and the staff at graduate division. One last word for the amazing and helpful staff at the Institute for Genomics and Bioinformatics, Janet Ko and Katarina Fletcher for everything they've done.

# CURRICULUM VITAE

## Behnam Dustin Compani-Tabrizi

Home page: http://computableplant.ics.uci.edu/~companib/
E-mail: companib@uci.edu

## EDUCATION

**PhD in Information and Computer Science**
**Concentration in Informatics for Biology and Medicine – 2012**
University of California, Irvine – Irvine, CA

**M.S. in Information and Computer Science – 2011**
University of California, Irvine – Irvine, CA

**B.S. in Biochemistry – 1993**
**B.S. in Molecular, Cellular and Developmental Biology - 1993**
University of Colorado, Boulder – Boulder, Colorado

## RESEARCH INTERESTS

- Modeling and Simulation of Biochemical Reaction Networks
- Enzymatic Reaction Mechanisms
- Database Design

## PUBLICATIONS

[Compani *et al.* 2010] B. Compani, T. Su, I. Chang, J. Cheng, K. H. Shah, T. Whisenant, Y. Dou, A. Bergmann, R. Cheong, B. Wold, L. Bardwell, A. Levchenko, P. Baldi, and E. Mjolsness, A Scalable and Integrative System for Pathway Bioinformatics and Systems Biology. Adv Exp Med Biol. 2010; 680: 523–534.

[Podkolodny *et al.* 2006] Podkolodny NL, Podkolodnaya NN, Miginsky DS, Poplavsky AS, Likhoshvai VA, Compani B, Mjolsness E. An integration of the descriptions of gene networks and their models presented in Sigmoid (Cellerator) and GeneNet, 5th International Conference on the Bioinformatics of Genome Regulation and Function (BGRS-2006), Volume 3, pp. 86-90.

## HONORS AND AWARDS

- NIH/NLM Biomedical Informatics Training Grant fellowship 2008-2011
- BIT Conference 2009 Portland Oregon *(Best Poster Award)*
  "Sigmoid: An integrative System for Pathway Bioinformatics and Systems Biology"

ABSTRACT OF THE DISSERTATION

# Schematic Representation and Database Population Strategies for Sigmoid, a Biochemical Network Modeling System

by

Behnam Compani-Tabrizi

Doctor of Philosophy in Information and Computer Science

Concentration in Informatics for Biology and Medicine

University of California Irvine, 2012

Professor Eric Mjolsness, Chair

**Research summary:**

Progress in systems biology critically depends on developing scalable informatics tools to model and visualize complex biological systems. Flexibly storing information about these systems and their models for subsequent retrieval and analysis is also a key concern. The focus of the research reported here has been the development of the Sigmoid project and associated pathway models. The Sigmoid project (www.sigmoid.org) provides biologists with a database, modeling and simulation platform for signal transduction, metabolic and biosynthetic pathways. Sigmoid has been implemented as a three-tier architecture, consisting of client, web service, and back end simulator/database. Sigmoid provides a front end to the xCellerator / kMech / Mathematica simulation platform and enables mathematical simulations of biochemical networks [Shapiro 2007] [Yang *et al.*2005b]. A visualization and simulation platform

such as this allows wet bench biologists to make targeted decisions about their experimental designs and save on unnecessary expenditures of wet bench resources. Many published models are currently available and functional within the Sigmoid framework. The models focus on virtual representation of intracellular pathways that include examples in signaling, metabolism, the cell cycle, and gene regulation.

To facilitate the modeling of organelle, multi-cellular and developmental models within the Sigmoid framework, a back-end representation for multi-compartmental modeling was added to the Sigmoid schema. Although the Schema representation for this is more comprehensive, a first step compatibility with the xCellerator "Cellzilla" utility was implemented. A model of Wuschel expression [Jönsson 2005] in the *Arabidopsis thaliana* shoot apical meristem has been completed with the Sigmoid representation.

kMech [Yang 2005] is an enzyme mechanism modeling tool designed for the mathematical modeling of enzymes. It comprises a collection of single and multiple substrate enzyme reactions. Over the years the library of requested enzymatic reactions implemented in the kMech utility and correspondingly in Sigmoid has grown substantially. A new generalized version of the kMech enzyme mechanism modeling tool has been developed. With this utility, SigMech, the approximately 35 existing enzyme mechanism models expressed explicitly in the kMech/Sigmoid platform can be expressed implicitly by a single parameterized input notation. Subsequent sub-reactions can be generated procedurally, and any potential "new" kMech enzyme mechanisms that fall within the pattern abstracted from previous motifs need not be created explicitly.

# Chapter 1. Sigmoid Overview

## 1.1 Introduction

As an introduction, this chapter reflects a team effort development of Sigmoid. To distinguish member contributions we will refer to the "Sigmoid team" as the contributing authors to the book chapter on Sigmoid published in Advances in Computational Biology [Compani, Su *et al.* 2010].

Sigmoid is a generative, scalable software infrastructure for systems biology designed to facilitate global modeling of biological systems. SIGMOID if deciphered as an acronym, would translate into a SIGnal MOdeling Interface and Database. Here the term Signal, in a biological sense, is broadly interpreted.

Sigmoid supports the process of cycling between model building, hypothesis generation, biological experimentation and data gathering, by integrating the hypothesis and discovery phases of the research process. In Sigmoid, the Sigmoid team address the problem of creating a scalable expert assistance system for modeling biological pathways, using current software technology to decrease the difficulty and cost of building the system. The reason for building such a system is to provide computational support to biologists and computational scientists who need to create and explore predictive dynamical models of complex biological systems such as metabolic, gene regulation, or signal transduction pathways in living cells [Cheng *et al.* 2005].

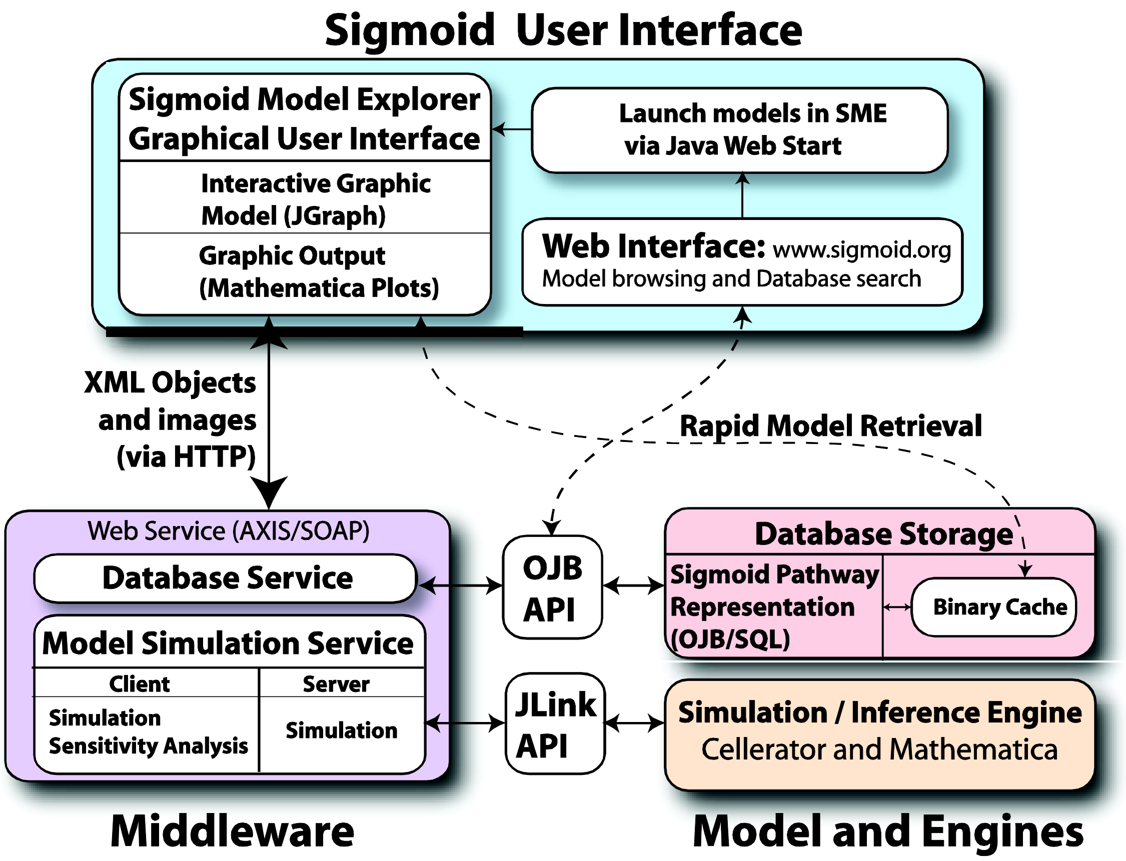

Sigmoid User Interface
Sigmoid Model Explorer Graphical User Interface
Interactive Graphic Model (JGraph)
Graphic Output (Mathematica Plots)
Launch models in SME via Java Web Start
Web Interface: www.sigmoid.org
Model browsing and Database search
XML Objects and images (via HTTP)
Rapid Model Retrieval
Web Service (AXIS/SOAP)
Database Service
Model Simulation Service
Client
Server
Simulation
Sensitivity Analysis
Simulation
OJB API
JLink API
Database Storage
Sigmoid Pathway Representation (OJB/SQL)
Binary Cache
Simulation / Inference Engine
Cellerator and Mathematica
Middleware
Model and Engines

**Figure 1.1** Sigmoid three-tier architecture. Separation of modules into a communicating distributed system increases scalability of the architecture. The Sigmoid simulation results are provided by the xCellerator model generator/ simulator. The database is Sigmoid (autogenerated from a UML schema) and the user interface is the Sigmoid Model Explorer (SME).

The Sigmoid modeling system core consists of distributed modules implementing: (1) pathway/cell model generation and simulation (Cellerator [Shapiro *et al.* 2003]), (2) a pathway modeling database (Sigmoid proper), (3) a Web service-oriented middleware, (4) a world wide web model browser, and (5) a graphical user interface (Sigmoid Model Explorer) friendly to a biologist user. Other components have been integrated into the system such as a parameter optimization module and functional connections to compatible external data sources. These modules are organized in a classical three-tier architecture (Figure 1.1). The back-end currently consists of the database, the simulator, and other model manipulators. The GUI front-end does not access the back-end modules directly but rather via a Web service middleware module. The extra development overhead introduced by the middle layer is more than compensated by its advantages in terms of distributed computing, performance, flexibility, and scalability. With the exception of rapid model retrieval, the middleware layer brokers all communications between the GUI and the back-end components and also between the back end components themselves. The Sigmoid team has found that storing binary instances of models in a database cache can provide significant improvements in model retrieval times in comparison to full model reconstruction and retrieval through the middleware layer. In the event that the rapid model retrieval interface is not accessible, the system will shift access to the database through the middleware. This infrastructure was created in a close collaboration between bioinformaticians and biologists. The design of many of the essential software objects and their relationships became visible as the implementation proceeded. The Sigmoid team has coordinated the development of various software modules in Sigmoid by using the Universal Modeling Language (UML) to diagram the most important biological objects, notably reactions and molecular reactants. This UML diagram is used as a template to automatically generate several parts of Sigmoid, in particular a realization of the Sigmoid pathway modeling database (in SQL) and the corresponding Java object hierarchy along with support files for facilitating the object-relational mapping. Also the Graphical User Interface (GUI) relies heavily on the Java reflection utility to automatically discover much of what it needs to know about the Sigmoid schema. Thus there is a guarantee that the software actually implements something very close to the UML construction of biological objects. In addition, coding time for different modules of the system is reduced. To keep the infrastructure flexible and manageable as it grows, the Sigmoid team has resorted to a “generative” approach that seeks to partially automate the generation of both executable code and mathematical models. The Sigmoid team has applied this approach to as many of the modules in Figure 1.1 as possible, starting from high-level inputs such as UML diagrams and reaction notations understandable to non-computer scientists.

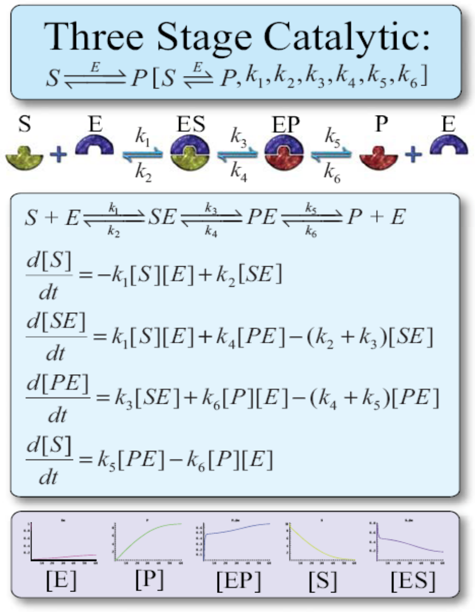
## ***1.2 Methods***

### 1.2.1 ***Model Generation and Simulation: xCellerator***

In order to facilitate the modeling of biochemical reactions a library of re-usable reaction models that can be expressed in a simple higher-level language that specifies the molecular species and the type of reaction is required. Cellerator [Shapiro *et al.*2003] code is implemented as a Mathematica notebook and is designed to facilitate biological modeling via automated equation generation. Sigmoid now supports xCellerator [B. Shapiro2007], the most recent version of Cellerator.

Many models of molecular interactions have been implemented in xCellerator using different formalisms, such as differential equations or stochastic molecular simulation formalism and ranging from the law of mass action and simple Michaelis-Menten models to more complex models of enzyme reactions (e.g. the Monod-Wyman-Changeaux or MWC model for allosteric enzymes [Najdi *et al.*2005]) and gene regulation [Segel1992]. The list of reaction models continues to expand along with the library of actual pathway models comprising sets of coordinated reactions with parameters derived from the literature whenever possible. In addition, an extended set of enzyme mechanism models for single and multi-substrate, positively and negatively regulated and allosteric enzymes, called kMech, has been written for xCellerator and continues to develop [Yang *et al.*2005b]. Sigmoid currently supports all the available xCellerator and kMech reaction models.

To illustrate xCellerator utility, consider the example of a three stage catalytic model. This reaction is a composite representation of 3 reversible reactions: substrate enzyme complex formation, the conversion of the substrate to product within the complex and, subsequent disassociation of the enzyme-product complex into free enzyme and product. When presented with the correct input notation, xCellerator will translate the symbolic reaction to differential equations. The resulting differential equations and variable definitions are passed to Mathematica where they are solved by the numeric solver function (NDSolve) and time plots are generated. For an example of these steps, see example in Figure 1.2. The parameters for this enzyme mechanism are stored in the Sigmoid Pathways Database. In short, xCellerator converts symbolic reactions to mathematical equations, and solves the corresponding equations.


Three Stage Catalytic:
$S \underset{}{\overset{E}{\rightleftharpoons}} P[S \overset{E}{\rightleftharpoons} P, k_1, k_2, k_3, k_4, k_5, k_6]$
S
E
ES
EP
P
E
$k_1$
$k_2$
$k_3$
$k_4$
$k_5$
$k_6$
[E]
[P]
[EP]
[S]
[ES]

$$S + E \underset{k_2}{\overset{k_1}{\rightleftharpoons}} SE \underset{k_4}{\overset{k_3}{\rightleftharpoons}} PE \underset{k_6}{\overset{k_5}{\rightleftharpoons}} P + E$$

$$\frac{d[S]}{dt} = -k_1[S][E] + k_2[SE]$$

$$\frac{d[SE]}{dt} = k_1[S][E] + k_4[PE] - (k_2 + k_3)[SE]$$

$$\frac{d[PE]}{dt} = k_3[SE] + k_6[P][E] - (k_4 + k_5)[PE]$$

$$\frac{d[S]}{dt} = k_5[PE] - k_6[P][E]$$

**Figure 1.2 Sigmoid Three Stage Catalytic model. From Top to bottom. xCellerator input notation, reaction cartoon, resulting differential equations and an example of numerical output.**

### 1.2.2 *Sigmoid Pathway Database*

The pathway model database is defined by a UML schema. Comprehensive UML class diagrams of the Sigmoid Schema can be found at www.sigmoid.org. The schema is organized into four main diagrams. The first diagram consists of the various top level container classes such as the Model Class and the Gene Ontology source class. The first diagram also contains the parameter set hierarchy, classes for graphical layout in SME, and various classes to handle units and measures. The three remaining diagrams consist of the three major class hierarchies: Reactions, Reactants and Knowledge Sources. Reactions utilize Reactants for their products, substrates, and enzymes, Models are composed of parameterized Reactions, and these three class hierarchies utilize Knowledge Sources in order to reference external information about themselves.

While initial versions of the Sigmoid database were implemented by hand, the Sigmoid team wished to automatically transform the class descriptions contained in the high-level UML diagram of this hierarchy into a set of instantiable objects upon which applications may be built. The current approach to the process of auto-generating software components from a master UML diagram relies on the capabilities of several existing open-source projects [Cheng *et al.* 2005]. These pre-existing projects remove much of the core software development responsibilities and allow the team efforts to focus on tying them together to produce the specific software products needed. Object-relational database code autogeneration from UML is itself a contribution of potentially general interest in database software engineering. The current version of Sigmoid is implemented using PostgreSQL the main Open Source database software.

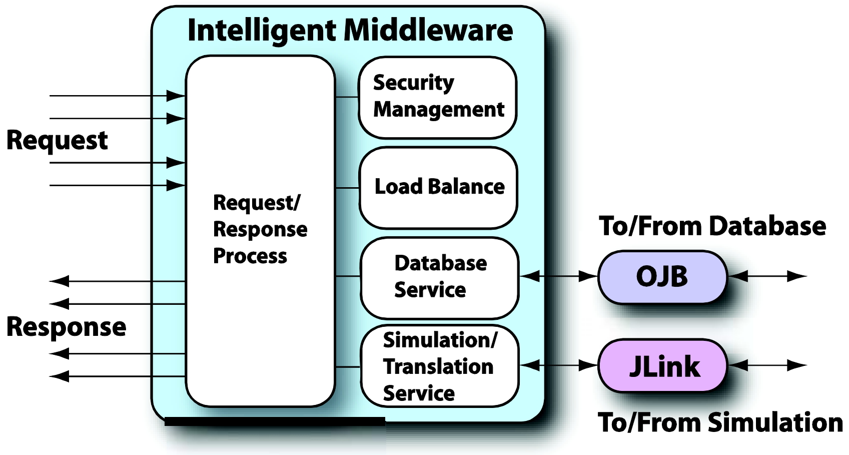
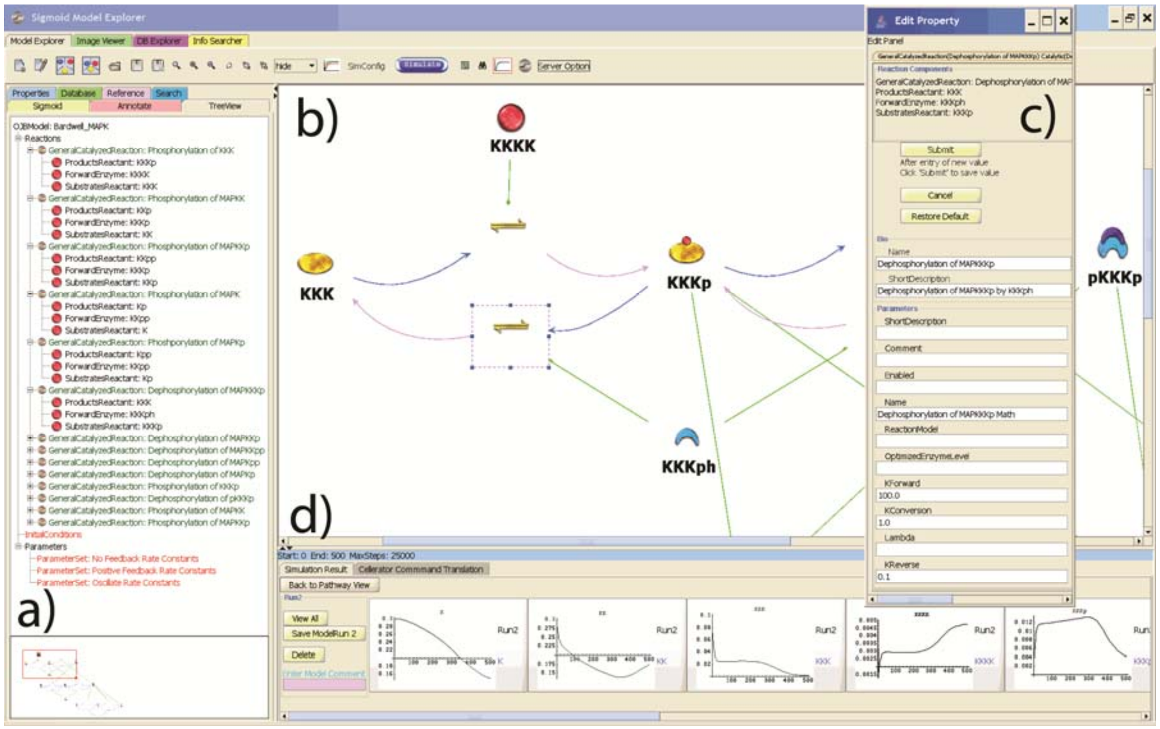


**Figure 1.3 The Sigmoid Intelligent Middleware utilizing Apache Axis (SOAP) brokers transactions for the Sigmoid framework. Clients such as SME can access the simulator and database back end components through the middleware. The SQL database is accessed through an Object Relational Bridge (OJB). The Mathematica/ xCellerator simulator is accessed through Mathematicas' JLink utility. The figure was redrawn from [Cheng *et al.*2005]**

### 1.2.3 *Sigmoid Web Middleware for Distributed Computing and Web Services*

A new distributed Web middleware layer was built which accesses the Sigmoid Database (Figure 1.3) and translates reaction sets into the input language of the xCellerator cell model generator, then calls xCellerator with requests for model generation and simulation and receives output plots in response. All these functions are exposed as Web services available to Java application programs and/or other clients. In addition to load balance and security management, the middleware provides a gateway between the front-end and the back-end of the architecture, allowing each one to evolve independently as long as the interface to the middleware is properly maintained. Furthermore, the middleware allows scalability in terms of the number of users that can be served simultaneously simply by increasing the computational and database server resources [Cheng *et al.*2005].



OJBModel: Bardwell_MAPK
- Reactions
  - GeneralCatalyzedReaction: Phosphorylation of KKK
    - ProductsReactant: KKKp
    - ForwardEnzyme: KKKK
    - SubstratesReactant: KKK
  - GeneralCatalyzedReaction: Phosphorylation of MAPKK
    - ProductsReactant: KKp
    - ForwardEnzyme: KKKp
    - SubstratesReactant: KK
  - GeneralCatalyzedReaction: Phosphorylation of MAPKKp
    - ProductsReactant: KKpp
    - ForwardEnzyme: KKKp
    - SubstratesReactant: KKp
  - GeneralCatalyzedReaction: Phosphorylation of MAPK
    - ProductsReactant: Kp
    - ForwardEnzyme: KKpp
    - SubstratesReactant: K
  - GeneralCatalyzedReaction: Phoshporylation of MAPKp
    - ProductsReactant: Kpp
    - ForwardEnzyme: KKpp
    - SubstratesReactant: Kp
  - GeneralCatalyzedReaction: Dephosphorylation of MAPKKKp
    - ProductsReactant: KKK
    - ForwardEnzyme: KKKph
    - SubstratesReactant: KKKp
  - GeneralCatalyzedReaction: Dephosphorylation of MAPKKp
  - GeneralCatalyzedReaction: Dephosphorylation of MAPKKpp
  - GeneralCatalyzedReaction: Dephosphorylation of MAPKpp
  - GeneralCatalyzedReaction: Dephosphorylation of MAPKp
  - GeneralCatalyzedReaction: Phosphorylation of KKKp
  - GeneralCatalyzedReaction: Dephosphorylation of pKKKp
  - GeneralCatalyzedReaction: Phosphorylation of MAPKK
  - GeneralCatalyzedReaction: Phosphorylation of MAPKKp
- InitialConditions
- Parameters
  - ParameterSet: No Feedback Rate Constants
  - ParameterSet: Positive Feedback Rate Constants
  - ParameterSet: Oscillate Rate Constants

Start: 0 End: 500 MaxSteps: 25000

Edit Panel

Reaction Components
GeneralCatalyzedReaction: Dephosphorylation of MAP
ProductsReactant: KKK
ForwardEnzyme: KKKph
SubstratesReactant: KKKp

After entry of new value
Click 'Submit' to save value

Name: Dephosphorylation of MAPKKKp
ShortDescription: Dephosphorylation of MAPKKKp by KKKph

Parameters
ShortDescription:
Comment:
Enabled:
Name: Dephosphorylation of MAPKKKp Math
ReactionModel:
OptimizedEnzymeLevel:
KForward: 100.0
KConversion: 1.0
Lambda:
KReverse: 0.1

**Figure 1.4** Sigmoid Model Explorer showing portion of MAPK pathway: (a) TreeView of compositional hierarchy. (b) network layout visualization. (c) parameter-editing panel. (d) output plot preview panel. Along the top are various action buttons for saving and running the model, and for switching the main panel to view output plots. User can select reaction icons.

## 1.2.4 ***The Graphical User Interface: Sigmoid Model Explorer (SME)***

A component of the system to be initiated by the Sigmoid team, and that has achieved functional maturity, is the SME Web-compatible Graphical User Interface. The GUI allows the user to visualize, design, edit, and store pathway models, parameters, and initial conditions and their properties, to simulate the models by calling the simulator through the middleware, and to view and compare the properties of simulated models by viewing the temporal evolution of the concentration of chemical species under different conditions. The GUI runs from any Web browser as a Webstart or as a local client program.

Recent enhancements to SME by the Sigmoid team are as follows: (1) Model creation. There exists a new mechanism to create biological models completely from within SME and save them locally or, commit them to the database. To facilitate the construction of more complex biological processes, one-to-many mathematical reactions can be assigned to each biological reaction. Also, there are utilities to facilitate the use of web pages as source of information for data input and perform queries to the Gene Ontology database from within SME. Gene Ontology entities can either be used to tag Sigmoid objects or, instantiated directly as Sigmoid objects, i.e. Reactants or Biological reactions. (2) Numerous enhanced display features. (3) Model translation; SME can perform local translation of Sigmoid models to xCellerator code and can perform translation of SBML 1.0 to Mathematica code. (4) Model simulation. SME supports simulation through a local Mathematica license using the JLink library as well as through the remote server and there is an option to retrieve and display the output graphs for intermediate complexes generated by xCellerator/kMech reaction types. (5) Connectivity. SME now supports the Web Services Description Language (WSDL), which is an XML grammar for describing network services. Supporting WSDL expedites adoption of supplementary datasets and functionalities from other systems that support this standard.

# Chapter 2. The Sigmoid Schema

## *2.1 Introduction to the Schema and the Development Environment*

The Sigmoid schema is an object-oriented set of Java classes designed within a Unified Modeling Language (UML) editor. The use of the UML editor isn't necessarily required but it is a helpful organizational tool can produce the requisite XMI (XML Metadata Interchange) files. XMI files are a standard interchange format for the UML specification. In the Sigmoid development cycle, these XMI files are parsed by Pheno [Cheng *et al.* 2005] to produce the Sigmoid Java API, the PostGres database tables and the Object Relational Bridge (OJB) files needed to bridge Sigmoid Java programming with the relational database that Sigmoid is composed of. Information on OJB can be found at http://db.apache.org/ojb/. All of the Sigmoid schema UML development the author has performed was with the Poseidon for UML tool (available at www.gentleware.com). There are other suitable UML modeling tools. The only concern for a potential Sigmoid developer would be the requirement that the editor produces a properly formatted xmi document which can be fed to Pheno for processing.

Here we describe in detail the Sigmoid schema structure, and the logic behind it, as it evolved during the years 2004-2011. Several factors influenced the growth and evolution of the Sigmoid schema. The schema was constructed in response to contemporary modeling requirements and with intention to expand capacity for interconnectivity and interoperability with other systems. A primary mission was to maximize compatibility and support for xCellerator as a base simulation platform. This compatibility was to be achieved without limiting Sigmoid to the representational system of an interpreted notebook such as xCellerator. The Sigmoid Application Programming Interface (API) provides an object-oriented set of classes that can be utilized by modeling biologists.

While the Sigmoid schema conforms to the UML standard, there are few if any methods present in the Sigmoid schema classes. Attributes are listed in the section under the class name. In a UML class diagram, methods would normally appear in the bottom section of the class box under the attribute section. Constructing these classes with an absence of methods reflects the intentional practice of separating data from implementation. The schema classes should serve as data objects with which operations can be performed on by external operators or code.

In this dissertation, certain conventions are borrowed from Java programming and the style in which the Sigmoid components have been composed. These conventions signify computational relevance and context. In cases where a biological object or concept has a computational representation in the Sigmoid schema, it will be capitalized. For instance, if the word “model” appears it can be taken within context, if it appears as “Model” this refers to a Sigmoid Model Java class object or its corresponding database object. An additional indication of computational implications is the lack of spaces between words. i.e. “MathematicalReaction” is a mathematical reaction representation within the schema. Non-capitalized words appearing with capitalization mid-word indicate a variable or attribute name. i.e. “shortDescription” is a String attribute or variable of Model where short descriptions are stored. In this way, an attempt was made to construct these classes and attribute names with care. Capitalization thus often carries the extra meaning of being a computational representation as well as a biological one.

The Sigmoid schema has been organized into four main class diagrams. These consist of: (1) The Model class diagram. This comprises the Model class and relevant utility classes to support models. The Model class is the main representational and container class for Sigmoid pathway models. Models can be constructed without reaction kinetics strictly for network visualization. In practice however, models are constructed to produce simulateable output through xCellerator. The utility classes consist primarily of parameter set representations, classes for units and measure, and data structures designed to support external utilities. (2) The Reactant class hierarchy and supporting classes. Reactants are data objects that represent biologically relevant entities. (3) The Reaction class diagram. The diagram consists of a Reaction class hierarchy and supporting classes. Reactions are data representations for processes that transform sets of Reactants into other sets of Reactants. (4) The KnowledgeSource diagram. KnowledgeSources serve to document and cite relevant information about Sigmoid Models, Reactants and Reactions. Full diagrams are available at www.sigmoid.org.

In the class diagrams, the [*] designation specifies that multiplicity (zero to many) is enabled for the attribute at hand. Attributes lacking this designation must possess zero or one instances of the attribute.

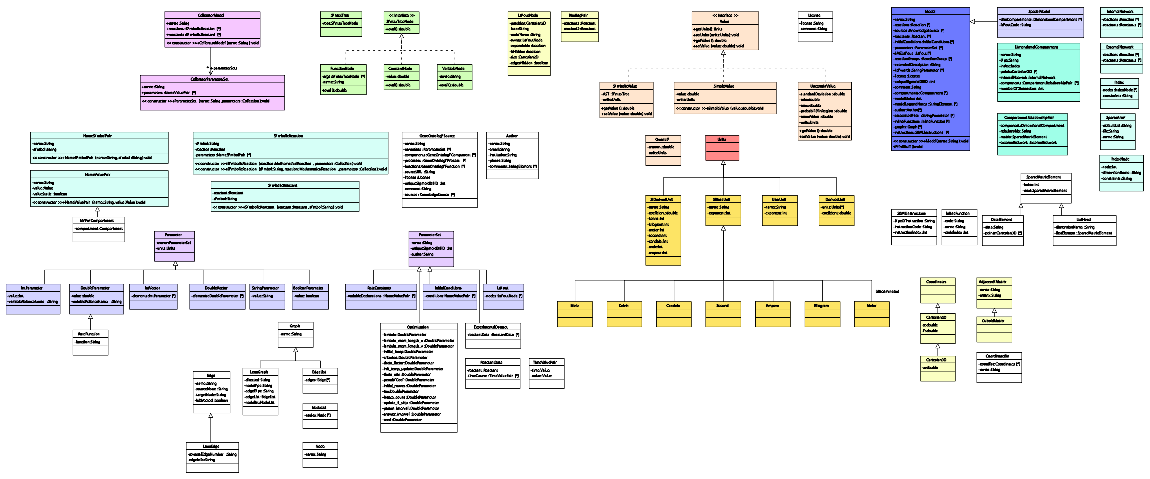


**Figure 2.1 Sigmoid Schema Model Class Diagram includes supporting utility classes such as parameter sets, parameters, classes that define units of measure, classes for graph analysis and classes for compartmental modeling.**

## *2.2 Modeling Class Diagram*

Figure 2.1 illustrates the classes contained in the Model class diagram for the Sigmoid schema. All major components will be illustrated more closely and explained in detail below. The classes contained in this diagram are primarily concerned with the top level Model class, its subclass for spatial and compartmental modeling, and supporting utility classes. The Model class and compartmental classes appear to the right of the diagram. Classes for parameter sets such as rate constants, rate equations and specie initial conditions appear on the left of the diagram. The class groups will be discussed in more detail below. Full diagrams of the Sigmoid schema are available at www.sigmoid.org.

### 2.2.1 ***The Model Class***

The Model class is the primary top level class used in producing all pathway model representations. These models can model signal transduction, biosynthetic, metabolic, biochemical or other kinds of biological processes. The primary components that will comprise a model are the Model class itself containing Reactants, Reactions, InitialConditions (reactant concentrations) and ParameterSets which contain kinetic rates, hill exponents, algebraic expressions or any relevant quantitative variable representation that occurs in a mathematical reaction equation as an input parameter. These are the bare minimum components to produce a model that is both viewable in SME and capable of being simulated by the back end Mathematica/xCellerator simulator. There are cases where kinetic rate laws and chemical species initial conditions are not available for pathways obtained from other databases or literature. In this case, it's best simply to provide SME with a dummy ParameterSet to view the pathway. Of course, simulation will not be available in that case.

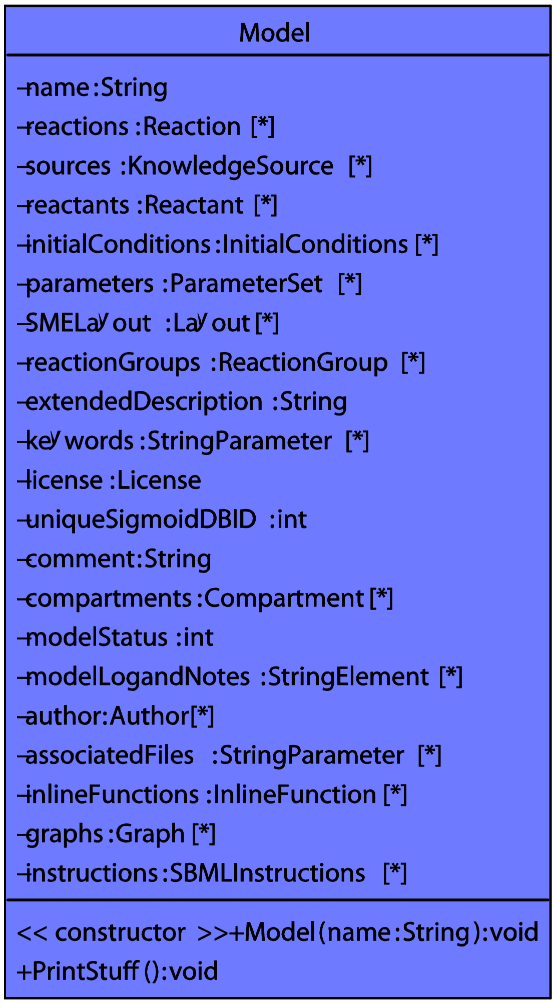

Model
-name:String
-reactions :Reaction [*]
-sources :KnowledgeSource [*]
-reactants :Reactant [*]
-initialConditions:InitialConditions [*]
-parameters :ParameterSet [*]
-SMELayout :Layout [*]
-reactionGroups :ReactionGroup [*]
-extendedDescription :String
-keywords :StringParameter [*]
-license :License
-uniqueSigmoidDBID :int
-comment:String
-compartments:Compartment [*]
-modelStatus :int
-modelLogandNotes :StringElement [*]
-author:Author[*]
-associatedFiles :StringParameter [*]
-inlineFunctions :InlineFunction [*]
-graphs :Graph [*]
-instructions:SBMLInstructions [*]
<< constructor >>+Model(name:String):void
+PrintStuff():void

**Figure 2.2 The Sigmoid Model class is the main container class for all Sigmoid biochemical pathway models. Essential components for a Model capable of display within the Sigmoid Model Explorer and simulation by the xCellerator/Mathematica simulator are: 1) A Model name. 2) A set of Reactants. 3) A set of Biological Reactions that only use Reactants specified in 2. Each Biological Reaction must have at least one associated MathematicalReaction referenced for simulation capabilities. Models that only possess a BiologicalReaction network may be visualized and edited within SME.**

The Model class contains other notable objects. KnowledgeSources are available for citation and relevant information associated with the model. Of course the Model name is what the user intends to call the pathway, and the extendedDescription attribute is for a more descriptive title. Keywords are provided for quick data searching (utilized by the Sigmoid website search) of models. The modelLogandNotes and modelStatus attributes were included for pathway model developers to keep track of modeling progress. The associatedFiles attribute provides URL references to notable documentation and most importantly the pathway model Mathematica notebooks.

Since Sigmoid relies on Mathematica/xCellerator, the inlineFunctions attribute provides a location for snippets of Mathematica code to be passed through the Sigmoid to Mathematica translator and to be included in the notebook output fed that is to the simulator. This allows raw Mathematica code to be inserted into notebook output allowing for deeper access to Mathematicas' extensive mathematical capabilities.

Likewise, "Instructions" are attributes intended for SBML support and "Graphs" are attributes intended for LEDA support which are discussed in Section 2.4.2.

## *2.3 Parameters*

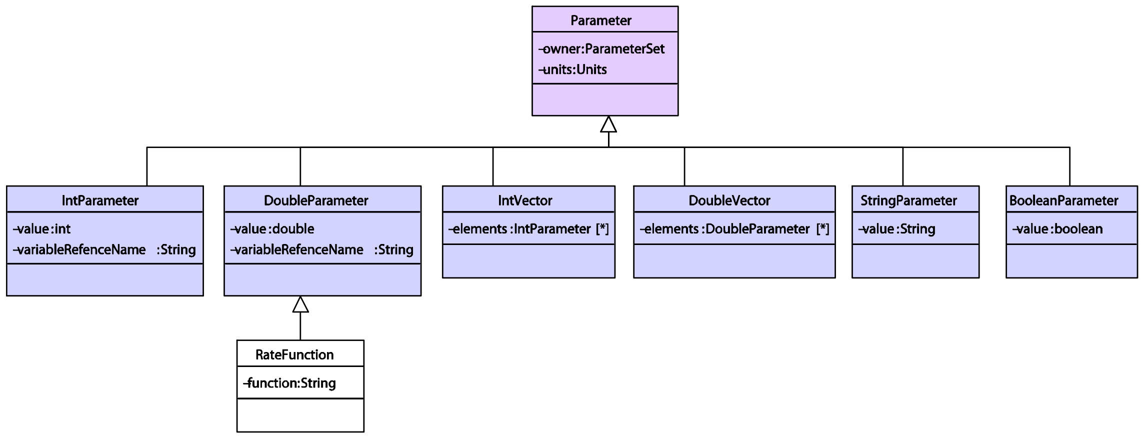


**Figure 2.3 The Sigmoid Parameter Classes store values that serve as parameter inputs for the various Sigmoid Reaction classes. Parameters store Java primitive types as data that can be used as inputs for computational or mathematical processing. RateFuntions are designed to store algebraic expressions that are to be fed to external simulators.**

Parameters are used to store kinetic rate values for reaction equations, initial condition values for reactants, Hill equation exponents, stochiometery values for reactants participating in reactions, or any reaction equation input parameter value stored in Sigmoid MathematicalReaction data classes for xCellerator compatibility. There are currently seven classes that inherit from the Parameter super-class. Every parameter possesses two key attributes. The Units attribute is a reference to (International System of Units) SI base unit classes which appear in the center of Figure 2.1 in yellow. This is done in order to provide to the user with a clear reference and meaning for the value of the parameter. It also was included in the case that the Sigmoid system were to perform any sort of dimensional analysis. The "owner" attribute references the ParameterSet in which a particular parameter belongs. By this arrangement, a parameter points to its "owner" ParameterSet.

The IntParameter and DoubleParameter classes as shown in Figure 2.3 are built to reference integer and Double value parameters respectively. They posess a string attribute "variableReferenceName" which is designed to handle variable declarations in models which implement any sort of rate equation. Models that implement AlgebraicPassthrough reactions will require this field in cases where global variables (reactant names) or constants are to be set for the notebook model.

The IntVector and DoubleVector Parameter classes have been constructed because a limitation in the OJB code implementation prohibits multiplicity from being used directly on Java primitive types. IntVector and DoubleVector serve as wrapper classes.

RateFunction is the primary mechanism by which algebraic expressions are stored, as strings, for AlgebraicPassthrough Reactions. These expressions are delivered to the xCellerator simulator. StringParameters hold strings and similarly may be of use for the construction of models that utilize algebraic rate expressions. The algebraic rate expressions are important because they provide a modeling biologist with a  mechanism to implement custom or unsupported (by xCellerator) kinetics in reaction equations thus adding calculable flexibility to the Sigmoid simulation platform.

## *2.4 Parameter Sets*

### 2.4.1 *Object-oriented Parameters*

Parameter sets, illustrated in Figure 2.4, serve as a high level single grouping of parameters for models.  This grouping allows a modeler to store different sets of parameters that may yield differing behaviors in the model.  It simplifies the modeling process for the user and facilitates organized storage of information into the Sigmoid database.  For instance, if a modeling biologist discovers an interesting oscillatory behavior in a model, and wishes to store the kinetic rate parameters for further investigation, he or she may do so without abandoning other rate information or having to tediously change individual parameters.   Sigmoid models can be constructed with any number of parameter sets to represent model data. A limitation of the SME graphical user interface prevents Models without initial conditions or rate constants from being rendered [Su 2004].   Furthermore, models lacking kinetic rate constants or parameters will not translate to simulateable models for xCellerator.

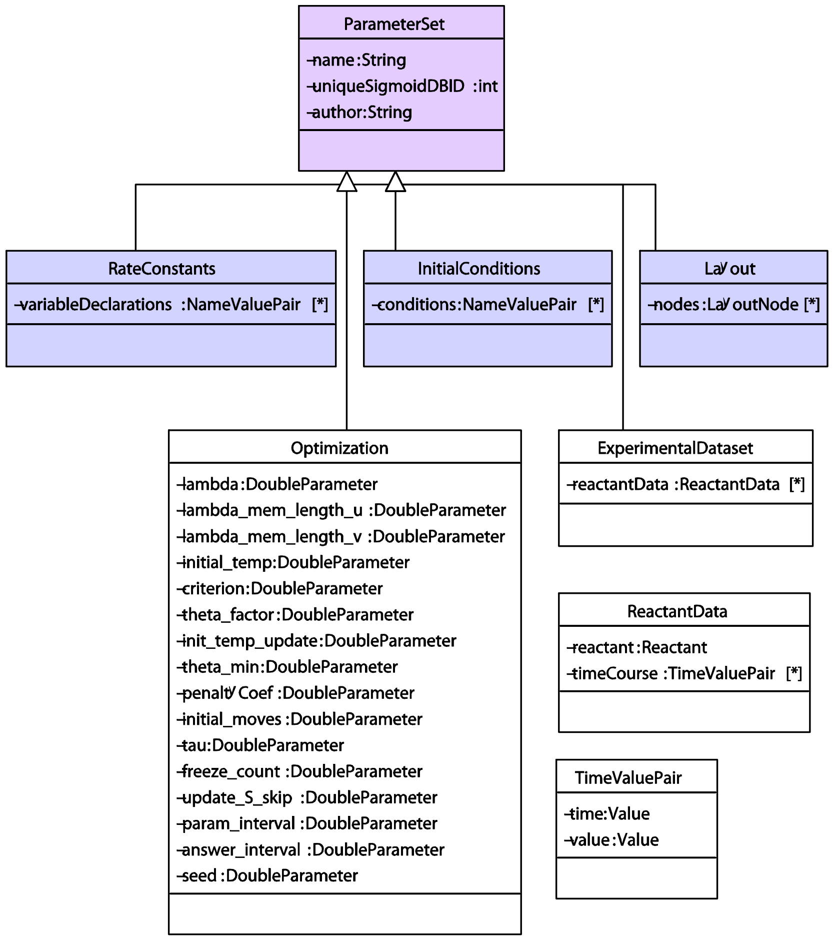

ParameterSet
-name:String
-uniqueSigmoidDBID :int
-author:String
RateConstants
-variableDeclarations :NameValuePair [*]
InitialConditions
-conditions:NameValuePair [*]
LaYout
-nodes:LaYoutNode [*]
Optimization
-lambda:DoubleParameter
-lambda_mem_length_u :DoubleParameter
-lambda_mem_length_v :DoubleParameter
-initial_temp:DoubleParameter
-criterion:DoubleParameter
-theta_factor:DoubleParameter
-init_temp_update:DoubleParameter
-theta_min:DoubleParameter
-penaltYCoef :DoubleParameter
-initial_moves :DoubleParameter
-tau:DoubleParameter
-freeze_count :DoubleParameter
-update_S_skip :DoubleParameter
-param_interval :DoubleParameter
-answer_interval :DoubleParameter
-seed :DoubleParameter
ExperimentalDataset
-reactantData :ReactantData [*]
ReactantData
-reactant:Reactant
-timeCourse :TimeValuePair [*]
TimeValuePair
-time:Value
-value:Value

**Figure 2.4 ParameterSets are collections of Sigmoid Parameters that serve as inputs for the variety of Sigmoid Reaction classes, concentration information Reactants and parameterized inputs for exernal simulators and analysis tools. Collections of parameters are useful for organizing model behavior and analysis at a meta level. Also, the Optimization ParameterSet stores parameters for a simulated annealing optimizer [Zhang 2008] that has been incorporated into Sigmoid.**

An important attribute to note is the ParameterSet name. This attribute must be set both in the ParameterSet and by reference within any corresponding reaction rate or InitialCondition. (Failure to do so will prevent SME and the ModelEmitter from functioning properly.) Reaction equation parameters are attributes of MathematicalReaction class or data objects. Each particular parameter possesses a corresponding "owner" ParameterSet thus being contained by external reference.

#### *2.4.1.1 Rate Constants*

In the Sigmoid schema, MathematicalReactions (see Section 2.9) are classes that represent and store the reaction kinetics that simulate biological processes. Each MathematicalReaction has a set of attributes that consist of reaction input parameters that are sent through the middleware translator and converted to xCellerator functions for simulation. The MathematicalReaction attributes are stored as Parameters and these Parameters refer, via their "owner" attribute, back to their ParameterSet. In this case the ParameterSets are RateConstants. The naming of "RateConstants" is slightly misleading and should be changed to better reflect what RateConstants stores. "Reaction Parameters" would probably be a better name. The variableDeclarations attribute is useful when constructing models that use eitherAlgebraicPassthrough reactions, inlineFunctions or some cases of SBML support.

#### *2.4.1.2 Initial Conditions*

InitialConditions refer to the initial concentrations of Reactants in a Sigmoid model which serve as starting points for simulation runs. Any model that is to be displayed by SME must possess this collection. Failing to provide this will result in a display error. Upon simulation, any uninitialized reactant concentrations will most likely be set to zero by xCellerator.

### *2.4.1.3 Layout Nodes*

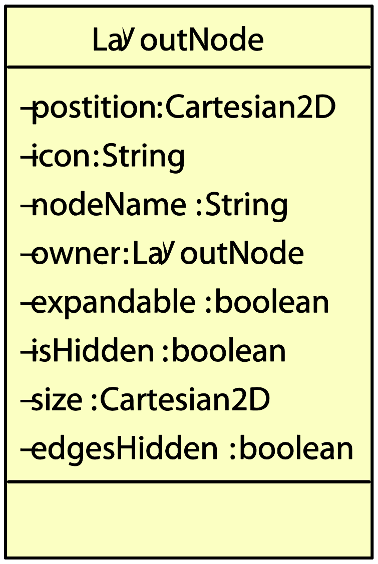


**Figure 2.5 LayoutNode Class is used for storing graphical layout settings that determine the Sigmoid Model Explorer display behavior. Model icons are positioned via Cartesian coordinates. Layouts were designed to be collapsible and thus can be stored in a nested fashion by using the "owner" reference. Toggles for hiding particular icons and associations to other icons are stored via the "isHidden" and "edgesHidden" attributes.**

The Layout ParameterSet was included to allow multiple graphical layout representations to be shown by SME. For reference the LayoutNode class is shown in Figure 2.5. Layouts use Cartesian coordinates to position Sigmoid Icons. The "expandable" attribute indicates that a node can be expanded into a sub-network of reactions or collapsed and represented by one icon. The expansion/contraction is coordinated with the ReactionGroup class which is comprised of collections of reactions. The idea behind this is to be able to treat a biological processes comprised of many subreactions as a unit. For instance, DNA to RNA translation should require a whole network of reactions to achieve, but a modeler may wish to view the whole process as a signal-in product-out node in one instance, and in another may wish to examine the whole sub-network as it relates to other processes. ReactionGroups are explained in more detail in Section 2.9.2.2. The “isHidden” attribute is used to hide nodes (icons) from layout view. The “edgesHidden” attribute does the same for the SME/JGraph arrows that point to, or from, the icon. The ‘size’ attribute can be used to scale Sigmoid icons.

A Simulated Annealing Optimizer (SAO) [Zhang 2008] has been integrated into Sigmoid through the web services interface. It uses a global optimization technique and Lam- Delosme schedule to make the optimization process faster and more efficient when compared with other general schedules available [Lam and Delosme1988]. It aims to reverse engineer model parameters (for example kinetic rate constants) given both the model structure (represented as ordinary differential equations) and empirical system dynamics as expressed by experimental time series data.[Compani *et al.* 2010]

The Optimization ParameterSet, ExperimentalDataset, ReactantData and TimeValuePair classes are built to support the storage of the SAO parameters in the case that an expert user wishes to do multiple optimization runs with differing parameter settings or to simply save their optimization settings with a corresponding model.

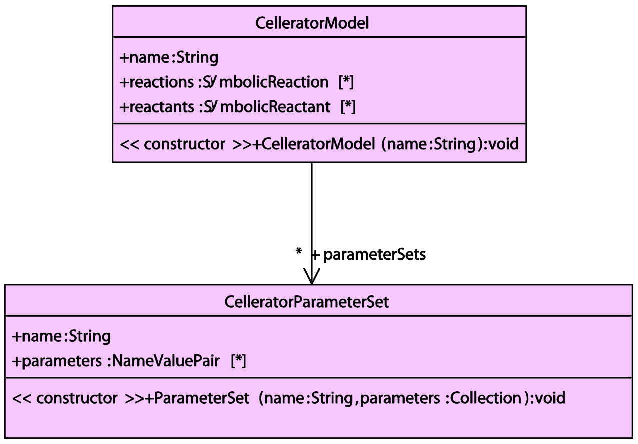

CelleratorModel
+name:String
+reactions :SYmbolicReaction [*]
+reactants :SYmbolicReactant [*]
<< constructor >>+CelleratorModel (name:String):void
* + parameterSets
CelleratorParameterSet
+name:String
+parameters :NameValuePair [*]
<< constructor >>+ParameterSet (name:String,parameters :Collection):void

**Figure 2.6 Cellerator ParameterSets are a legacy representation from earlier versions of Sigmoid. They are still a useful representation for initializing models that have corresponding Mathematica/xCellerator notebooks.**

Sigmoid was developed with the idea of having an external simulation engine. The Cellerator, now xCellerator, package for Mathematica provides a powerful set of functions to model and simulate biochemical reaction networks. Earlier versions of Sigmoid (Pre Sigmoid 2.0) used a parameter representation that directly mirrored Cellerator's structure. Parameters were stored as NameValuePairs (Figure 2.7), just as Cellerator notebooks commonly set a name for a variable and a corresponding initial value. The need to implement multiple parameter sets as well as the requirement to adhere to a more object-oriented design for parameter representation led to the abandonment of the use of CelleratorParameterSets within SME and Sigmoid. CelleratorParameterSets (CPS) are still useful however, during construction of hand coded models from Cellerator Notebooks. In this case it's useful to build a CPS from the notebook data and then run a translator method to build the Sigmoid model. This point will be discussed further in Chapter 4.

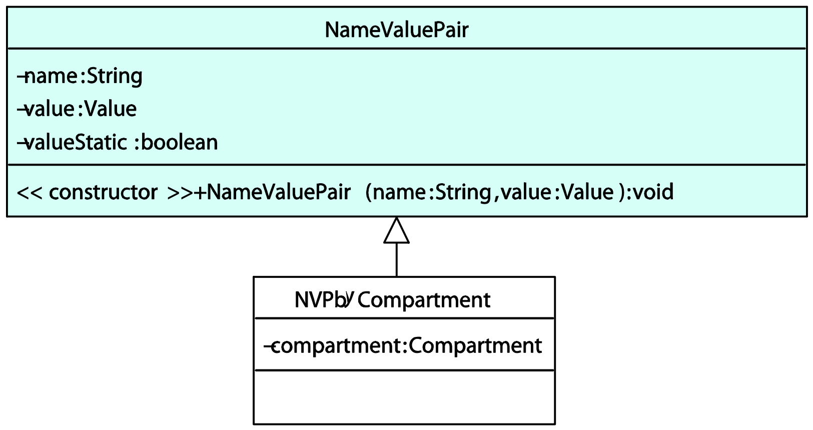


**Figure 2.7 NameValuePairs serve as a mirror representation of xCellerator notebook rules used to set initial conditions and rate constants for models.**

The NameValuePair (NVP) class is still a valuable class in the construction of hand coded Sigmoid Models because it closely mirrors the structure of parameter initialization in xCellerator notebooks. Although the NVP is not required for a functioning Sigmoid model, it provides the user with a useful handle and reference to existing Mathematica/xCellerator notebooks when building model code. The "valueStatic" attribute can signal to the reaction translator that a value is to remain a constant.

The NVPbyCompartment class (NVP meaning Name Value Pair) was designed to store NVPs for compartmental models designed for the Sigmoid schema version 2.17. The version distinction is required because there are two differing representations in the schema for compartmental modeling. Any class that refers to a Compartment will correspond to a representation built for the 2.17 version. References to SpatialModels and DimensionalCompartments were added to Sigmoid schema as of version 2.21. The representation for compartments spatial models will be discussed in Section 2.11.2.

### 2.4.2 *LEDA*

Since Sigmoid was designed to be a database of biochemical reaction networks, at some point the database could become heavily populated. The array of available Models, Reactions and Reactants would overlap in scope and become interconnected. At this point, the opportunity to perform graph analysis upon the vast biological networks represented within the Sigmoid framework could yield valuable inferences. A likely candidate for graph analysis integration with Sigmoid is GraphCrunch [Milenković 2008]. Therefore, a set of data structure support classes was built for LEDA formatted graphs which scale up to large problem sizes. A LEDA graph consists of an EdgeList and a NodeList with corresponding "type" descriptors. The graph may be directed or undirected so Edges possess "sourceNodes " and "targetNodes" as well as a Boolean toggle "isDirected".

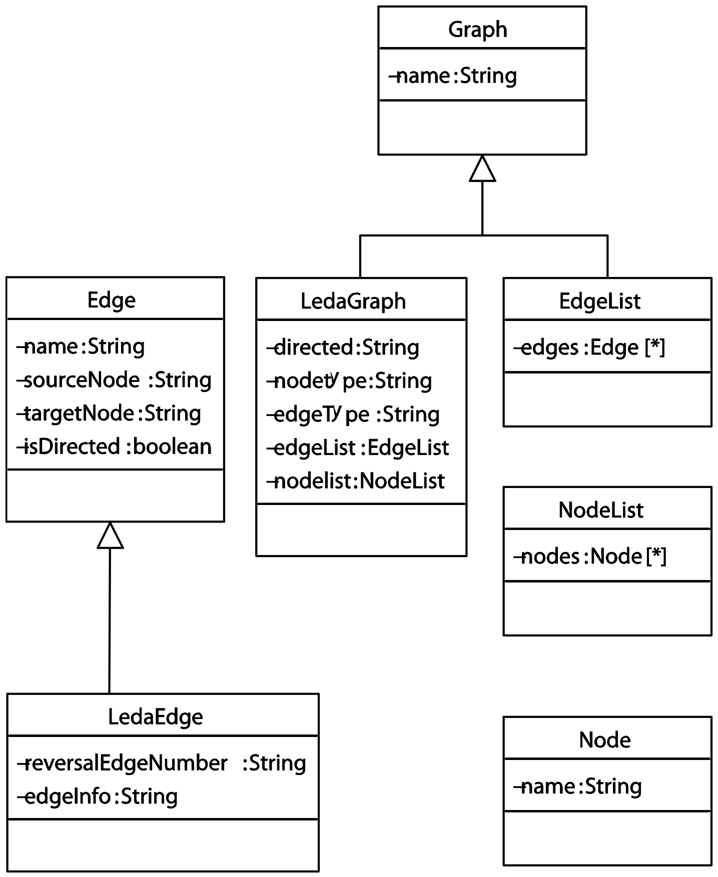


**Figure 2.8 The LEDA Graph data structure classes.**

## *2.5 The Reactant Hierarchy*

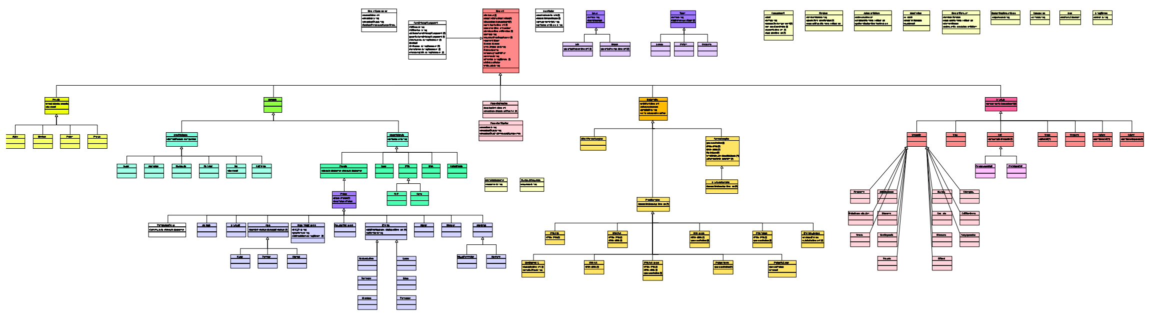


**Figure 2.9 The entire Reactant class diagram. The root node of the main tree of reactants is The Reactant class (red). The main sub categories of Reactants are from left to right, Particles (in yellow), Molecules (green), Proteins (purple), BioComplexes (orange) and Structures (reds and pinks). This hierarchy is expanded upon in more detail in subsequent figures.**

The Reactant class diagram (Figure 2.9) for Sigmoid consists of a top level superclass Reactant (in red) with a corresponding hierarchy of reactant subclasses and a dozen or so supporting classes. We will define the reactant hierarchy as the Reactant superclass and all classes that inherit from Reactant. The tree shown in Figure 2.9 indicates an "is a" relationship ie. an Enzyme "is a" Protein, a Protein "is a" Macromolecule, and so forth where all classes below inherit the properties of the class above it, ultimately all being Reactants. This hierarchy serves at least two main purposes. First it produces an object-oriented representation of Reactant class objects to serve the software engineering requirements for the Sigmoid modeling system. Secondly, the hierarchy serves as a special-purpose ontology to represent relevant biological entities as objects within the Sigmoid framework. The reactant hierarchy should be comprised of a wide domain of frequently referenced biological objects that participate in biological reaction processes. These reactants range greatly in scope in scale, composition and complexity.

An attempt was made to provide a distinguishable set of reactant objects that would both represent at least basic biological entities for modeling biologists and also interact properly with the software components of the system. In order to reduce coding overhead, SME uses Java reflection to discover properties of the objects it wishes to display. Java reflection discovers the class of its objects at runtime, to perform operations at runtime. This technique and level of abstraction allows SME to avoid hard coding behaviors based upon specific object types. This approach from the SME client side was to parse the Reactant objects with reflection and to make display determinations based upon a Reactants' class. For instance, rules could be set to display SmallMolecules with a particular icon (a graphical file image that represents a node in the display network of SME) shape, color or size, and MacroMolecules with a different shape etc. Also, SME makes determinations about display based on sub-class type.

For the most part, on a macroscopic level the objects denoted in the reactant hierarchy diagram increase in scale and complexity from left to right in Figure 2.9. The idea was to provide common biologically relevant objects for the biologist to utilize in constructing a model. From left to right the major groups are Particles, SmallMolecules, MacroMolecules, Complexes and on the far right Organelles, Cells, CellularStructures and Organisms.

## 2.5.1 *Particles*

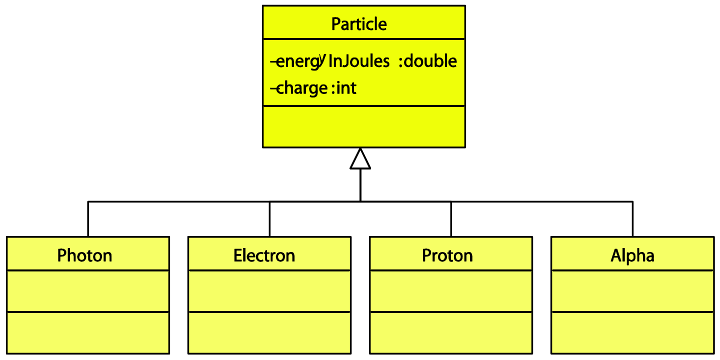


**Figure 2.10 Basic particle types are available for modeling. Photons are available for modeling of systems involving biological processes such as photosynthesis. Electrons were included for modeling electromotive forces involved in mitochondrial ATP synthesis. Protons are crucial for modeling acid-base chemistry and membrane bound proton pumps. Alpha particles and Photons could be useful in modeling cell damage induced by these forms of radiation.**

The Particle class, and its subclasses, serve to cover the low end of the modeling scale for Sigmoid as described in Figure 2.10. Particles inherit from Reactants (not shown on the diagram.) Energy and charge attributes are available for particles. The particle classes arose as a modeling request at one point and should serve as a rough template for further development if necessary. Of the various particle types, Photons are available for modeling of systems involving biological processes such as photosynthesis. Electrons were included for modeling electro motive forces involved in mitochondrial ATP synthesis. Protons are crucial for modeling acid-base chemistry and membrane bound proton pumps. Alpha particles and Photons could be useful in modeling cell damage induced by these forms of radiation.

### 2.5.2 *Molecules*

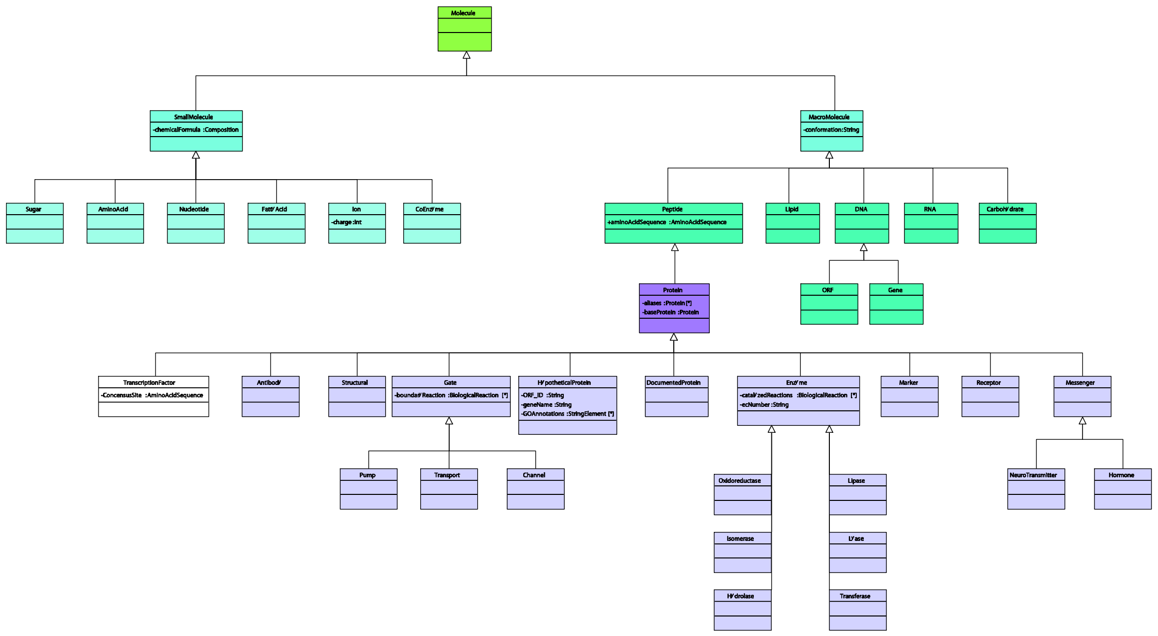


**Figure 2.11 The Molecule Sub-hierarchy of Reactants. At the molecule level, the reactant tree is split into two categories, SmallMolecules (left branch) and MacroMolecules (right branch). The MacroMolecule domain includes all types of Sigmoid Protein (Purple subtree). Within the scope of existing and curated Sigmoid models, the molecule tree possesses the most frequently instantiated types of Sigmoid Reactants.**

Molecule is a class that inherits from Reactant in the hierarchy and possesses a vast number of subclass types arranged into relevant biological groupings. The major groups fall under SmallMolecules, MacroMolecules and Proteins. These classes of molecules were chosen because of their biological significance. This significance is recognized for many of the classes by other markup languages like KEGG and SBML. The intent was to provide a class array that would facilitate automated conversion of information from other databases and markup languages to the object-oriented representation present in Sigmoid.

### 2.5.3 *SmallMolecules*

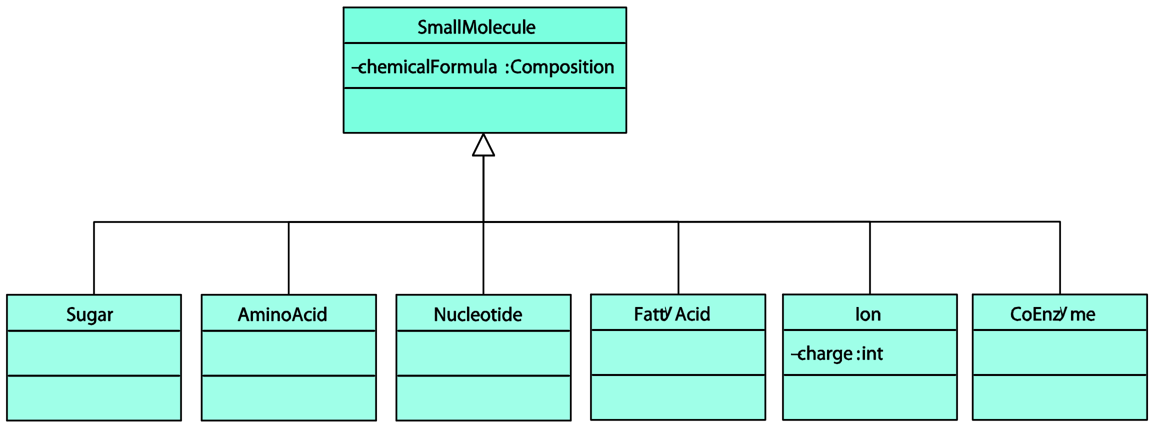


**Figure 2.12 Several common kinds of biological SmallMolecules of are available for modeling. SmallMolecules possess the chemicalFormula designation. The particular convention of molecular formula is unspecified for user flexibility. Charged Ions can be used for use cases such as acid-base reactions.**

Small molecules arise frequently in biochemical and pathway models. A basic array of biological types is provided: Sugars, AminoAcids, Nucleotides, Fatty Acids, Ions, and a spot for CoEnzymes occur in the schema. All these classes inherit a Composistion from SmallMolecule which basically stores a string for empirical, molecular or structural formulas. No particular form is enforced by Sigmoid and this convention is left to the expert user.

### 2.5.4 *MacroMolecules*

Basic support for common biological macromolecules such as DNA, RNA, Lipids, Carbohydrates and Proteins is provided by the classes shown in Figure 2.13.

(Although there are some lipid polymers such as Polyketides, Lipids should be relocated to the SmallMolecule tree.  Also, The Carbohydrate class should be re-termed Polysaccharide.) The "conformation" attribute is present to give the user some flexibility in storing relevant macromolecule information.  Its primitive type was left as a string for user flexibility until domain specific modeling requests or modeling plans determine specialized attributes.  Relevant sequence or domain specific molecule information could be placed here.

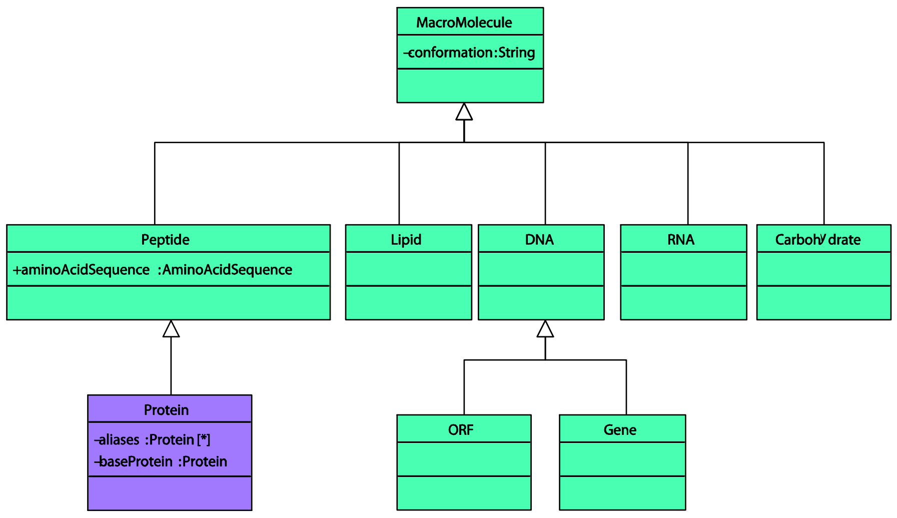


**Figure 2.13 The Sigmoid MacroMolecule classes represent common but pertinent classes of biological macromolecules.  Appearing from left to right are, Peptides which have the Protein subclass tree, Lipids, DNA, RNA, and Carbohydrates (Should be termed Polysaccharide).  The "conformation" attribute of Macromolecule can be used to store domain specific information about a particular macromolecule such as sequence information.**

### 2.5.4.1 *Proteins*

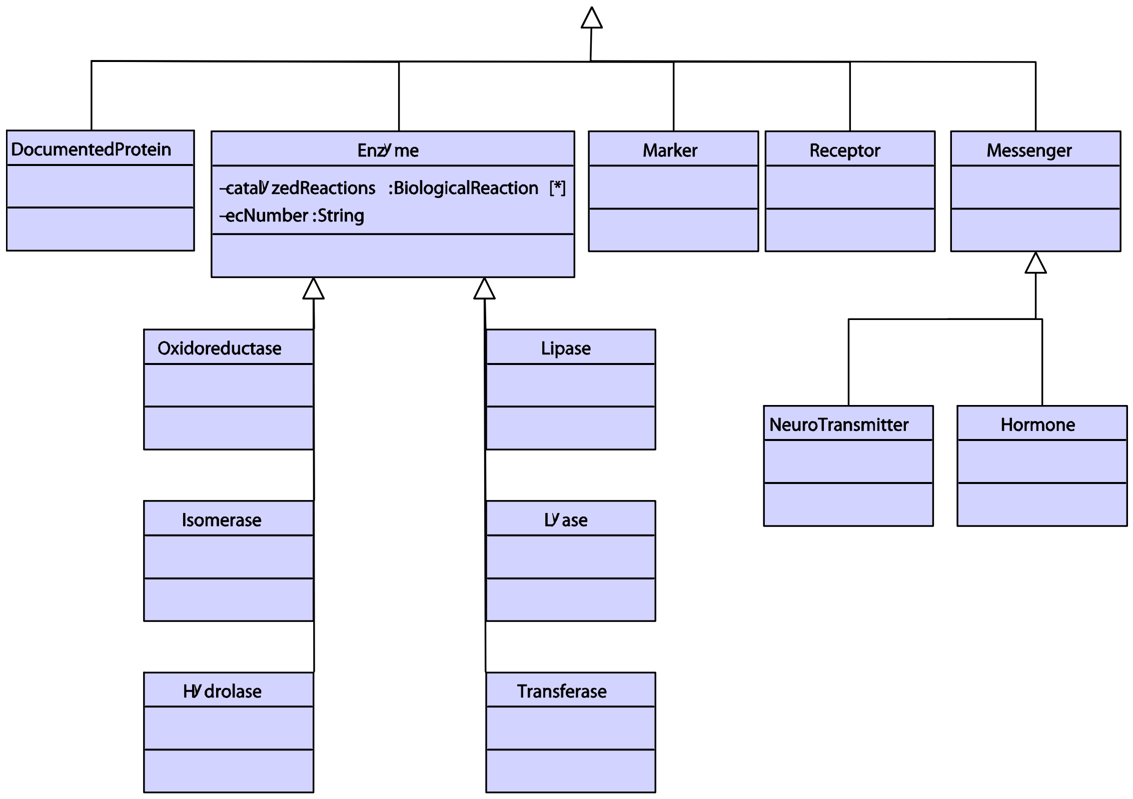


**Figure 2.14 This class diagram presents some of the Classes that inherit the Protein designation. The six major types of enzymatic proteins are available as an Enzyme Reactant. Other Protein class names are designated by roles the protein might be playing in a particular Model or Reaction. Providing classes for these types of proteins allows a coding investigator to build models using objects that represent relevant biological entities. The representation can be expanded, with a schema revision, to include additional attributes when a modeling demand arises. For example, Odixoreductases have associated coenzymes or cofactors. Naturally if analysis of this relationship were to become a modeling requirement, it could be added to the schema.**

Several protein sub-classes are available to choose from. The Protein class names correspond to common protein types and roles. Markers and Receptors would indicate surface proteins, Messengers would be used for intercellular functions. Six basic types of enzyme as categorized by activity (Oxidoreductase, Isomerase, Hydrolase, Lipase, Lyase and Transferase) are present and are marked by “ecNumber” to represent the Enzyme Commission number (EC number), which is the numerical classification scheme for enzymes. Enzymes also inherit a reference to potentially many BiologicalReactions.

Since the EC number is a reference to enzyme-catalyzed reactions, the idea is to store the reactions in the “catalyzedReactions” attribute for an enzyme. GateProtein and its three subclasses are available to represent trans-membrane types of proteins involved in potentially multi-compartment models, and therefore possess a reference to a set of associated “boundaryReaction” trans-membrane reactions. HypotheticalProtein was included for potential gene products and has fields for the open reading frame and GO annotations. Structural and Anitibody protein class designations are available for cases where that role must be indicated.

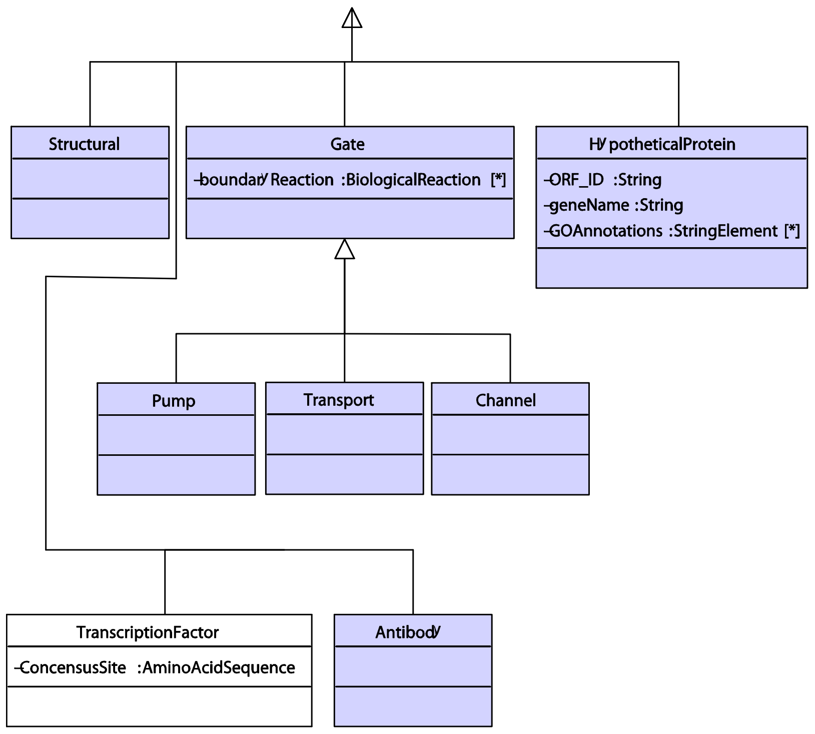


**Figure 2.15 Additional Protein class types include Gate Proteins that are involved in trans-membrane or boundary Reactions, roles as Structural proteins and Antibodies. HypotheticalProteins are available to represent potential gene products and TranscriptionFractors can specify sequence specificity.**

### 2.5.5 ***Complexes***

Moving up in scale, next is the BioComplex group of Reactants. BioComplexes have been designed to represent biological complexes, mainly of macro-molecules. The BioComplex class has an “entryPoint” Reactant specified to be the root of a tree or graph structure. This was included mainly to provide a focal point for the user. The “collapsed” attribute is a logical operator included for display purposes in SME. Since complexes under this representation have associated Reactions, the design intent was to include options for expanding or collapsing the reaction network display for a complex. “complexID” was and identifier similar to “uniqueDBID”. Also, the “formationLogic” SyntaxTree provides a handle to include a formal grammar for the formation of potential complexes [Mjolsness 2007].

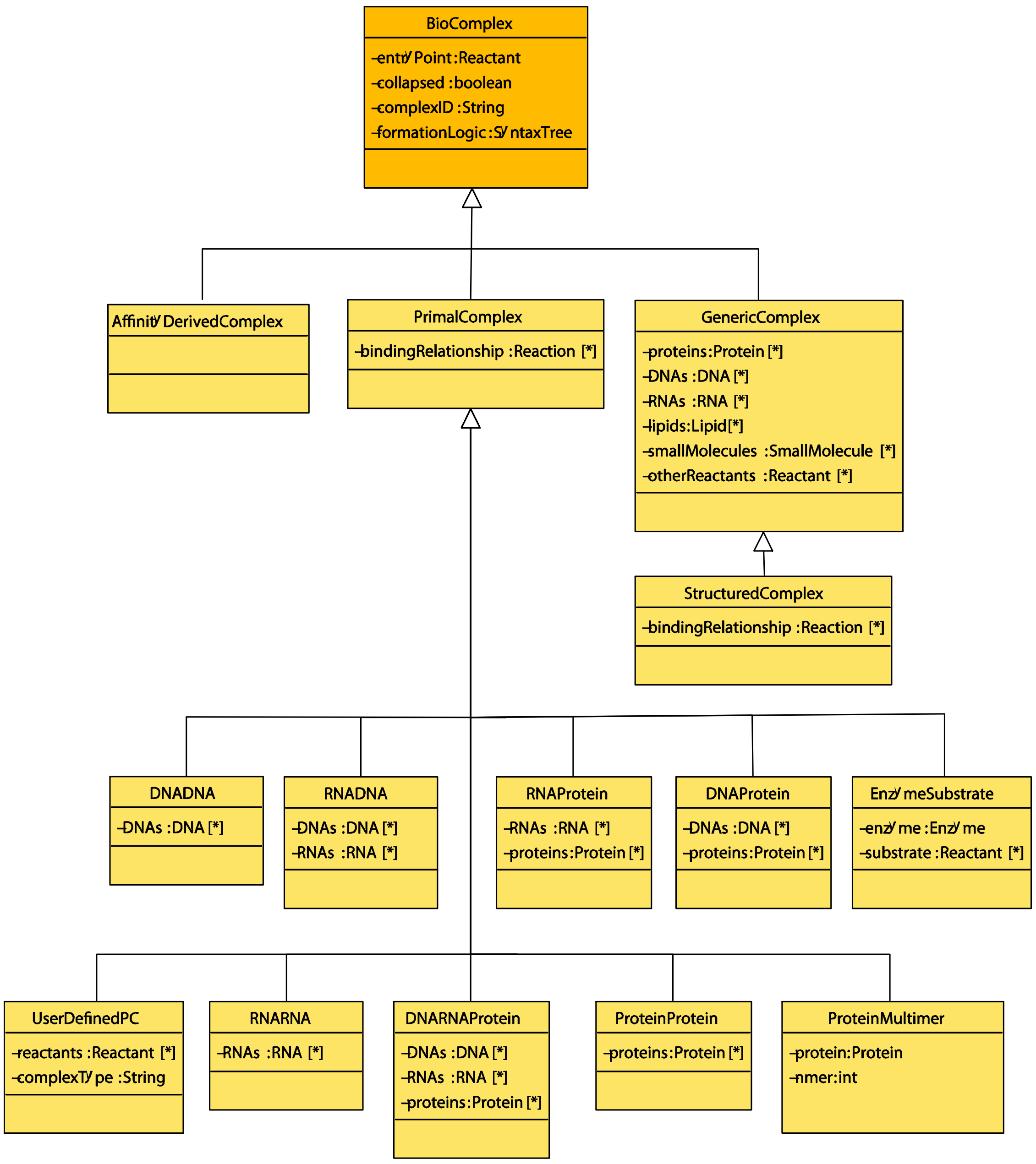


**Figure 2.16 The BioComplex sub-hierarchy of classes.**

The BioComplex sub-hierarchy contains a couple branches of note. There are several PrimalComplex subclasses built so that the class name describes the complexes' respective components. The other branch, GenericComplex was designed as a container class for assorted MacroMolecues. The "otherReactants" attribute and were included in case the modeling biologist wises to use another reactant type in a complex. The intent behind the GenericComplex was to provide a data class that would serve as a bin for software processing and traversal of a complex network. StructuredComplex, inheriting from GenericComplex contains its respective Reaction vector.

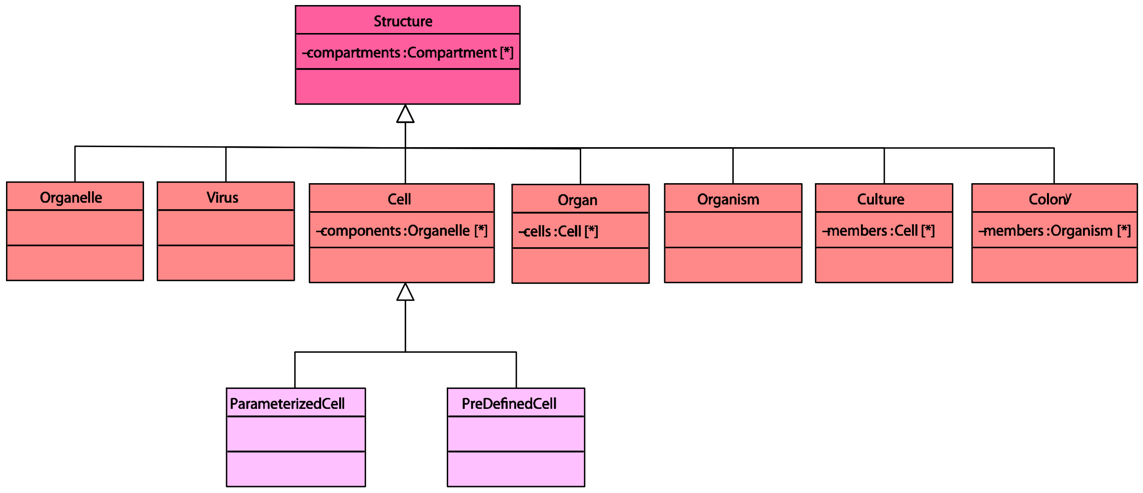


**Figure 2.17 Structures are intended support large scale reactants with high complexity. Structures can contain other Reactants thus provide a nested representation for biologically complex entities. Every structure contains a collection of compartments. For example, this can represent multi-compartmental Organs with cellular constituents that possess organelles.**

### 2.5.6 *Structured Reactants*

To the far right of the Reaction hierarchy are the Structured Reactants. Structured Reactants are provided for cases where the complexity of a reactant exceeds macromolecules or complexes. In left to right order of increasing complexity such reactants include Organelles, Viruses Cells, Organs, Organisms and Colonies. Tissues should be added to this diagram as they participate in developmental models. Representational nesting of structures was included. For instance, a multi-compartmental Organ, containing multi-compartmental Cells could contain multi-compartmental

Organelles.

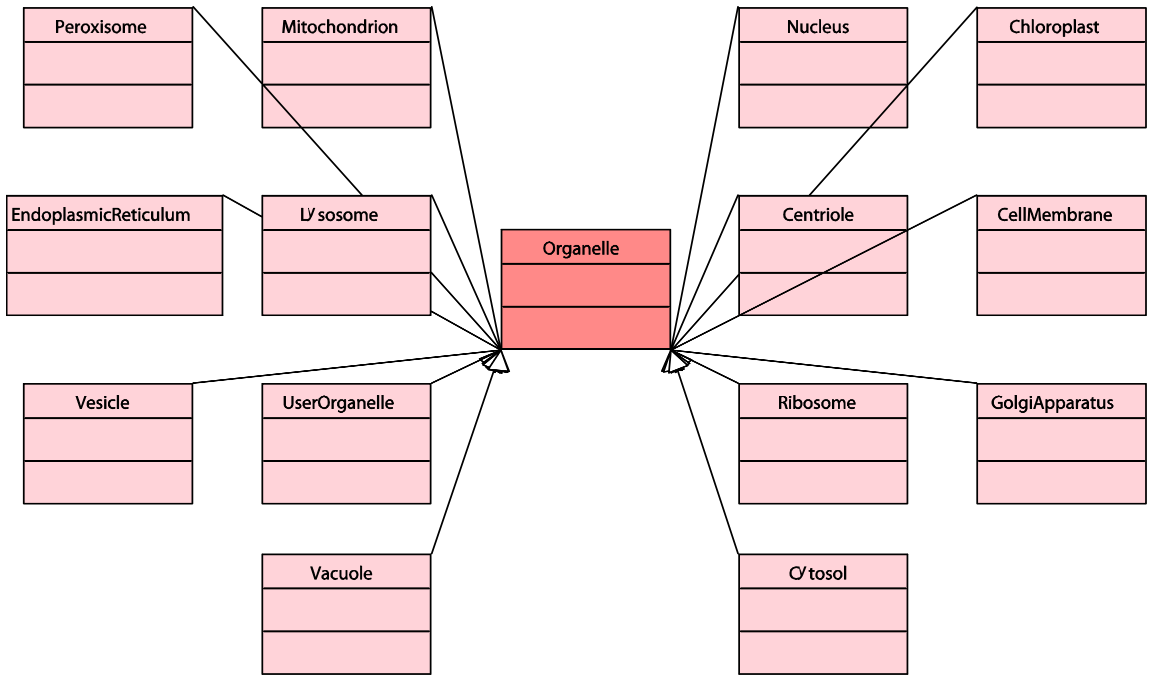


**Figure 2.18 The available Organelle classes inherit from Structure and possess collections of compartments. A set of classic cellular structures is available for the modeling biologist to choose from to construct Models.**

Several class types for Organelles are illustrated in Figure 2.18. These serve as templates for later expansion of the ontology.

Many classes in this hierarchy are lacking class specific attributes which, from an object-oriented perspective is perhaps an inefficient use of a class structure and tables in the database. The long-term intention was to add relevant attributes to the classes as need emerged from modeling requirements and as the corresponding ontology evolved. The author would classify a large amount of the ontology as mental place holders that still provide a decent handle when constructing new models as well as a coding convenience. Java code authors can simply instantiate a common biological component by name. This is fairly important, as typically building new models yields a new modeling requirement for the Sigmoid system. For instance, a user hand coding a Sigmoid model that implements Transport, Pump, or Channel Proteins may have a domain-specific requirement for the representation in the form of class attributes or additional supporting classes. At that time, the new class requests can be considered for Sigmoid representation and added to the schema if necessary. Having a basic Reactant class (for this example, the Transport, Pump or Channel Protein) to serve as a handle, at least allows a framework model to be encoded and tested until a schema revision, with the deeper representation, is available.

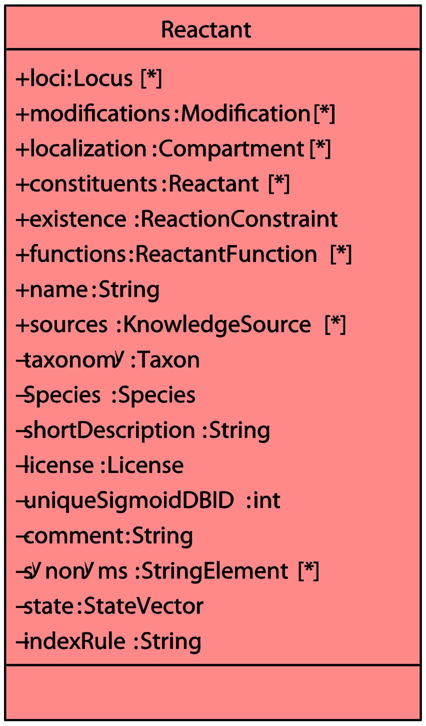

Reactant
+loci:Locus [*]
+modifications:Modification [*]
+localization :Compartment [*]
+constituents :Reactant [*]
+existence :ReactionConstraint
+functions:ReactantFunction [*]
+name:String
+sources :KnowledgeSource [*]
-taxonomy :Taxon
-Species :Species
-shortDescription :String
-license :License
-uniqueSigmoidDBID :int
-comment:String
-synonyms :StringElement [*]
-state:StateVector
-indexRule :String

**Figure 2.19 The Reactant super class is the container class for all common Reactant attributes. It is abstract in that its biological role has not been explicitly designated. Functionally at a minimum for xCellerator simulation the only required field for a Reactant is a "name". A corresponding initial condition should be specified.**

Furthermore, a substantial number of Reactant class types have designations in other biological markup languages such as Kyoto Encyclopedia of Genes and Genomes

(KEGG) Markup Language, (KGML) or the Systems Biology Markup Language (SBML). A reservoir of compatible Sigmoid Reactant classes facilitates automated translation of models from the other markup languages to the Sigmoid database.

The Reactant class serves as the top level designation in the hierarchy and all other reactant classes inherit its attributes. Reactants all should have basic attribute identifiers for use within the Sigmoid software system. SME and the translation to xCellerator notebook require a "name" field in order to function. Names should not contain any spaces or special characters that Mathematica/xCellerator cannot parse. A "shortDescription" serves for the purposes of pop-up text within SME and can also be a more descriptive name field. ShortDescriptions are not parsed by the simulator and are not subject to restrictions. Additional user notes or information can be stored in the "comment" field. The license field was included in case there were any licensing issues with model data. "Synonyms" were included for cases where something might have many names. The need for a unique identifier, "uniqueSigmoidDBID", arose because ultimately Sigmoid is to store relevant reactant and reaction data without redundancy. For instance the ATP molecule possesses its own unique attributes and should be stored in such a way that it can be used across all potential pathway models it may occur within.

With regard to the "constituents" field, Reactants were built to have references for recursive self containment in the event the modeler wanted to model one reactant as a conglomeration of reactants. Reactants such as organelles or complexes may possess several distinct Reactant components.

The "state" attribute of StateVectors are an important representation for macromolecules, as discussed in the next section.

## 2.6 *Reactant Utility Classes*

### 2.6.1 *StateVectors and Modifications*

The Reactant hierarchy has a few utility classes constructed to allow Reactants the ability to represent several states. This is particularly important for macromolecule type reactants. For instance, proteins may undergo post-translational modifications such as phosphorilation, ubiquitination, methylation, etc. (See Section 2.9.3 for a much more extensive list.) The idea is to store a StateVector for a particular Reactant that would contain a vector of potential modification sites and bindingRegions for potential complex formation binding partners. Each Site would have references to particular reactants that could modify the site. Regions would reference those Reactants which might potentially interact or complex with a Reactant within a binding region. Both Site and Region inherit from Locus and possess the Boolean "bound" attribute intended for the software system to record binding state (bound or unbound). Reactants also possess Modifications which specify the Site of a modification and the modifying molecule type.

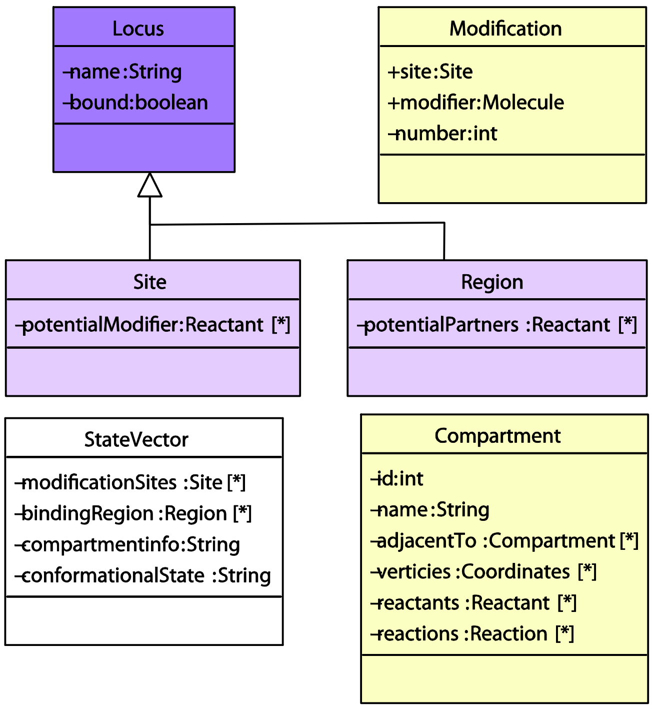


**Figure 2.20 StateVectors, Modifications and the Sigmoid 2.17 Compartment representation.**

### 2.6.2 *Compartments*

A Compartment class was also provided in the 2.17 schema to support multi-compartmental models. The class is shown here because it was associated with the StateVector class. Consideration was given to specifying a Reactants location by compartment within the StateVector, but this approach proves problematic because the presence of a Reactant within a compartment can be more properly modeled as a property of the compartment as opposed to being a property of the Reactant. The Compartment class should be deprecated, since a more feasible approach is introduced in Section 2.11.2 on spatial and compartmental modeling.

### 2.6.3 *Taxonomy*

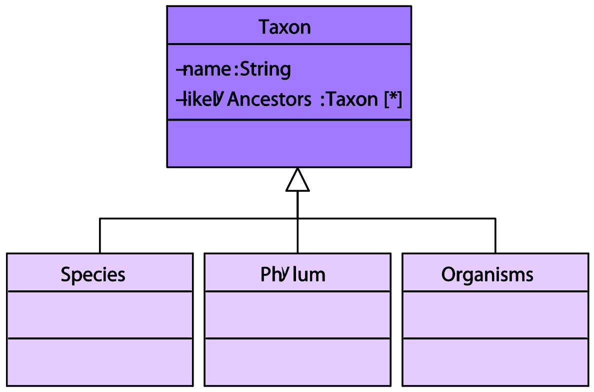


**Figure 2.21 A simple Taxon representation. Taxonomic ancestor information can be stored in this simple recursive data structure. The reference to Taxon is contained in the root Reactant class. Although most entities in the Reactant hierarchy may have a taxonomic designation, a few exceptions are present. i.e. particles and small molecules do not inherently possess this attribute.**

The Reactant class includes a taxonomy attribute mainly included for taxonomic references for proteins. The Taxon Class has a simple recursive inheritance structure to indicate ancestors. It's recommended to either modify, or remove this from the schema for a couple reasons. Reactants like a particle or small molecule will not possess taxonomy so, the Taxon attribute inherited from Reactant would be inapplicable. Also, the Gene Ontology classes provide a more robust representation that includes taxonomy identifier numbers for this function. The Gene Ontology classes are discussed in more detail in Section 2.11.1.

### 2.6.4 ***Functional Ontology***

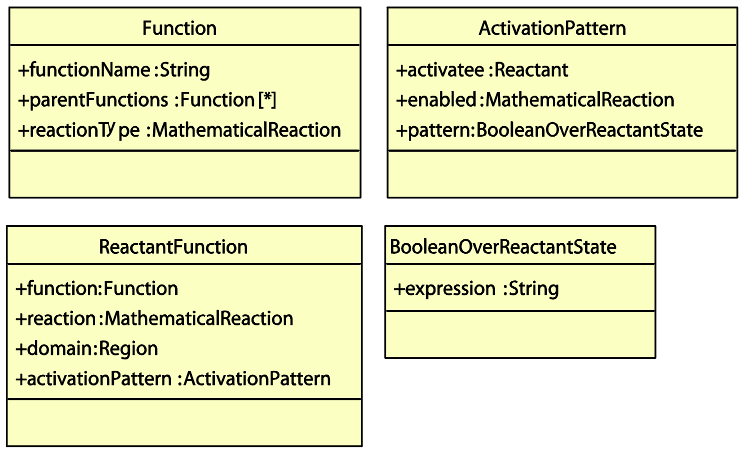

Function
+functionName :String
+parentFunctions :Function [*]
+reactionType :MathematicalReaction
ActivationPattern
+activatee :Reactant
+enabled :MathematicalReaction
+pattern:BooleanOverReactantState
ReactantFunction
+function:Function
+reaction :MathematicalReaction
+domain:Region
+activationPattern :ActivationPattern
BooleanOverReactantState
+expression :String

**Figure 2.22 The Sigmoid functional ontology.**

Certain classes were built to represent a functional ontology by a previous Sigmoid developer. It appears as though the idea behind this was to provide a mechanism by which a modeler could specify a reference to particular Reactions that a reactant was involved in or would be activated by, thus indicating its function. These classes were never used in any models and could possibly be better represented the by the GeneOntology (GO) classes in the schema along with a set of associated MathematicalReactions. One argument in support of maintaining these classes and that distinguish it from a gene ontology representation is that these classes build a graph of MathematicalReaction relationships. Function, ActivationPattern, ReactantFunction and

BooleanOverReactantState require evaluation (and consultation with the previous author) for effective integration with the GO annotated or other Reactions and should be scheduled for revision and/or possible deprecation. Function and ReactantFunction appear might be better represented by GO Functions in the BiologicalReaction hierarchy ActivationPattern and BooleanOverReactantState might be better represented by StateVectors. The recommendation for deprecation is made because these classes were introduced in an early schema version (Sigmoid v45 or prior) that did not have a distinction or separation between BiologicalReactions and MathematicalReactions. Schema v45 was structured so that all Reactions were closely representing Cellerator functions and thus were MathematicalReactions.

## *2.7 Reactions*

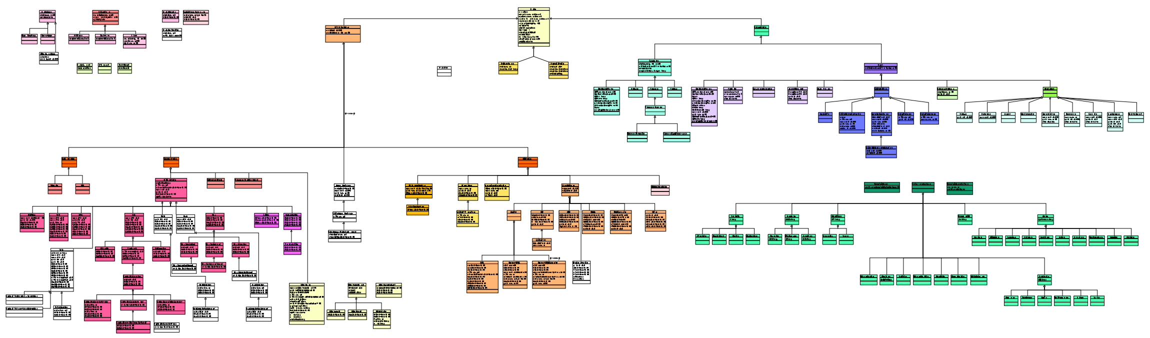


**Figure 2.23 The entire Sigmoid Reaction diagram. This diagram is comprised of a Reaction hierarchy (yellow root class), a post-translational modification hierarchy (the green tree on bottom right), and supporting utility classes (top left). The Reaction hierarchy is split into two major sections. The top right most branch of the tree is composed of Biological Reactions. The left branch is composed of Mathematical Reactions. Components and attributes of this diagram will be illustrated in more detail in the following sections.**

The Reaction class diagram of the schema shown in Figure 2.23 contains the hierarchical structure of Reactions, a hierarchical structure of

PostTranslationalModifications, and a few utility classes to support classes. The Reaction hierarchy is comprised of groupings of Reactions. Reactions are composed of biologically and mathematically relevant data classes constructed to facilitate the transformation of sets of Reactants to produce other sets of Reactants. The Post-translational hierarchy consists of common chemical modifications of protein amino acid residues that have a functional impact on proteins. The hierarchy of post-translational modifications appears to the bottom right of Figure 2.23, in green. The PostTranslationalModification and the Reaction diagram supporting utility classes are discussed in more detail in subsequent subsections.

### 2.7.1 *The Reaction Super Class.*

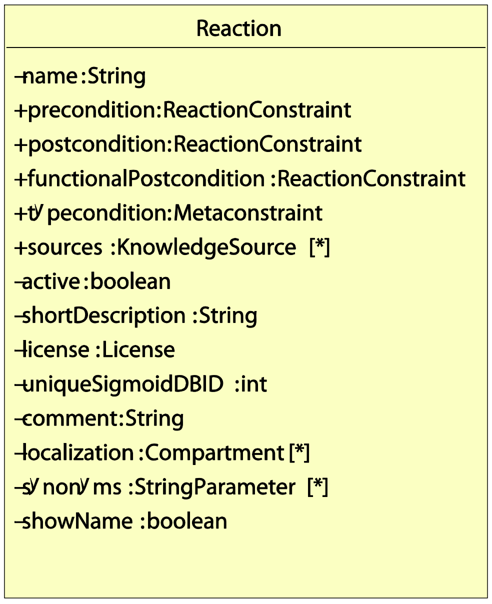

Reaction
-name :String
+precondition:ReactionConstraint
+postcondition:ReactionConstraint
+functionalPostcondition :ReactionConstraint
+typecondition:Metaconstraint
+sources :KnowledgeSource [*]
-active :boolean
-shortDescription :String
-license :License
-uniqueSigmoidDBID :int
-comment:String
-localization :Compartment [*]
-synonyms :StringParameter [*]
-showName :boolean

**Figure 2.24 Reaction is the root class of the entire Reaction Hierarchy. Every class of Reaction will inherit its attributes. The name attribute is requisite for display in SME and is required for simulation in xCellerator. KnowledgeSources, "shortDescription", and the comment field are available for annotations.**

Reaction is the root of all Biological and Mathematical reactions. We will introduce the details of the class here because all subsequent Reaction subclasses inherit all of Reactions' attributes. The "name" attribute is required for all Sigmoid Models that are to be either displayed within SME or simulated by xCellerator. The name string is passed to both for model parsing. "Comment" and "shortDescription" attributes are available for supplementary reaction information. As with the Model and Reactant classes, KnowledgeSource serves as a reference for citation and supplementary reactant documentation.

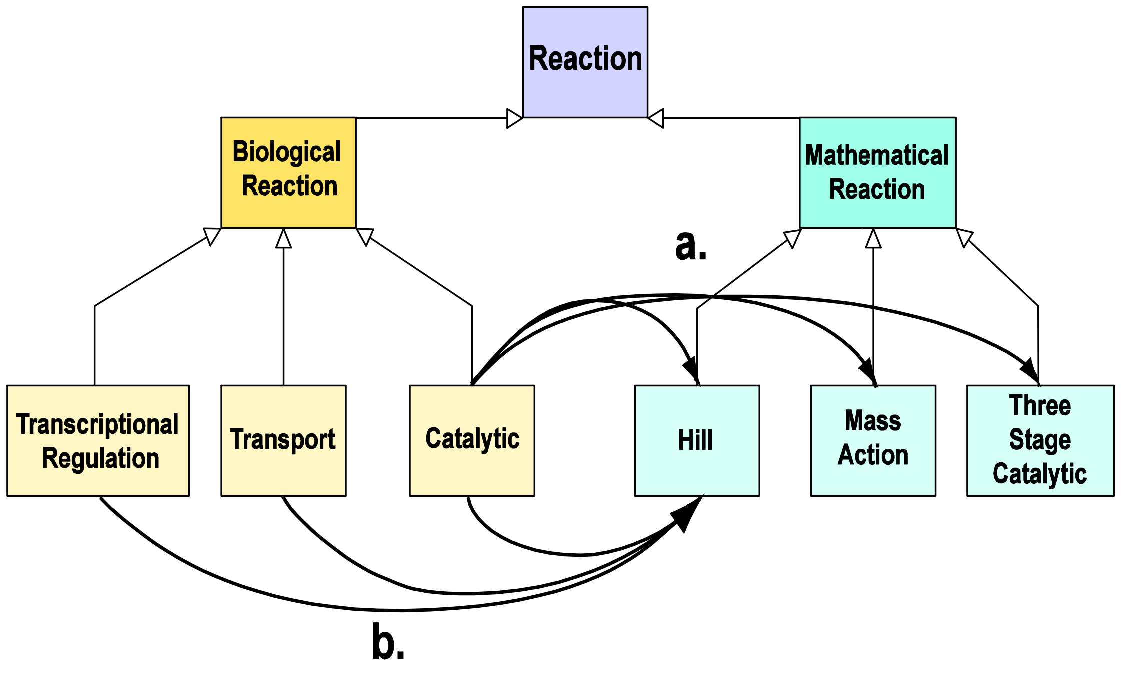


**Figure 2.25 Simplified version of the Sigmoid Schema Reaction hierarchy. (a.) There may exist a one to many relation between a particular biological reaction and potential functions (Mathematical-Reactions) that may be assigned to model the kinetics of the interaction. For instance numerous mathematical functions can be assigned to model a catalytic process. (b.) In reverse, the functional application of a particular set of differential equations may be conserved over a variety of biological phenomena so, there also may be a one to many association between a particular mathematical function (Reaction) and the biological scenarios it may be applied to. For instance a Hill equation may provide useful in modeling a catalytic reaction, transcriptional regulation or even a transport process.**

An essential function of Sigmoid is to assist in the translation of biological knowledge into mathematical form. The representation of Reactions in Sigmoid is aimed at this goal. Sigmoid Reactions represent biochemical processes that transform molecular or other biological objects that are represented as Sigmoid Reactants. A major design feature of Sigmoid is that, to support translation of biology to mathematics, Reactions are defined in two sub-hierarchies: Biological Reactions and Mathematical Reactions. The Biological Reaction hierarchy is intended to provide biologically oriented users with symbolic representations of a biochemical reaction or process. Attributes that represent the basic reactants with primary roles are included. The kinetics of the reaction are abstracted out and delegated to Mathematical Reactions. Mathematical Reactions constitute a type hierarchy of mathematical models of reactions or other processes in the Sigmoid schema. Such representations include particular rate laws, as well as the translation of compound reactions into a sub-network of more elementary reactions each of which has a more elementary mathematical model. With a few exceptions, Mathematical Reactions currently have direct xCellerator/kMech implementation functions associated with them. Numerical parameters associated with each reaction are contained by reference, which enables key reaction parameters to be shared within a MathematicalReaction or across a full reaction network. In this way the Sigmoid architecture can offer explicit support for the translation of biological processes into mathematical process models. Each type of biological reaction may in principle be translated into several alternative mathematical reaction models, and each mathematical reaction model can serve as the translation of several different biological reactions. An example of the importance of many-to-many reaction translations is shown in Figure 2.25.

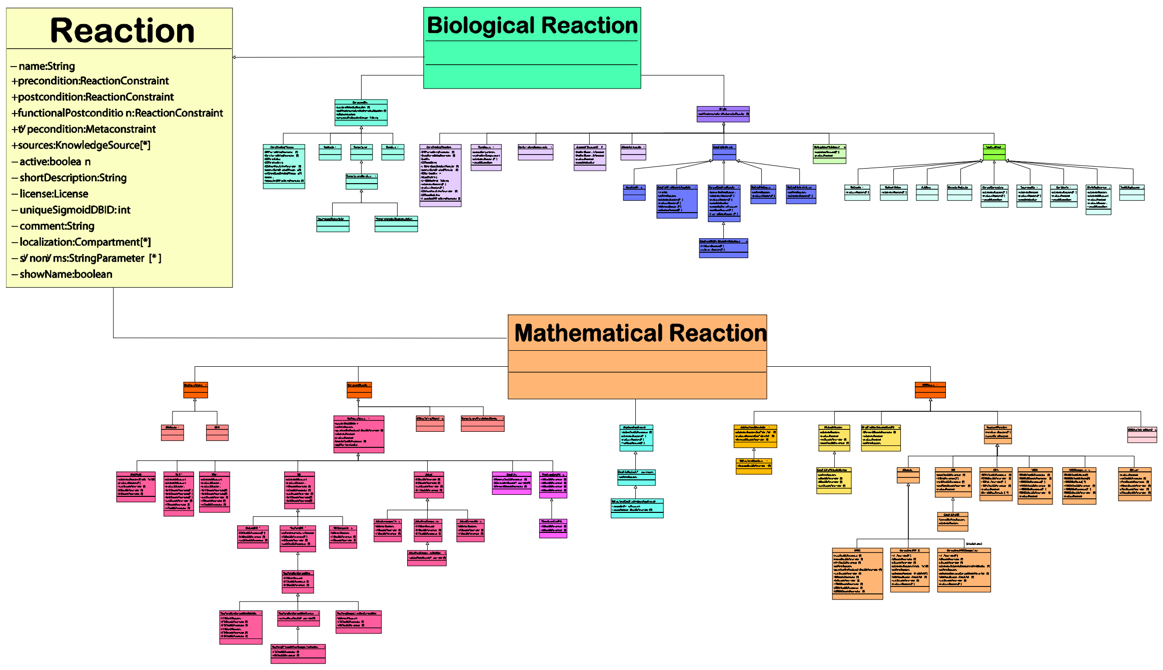

Reaction
– name:String
+precondition:ReactionConstraint
+postcondition:ReactionConstraint
+functionalPostconditio n:ReactionConstraint
+t/ pecondition:Metaconstraint
+sources:KnowledgeSource[*]
– active:boolea n
– shortDescription:String
– license:License
– uniqueSigmoidDBID:int
– comment:String
– localization:Compartment[*]
– s/ non/ ms:StringParameter [*]
– showName:boolean
Biological Reaction
Mathematical Reaction

**Figure 2.26 Other elements of the Reaction class diagram have been extracted to illustrate the Sigmoid Reaction Hierarchy of Biological and Mathematical reactions.. The top branch of this hierarchy consists of Biological Reactions. Biological Reactions contain information about the players in a reaction and are classified according to category of biological process, but are abstract in that they contain no kinetic details about a reaction. The bottom branch of the diagram consists of MathematicalReactions. MathematicalReaction classes store parameter information about specific xCellerator functions designed to model reaction kinetics.**

The Reaction hierarchy can be seen in Figure 2.26 independently of the other classes that appear in the Reaction class diagram. Other elements of the Reaction class diagram have been extracted to highlight the Biological and Mathematical subdomains of Reaction. Every BiologicalReaction contains a reference to a collection of MathematicalReactions. This collection can contain one-to-many MathematicaReactions. The mapping between biological and mathematical reactions is subject to applicability constraints. An expert user is required to determine which mathematical functions will map properly to a particular biological process. SME implements a table of suggested mathematical reactions in the model creation portion of the program, but the implementation of this feature is immature and requires further development.

## *2.8 The BiologicalReaction Sub-domain of Reactions.*

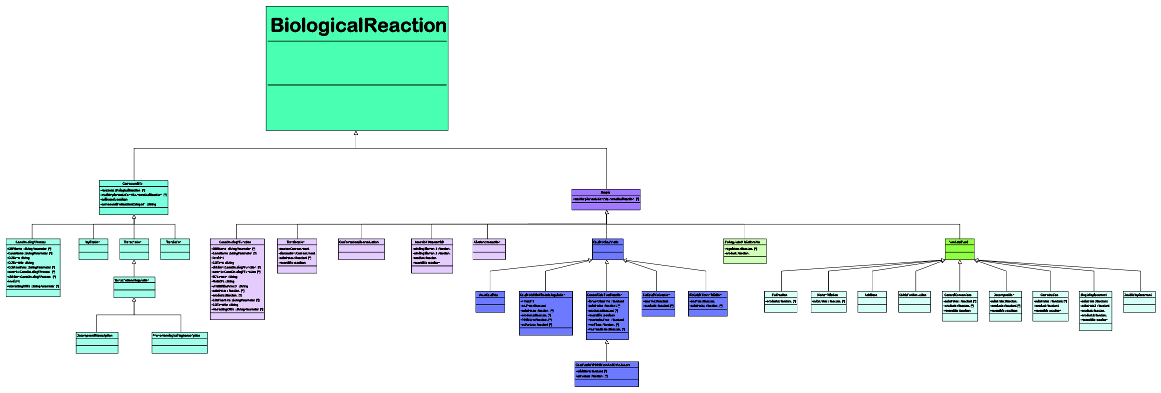


**Figure 2.27 The BiologicalReaction sub-hierarchy of Reactions has two main branches. The right branch is composed of Simple reactions, representing typically singular BiologicalReactions. There are three sections of Simple Bio Reactions, NonCatalized (far right in cyan), CataytícEnzymatic (blue), and other Simple Reactions (purple). The left branch of BioReactions (in green) is composed of CompoundBio Reactions. CompoundBio reactions are composed of many Biological reactions and can represent complex systems of biological processes. These Sections will be expanded upon in detail in the following sections.**

The biological domain of Reactions in Sigmoid was designed to present a concise representation of biological process options. An effort was made to cover the widest possible array of biological scenarios with a simple concise set of classes. The BiologicalReactions (BioReactions) contain references to the players (Reactants) in the reactions along with certain character attributes that describe the Reaction process. A common example of one of these attributes is reaction reversibility, typically stored as a Boolean value. The BioReaction sub-hiearchy is split into two main branches as illustrated in Figure 2.27. The left branch consists of non-catalyzed reactions, catalytic or enzymatic reactions, and a few other reaction types that will be discussed in the following sections. The right main branch is consists of CompoundBio Reactions. The basic principle behind the CompoundBio reactions is that they are comprised of sets of BioReactions as opposed to being singular reaction processes.

### 2.8.1 *NonCatalyzed Reactions*

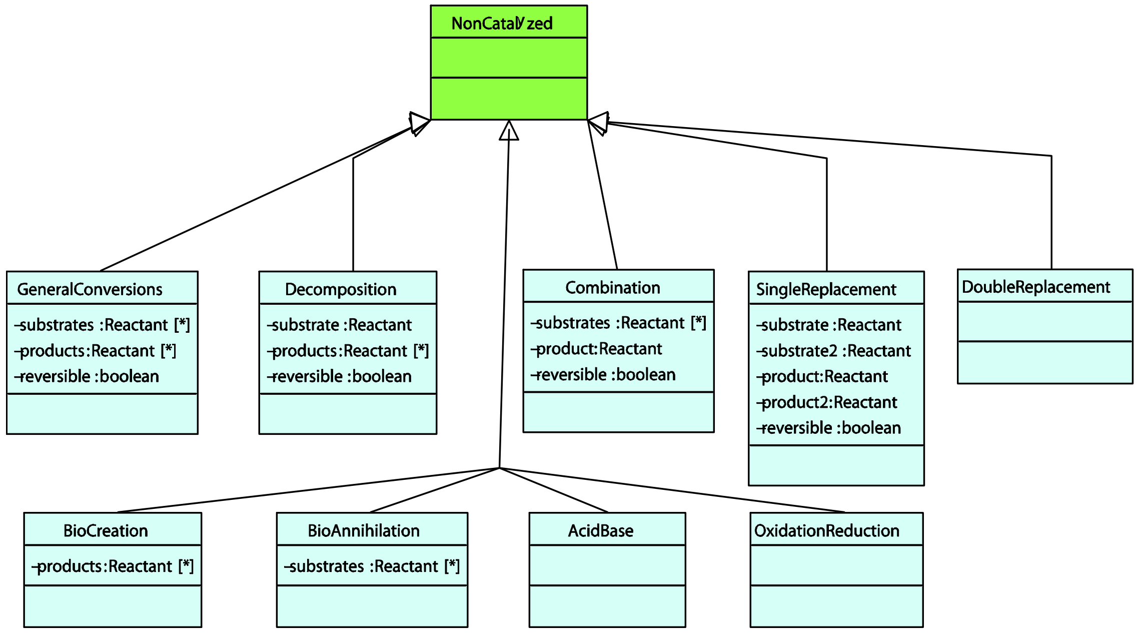


**Figure 2.28 NonCatalyzed Reactions, inheriting from the Simple BioReaction class, were constructed to represent general chemical (or non-catalytic biochemical) processes. GeneralConversions is a generalized form of simple non catalytic reaction that converts any number of substrates into any number of products. It is the most flexible representation in this sub-tree of BioReactions and can functionally represent any reaction in this figure. AcidBase, DoubleRelplacement and the OxidationReduction classes should be subclasses of, and inherit attributes from the GeneralConversions class. This inheritance error was corrected in the 2.25 schema update.**

Although not shown in Figure 2.28, all NonCatalyzed BioReactoins inherit the "mathImplementation" attribute from the Simple BioReaction class and can have zero-to-many associated MathematicalReactions (MathReaction). The NonCatalyzed sub-domain of BioReactions consists of a set of basic reactions that possess no explicit enzyme or catalyst. Basic chemical processes can be functionally represented by these reactions in the schema.

All of the reaction classes within this sub-domain of BioReactions, in practicality, can be functionally represented by the GeneralConversions class. All other subclasses of NonCatalyzed are actually limited cases of GeneralConversion. An important property of the Sigmoid classes is to provide a viable language for the modeling biologist to use in the creation of their models. These classes are within the schema because the class names are descriptive and confer biologically relevant information about common types of reactions. These classes also provide constraints on the number of substrates and products of a reaction plus a Boolean attribute to set reaction reversibility.

The BioCreation or Annihilation classes are commonly used in the existing Sigmoid models that have been coded. Models typically feature creation or destruction of a reactant species without details of the process. i.e Anabolic processes require an ATP (adenosine triphospate) source. The only relevant detail may be the abstract manifestation of the product ATP at a certain rate. Conversely with regard to BioAnnihilation, Reactant destruction may be required for a model, but including the details (enzyme that catalyzes the destruction) of that reaction not an important detail of the model reaction.

### 2.8.2 *Catalytic and Enzymatic Biological Reactions*

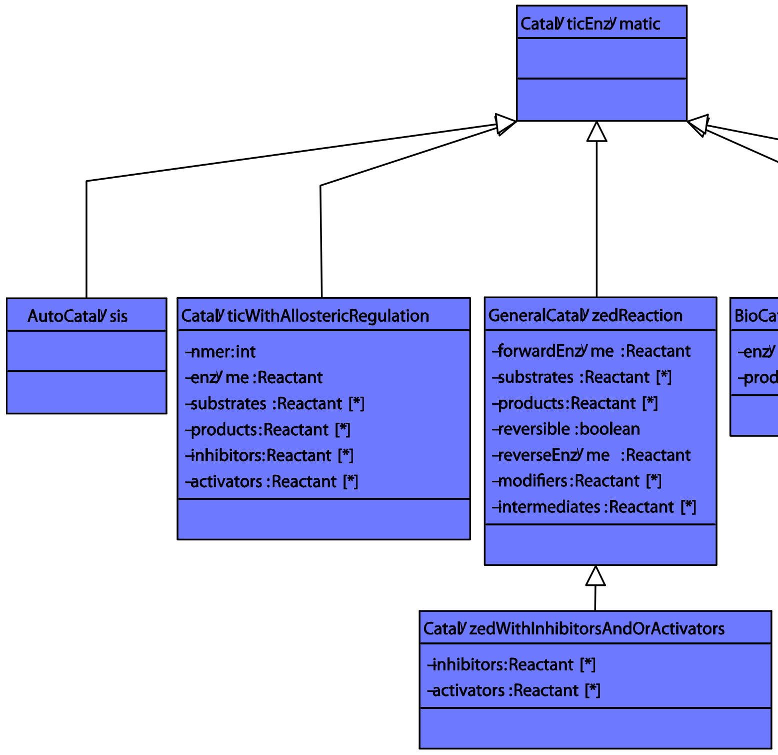


**Figure 2.29 Classes inheriting from CatalyticEnzymatic are designed to represent biological process reactions that involver either chemical catalysts or biological enzymes. Key commonalities are that both catalysts and enzymes are not consumed in the reactions and modify the rates of their reaction processes. The focus of these classes was primarily designed to support enzymatic processes but purely chemical processes and be represented by expert use.**

Although not illustrated in Figure 2.29, all CatalyticEnzymatic inherit the "mathImplementation" attribute from the Simple BioReaction class and can have zero-to-many associated MathematicalReactions. Classes inheriting from CatalyticEnzymatic are designed to represent biological process reactions that involve either chemical catalysts or biological enzymesKey commonalities are that both catalysts and enzymes are not consumed in the reactions and modify the rates of their reaction processes. These classes were primarily designed to support enzymatic processes but purely chemical processes and be represented as well if necessary.

The GeneralCatalyzedReaction BioReaction was constructed to handle the general cases of enzymatic processes for zero-to-many substrates and products. Frequently, biologically catalyzed reactions posses a reverse reaction counterpart in which the reverse reaction is catalyzed by a different enzyme. For instance, enzyme E may convert substrate A into product B. The reverse enzyme E2 may convert product B back into substrate A. Markup languages such as SBML or KGML typically provide reverse reactions when available. The "intermediates" attribute of GeneralCatalyzedReaction specifies enzyme intermediate states for cases like PingPong Reactions. The "modifiers" attribute is available for regulatory reactants. The BioCatalyticCreation and BioCatalyticAnnihilation classes are special cases of GeneralCatalyzedReaction in that they designate the corresponding creation and destruction reactions. The need to specifically designate inhibitors or activators can be accommodated by the CatalzyedWithInhibitorAndOrActivators subclass of GeneralCatalyzedReaction.

Enzyme process models that specify allosteric regulation should be implemented by the CatalyticWithAllostericRegulation class. A Sigmoid pathway model that uses this class is the (Yang 2005) model.

AutoCatalysis was included as a reaction class designation but a set of use cases have not stimulated the addition of relevant attributes for the class. The stub class remains in the schema as an important biochemical process, but awaits implementation.

### 2.8.3 *Additional Simple Biological Reactions*

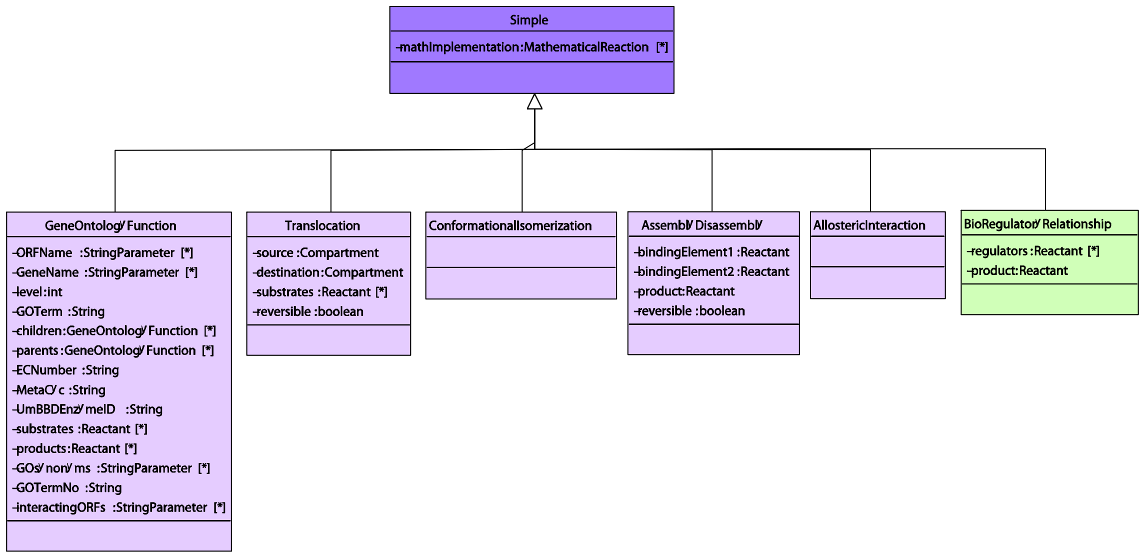

Simple
-mathImplementation:MathematicalReaction [*]
GeneOntologyFunction
-ORFName :StringParameter [*]
-GeneName :StringParameter [*]
-level:int
-GOTerm :String
-children:GeneOntologyFunction [*]
-parents:GeneOntologyFunction [*]
-ECNumber :String
-MetaCyc :String
-UmBBDEnzymeID :String
-substrates :Reactant [*]
-products:Reactant [*]
-GOSynonyms :StringParameter [*]
-GOTermNo :String
-interactingORFs :StringParameter [*]
Translocation
-source :Compartment
-destination:Compartment
-substrates :Reactant [*]
-reversible :boolean
ConformationalIsomerization
AssemblyDisassembly
-bindingElement1 :Reactant
-bindingElement2 :Reactant
-product:Reactant
-reversible :boolean
AllostericInteraction
BioRegulatoryRelationship
-regulators :Reactant [*]
-product:Reactant

**Figure 2.30 Additional Simple reactions from right to left, BioRegulatoryRelationships for regulatory proceses, AllostericInteractions (attributes unimplemented), AssemblyDisassembly for complex formation, ConformationalIsomerizations (attributes unimplemented), Translocation for inter-compartmental reactions and the GeneOntologyFunction. GeneOntologyFunction is a class designed to represent reactions within a hierarchy of reactions.**

A few additional Simple BioReaction classes have been included in the schema. BioRegulatoryRelationship is present to allow a regulatory system to be modeled where substrates are not specified. Only single product reactions with multiple regulators are allowed for the BioRegulatoryRelationship class. AssemblyDissasembly, a reversible reaction, was included for complex formation and specifies two binding partners and a single Reactant product. Translocation is a class designed for movement of Reactants across compartments. Source and destination compartments are specified for Translocation and the reaction is reversible. ConformationalIsomerization is an important Reactant transformation reaction but has remained unimplemented and therefore possesses no attributes.

### 2.8.4 *CompoundBio Reactions*

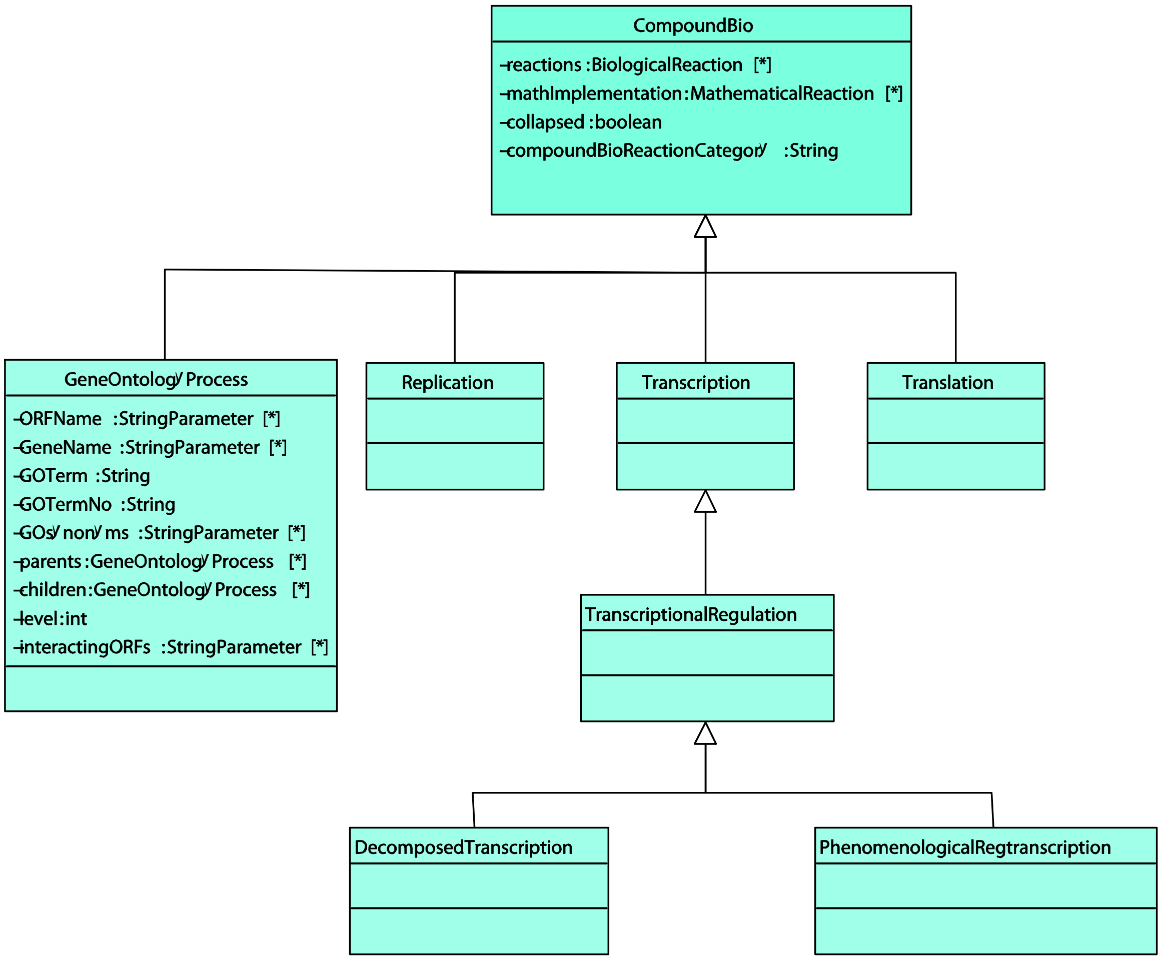

CompoundBio
-reactions :BiologicalReaction [*]
-mathImplementation:MathematicalReaction [*]
-collapsed :boolean
-compoundBioReactionCategory :String
GeneOntologyProcess
-ORFName :StringParameter [*]
-GeneName :StringParameter [*]
-GOTerm :String
-GOTermNo :String
-GOSynonyms :StringParameter [*]
-parents :GeneOntologyProcess [*]
-children:GeneOntologyProcess [*]
-level:int
-interactingORFs :StringParameter [*]
Replication
Transcription
Translation
TranscriptionalRegulation
DecomposedTranscription
PhenomenologicalRegtranscription

**Figure 2.31 The CompoundBio reactions represent biological processes that span whole collections of biological processes. CompoundBio contain other Biological Reactions, including other CompoundBio reactions, resulting in a nested structure that is built to represent highly complex and involved biological processes. Of the vast number of complex biological processes, DNA replication, transcription and translation have been added as subclasses to CompoundBio because of their biological prominence. To handle the vast array of biological processes, the GeneOntologyProcess class has been included.**

CompoundBio Reactions are composed of collections of BioReactions and associated MathematicalReactions. They are designed to provide a layer of abstraction and scalability within a model. Many biological processes such as DNA replication, transcription and translation can be highly complex and involve many sub-processes and reactions. The CompoundBio reactions may assist modeling biologists to reference these sub-networks of reactions as a single process. Since CompoundBio Reactions contain other BioReactions by reference, biological processes can be represented in a nested fashion, thus adding a layer of scalability to the representation. Although the number and attributes of biological processes is vast, only a few key subclasses of CompoundBio have been added to the schema. To address the diversity of biological processes that require representation, the Gene Ontology process class was included. Gene Ontologies provide access to a hierarchy of biological processes. The Gene Ontology representation will be discussed in more detail in Section 2.11.1.

## 2.9 *MathematicalReactions*

### 2.9.1 ***The MathematicalReaction sub-domain.***

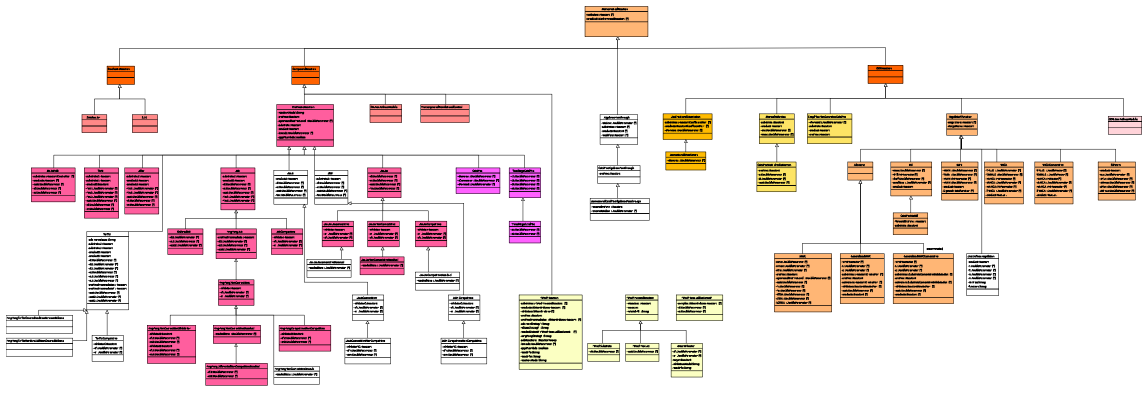


**Figure 2.32 The MathematicalReaction domain of Reactions. The branch of equation classes to the right is comprised of ODEReaction equations that have direct counterparts in xCellerator. The ODEReaction classes are data classes that store reaction parameters for simulation. Reactions on the large left branch are Compound reactions which comprise sets of ODEReactions. The center Reactions in white are AlgebraicPassthrough reactions designed to pass algebraic expressions directly to the Mathematica simulator. The small branch of reactions on the far left are reserved for stochastic simulations and serve as stub classes awaiting further implementation.**

The MathematicalReaction (MathReactions) sub-hierarchy serves as the data structures to store reactant and parameter information for all xCellerator and kMech reactions implemented by Sigmoid. It should be clear that only MathReactions are processed by the middleware translation package and sent to xCellerator for simulation. BioReactions are not processed in this manner. MathReactions are not visualized currently in the SME network layout view, but instead are referenced by BioReactions. MathReactions associated with a particular BioReaction are accessible in the side panel of SME when the BioReaction is selected (shown in Figure 1.4 as item C).

The MathReaction hierarchy of Reactions is composed of four main branches of MathReaction. There are three reactions on the far left branch of Figure 2.32 which are stub classes reserved for stochastic simulation. This type of modeling has not been implemented in Sigmoid so these classes serve as place holders awaiting further development. Three major classifications of MathematicalReactions are present in the Schema. First, Ordinary differential equation (ODE) ODEReactions are reactions that are direct parameterized representations of xCellerator functions. Second, Compound MathReactions are reactions that consist of sets of reactions. Lastly, AlgebraicPassthrough Reactions are reactions that store algebraic expressions to be passed directly to Mathematica for processing. These sub-domains of MathReaction will be discussed in more detail in the subsequent sections.

### *2.9.1.1 ODEReaction MathematicalReactions*

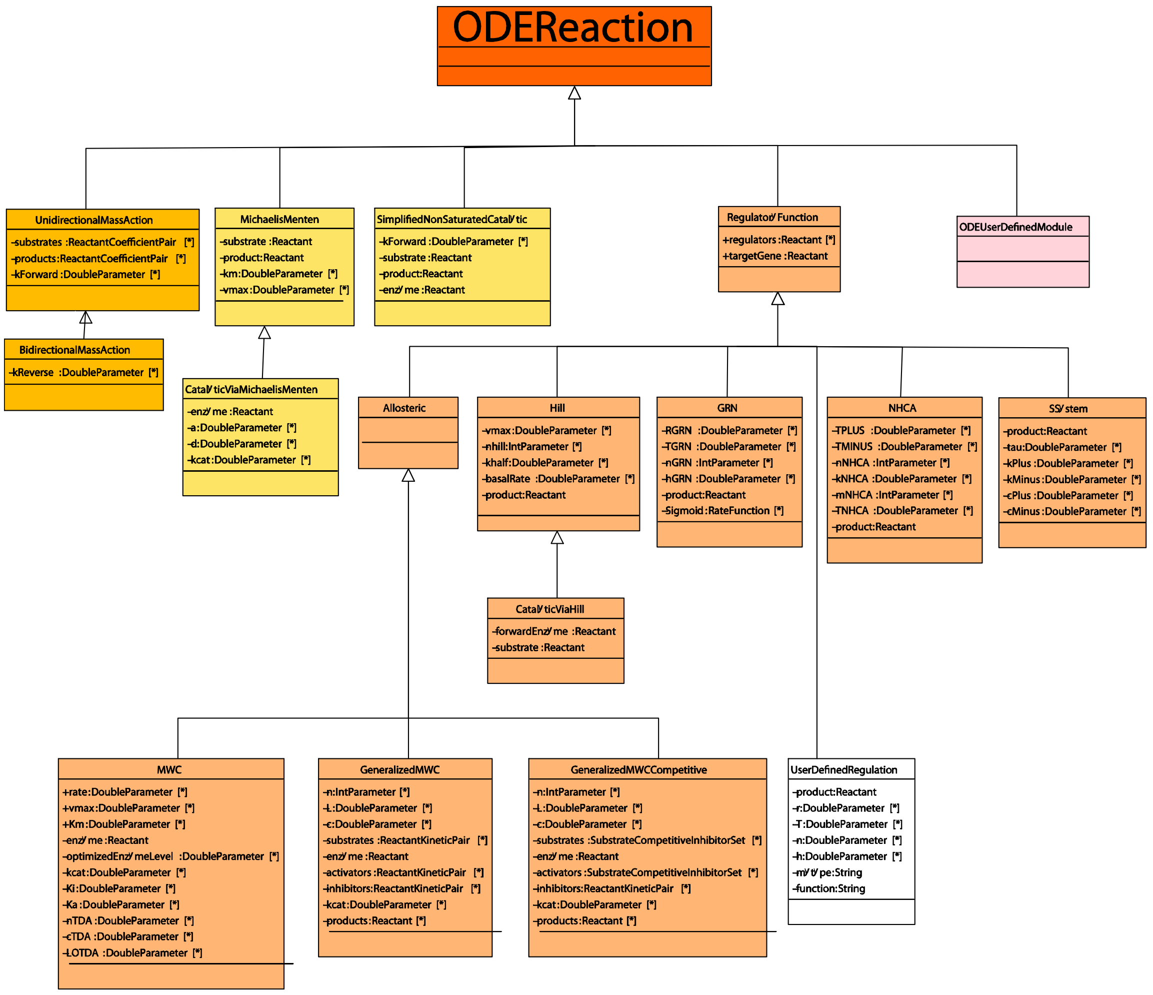

ODEReaction
UnidirectionalMassAction
-substrates :ReactantCoefficientPair [*]
-products:ReactantCoefficientPair [*]
-kForward :DoubleParameter [*]
BidirectionalMassAction
-kReverse :DoubleParameter [*]
MichaelisMenten
-substrate :Reactant
-product:Reactant
-km:DoubleParameter [*]
-vmax:DoubleParameter [*]
CatalyticViaMichaelisMenten
-enzyme :Reactant
-a:DoubleParameter [*]
-d:DoubleParameter [*]
-kcat:DoubleParameter [*]
SimplifiedNonSaturatedCatalytic
-kForward :DoubleParameter [*]
-substrate :Reactant
-product:Reactant
-enzyme :Reactant
RegulatoryFunction
+regulators :Reactant [*]
+targetGene :Reactant
ODEUserDefinedModule
Allosteric
Hill
-vmax:DoubleParameter [*]
-nhill:IntParameter [*]
-khalf:DoubleParameter [*]
-basalRate :DoubleParameter [*]
-product:Reactant
CatalyticViaHill
-forwardEnzyme :Reactant
-substrate :Reactant
GRN
-RGRN :DoubleParameter [*]
-TGRN :DoubleParameter [*]
-nGRN :IntParameter [*]
-hGRN :DoubleParameter [*]
-product:Reactant
-Sigmoid:RateFunction [*]
NHCA
-TPLUS :DoubleParameter [*]
-TMINUS :DoubleParameter [*]
-nNHCA :IntParameter [*]
-kNHCA :DoubleParameter [*]
-mNHCA :IntParameter [*]
-TNHCA :DoubleParameter [*]
-product:Reactant
SSystem
-product:Reactant
-tau:DoubleParameter [*]
-kPlus :DoubleParameter [*]
-kMinus :DoubleParameter [*]
-cPlus :DoubleParameter [*]
-cMinus :DoubleParameter [*]
MWC
+rate:DoubleParameter [*]
+vmax:DoubleParameter [*]
+Km:DoubleParameter [*]
-enzyme :Reactant
-optimizedEnzymeLevel :DoubleParameter [*]
-kcat:DoubleParameter [*]
-Ki:DoubleParameter [*]
-Ka :DoubleParameter [*]
-nTDA :DoubleParameter [*]
-cTDA :DoubleParameter [*]
-LOTDA :DoubleParameter [*]
GeneralizedMWC
-n:IntParameter [*]
-L:DoubleParameter [*]
-c:DoubleParameter [*]
-substrates :ReactantKineticPair [*]
-enzyme :Reactant
-activators :ReactantKineticPair [*]
-inhibitors:ReactantKineticPair [*]
-kcat:DoubleParameter [*]
-products:Reactant [*]
GeneralizedMWCCompetitive
-n:IntParameter [*]
-L:DoubleParameter [*]
-c:DoubleParameter [*]
-substrates :SubstrateCompetitiveInhibitorSet [*]
-enzyme :Reactant
-activators :SubstrateCompetitiveInhibitorSet [*]
-inhibitors:ReactantKineticPair [*]
-kcat:DoubleParameter [*]
-products:Reactant [*]
UserDefinedRegulation
-product:Reactant
-r:DoubleParameter [*]
-T:DoubleParameter [*]
-n:DoubleParameter [*]
-h:DoubleParameter [*]
-mytype:String
-function:String

**Figure 2.33 The ODEReactions are data representations designed to store reaction parameters for xCellerator simulator processing. Each class shown here, with the exception of the attribute empty classes, possesses a directly corresponding xCellerator function. There are two main classifications of Reaction: Regulatory reactions appear in the right main (dark orange) branch; Nonregulatory reactions appear to the left. Subsequent class diagrams will reveal the attributes in more detail.**

The ODEReactions consist of MathematicalReactions that are parameter input representations for xCellerator [B. Shapiro2007] functions. The ODEReaction are data classes designed to store parameter inputs for reaction equations consisting of ordinary differential equations (ODE). The particulars of the reaction equations themselves are located at http://xlr8r.info and deprecated functions (for older versions of Sigmoid) are located at http://cellerator.info.

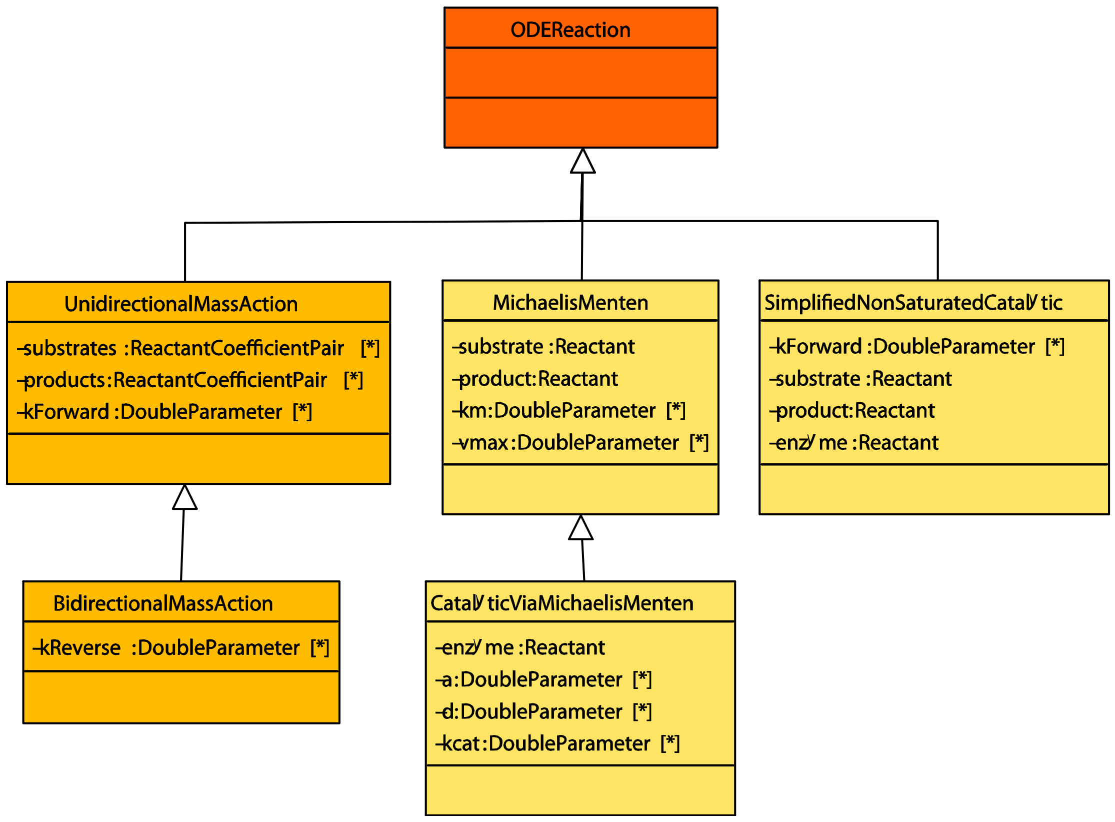

ODEReaction
UnidirectionalMassAction
-substrates :ReactantCoefficientPair [*]
-products:ReactantCoefficientPair [*]
-kForward :DoubleParameter [*]
MichaelisMenten
-substrate :Reactant
-product:Reactant
-km:DoubleParameter [*]
-vmax:DoubleParameter [*]
SimplifiedNonSaturatedCatalytic
-kForward :DoubleParameter [*]
-substrate :Reactant
-product:Reactant
-enzyme :Reactant
BidirectionalMassAction
-kReverse :DoubleParameter [*]
CatalyticViaMichaelisMenten
-enzyme :Reactant
-a:DoubleParameter [*]
-d:DoubleParameter [*]
-kcat:DoubleParameter [*]

**Figure 2.34 Massaction, MichaelisMenten, and a now deprecated SimplifiedNonSaturatedCatalytic enzymatic reaction appear in the ODEReaction tree.**

Two forms of mass action kinetic equation data classes are available for modeling: UnidirectionalMassAction, which stores only a forward rate, and a bidirectional subclass possessing a reverse rate. The mass action equations form the basis of the Compound MathReactions. All Compound MathReaction enzyme mechanisms (Section 2.9.1.2) are composed of sets of mass action reactions. A classic MichaelisMenten parameter class is available to model enzyme kinetics. The SimplifiedNonSaturatedCatalytic equation is a deprecated function and is now handled by the TwoStageCatalytic Compound MathReaction. SimplifiedNonSaturatedCatalytic persists in the schema to support some older Sigmoid models until they are recoded.

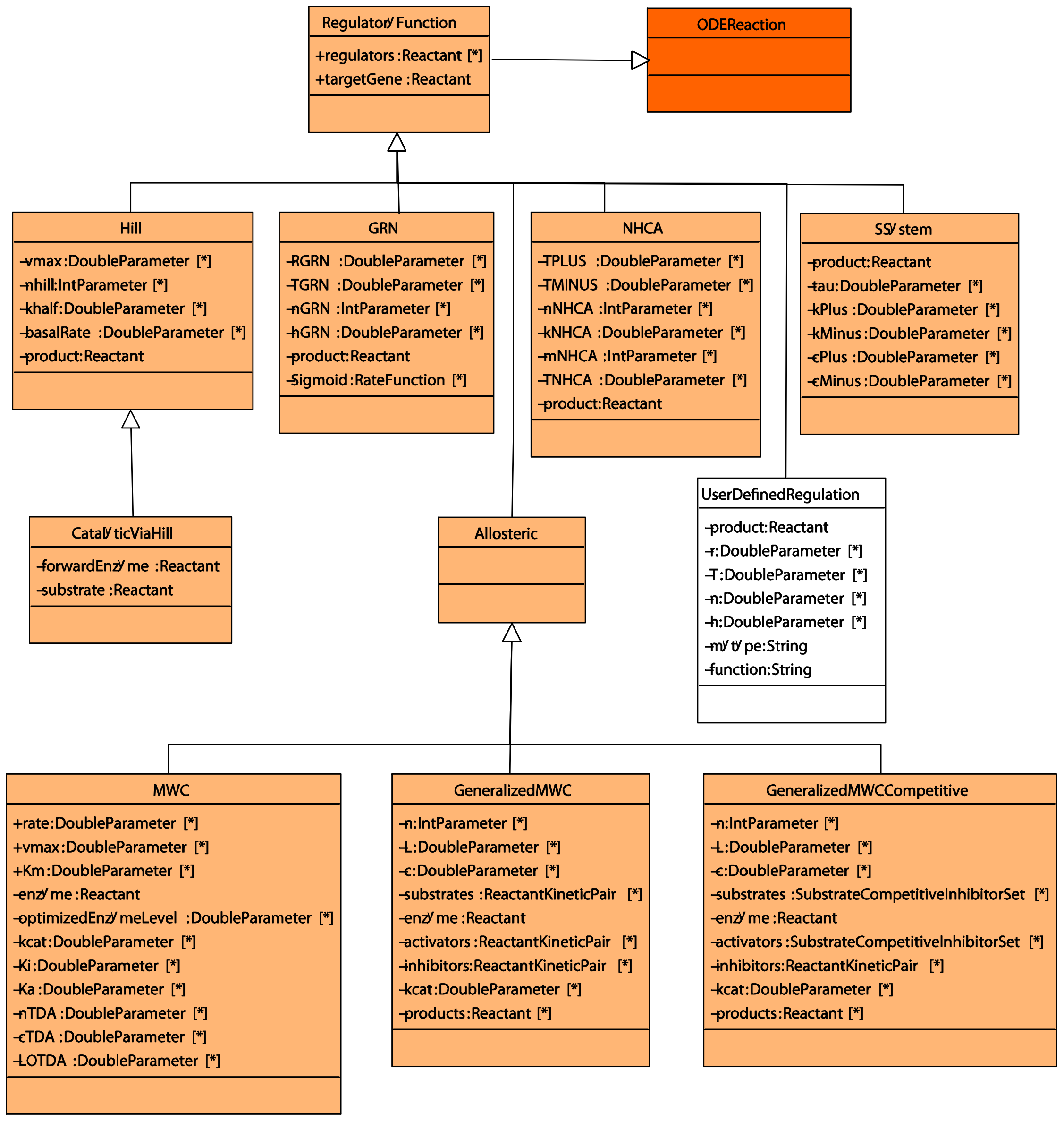

RegulatoryFunction
+regulators :Reactant [*]
+targetGene :Reactant
ODEReaction
Hill
-vmax:DoubleParameter [*]
-nhill:IntParameter [*]
-khalf:DoubleParameter [*]
-basalRate :DoubleParameter [*]
-product:Reactant
GRN
-RGRN :DoubleParameter [*]
-TGRN :DoubleParameter [*]
-nGRN :IntParameter [*]
-hGRN :DoubleParameter [*]
-product:Reactant
-Sigmoid:RateFunction [*]
NHCA
-TPLUS :DoubleParameter [*]
-TMINUS :DoubleParameter [*]
-nNHCA :IntParameter [*]
-kNHCA :DoubleParameter [*]
-mNHCA :IntParameter [*]
-TNHCA :DoubleParameter [*]
-product:Reactant
SSystem
-product:Reactant
-tau:DoubleParameter [*]
-kPlus :DoubleParameter [*]
-kMinus:DoubleParameter [*]
-cPlus :DoubleParameter [*]
-cMinus:DoubleParameter [*]
CatalyticViaHill
-forwardEnzyme :Reactant
-substrate :Reactant
Allosteric
UserDefinedRegulation
-product:Reactant
-r:DoubleParameter [*]
-T:DoubleParameter [*]
-n:DoubleParameter [*]
-h:DoubleParameter [*]
-mytype:String
-function:String
MWC
+rate:DoubleParameter [*]
+vmax:DoubleParameter [*]
+Km:DoubleParameter [*]
-enzyme:Reactant
-optimizedEnzymeLevel :DoubleParameter [*]
-kcat:DoubleParameter [*]
-Ki:DoubleParameter [*]
-Ka :DoubleParameter [*]
-nTDA :DoubleParameter [*]
-cTDA :DoubleParameter [*]
-LOTDA :DoubleParameter [*]
GeneralizedMWC
-n:IntParameter [*]
-L:DoubleParameter [*]
-c:DoubleParameter [*]
-substrates :ReactantKineticPair [*]
-enzyme :Reactant
-activators :ReactantKineticPair [*]
-inhibitors:ReactantKineticPair [*]
-kcat:DoubleParameter [*]
-products:Reactant [*]
GeneralizedMWCCompetitive
-n:IntParameter [*]
-L:DoubleParameter [*]
-c:DoubleParameter [*]
-substrates :SubstrateCompetitiveInhibitorSet [*]
-enzyme :Reactant
-activators :SubstrateCompetitiveInhibitorSet [*]
-inhibitors:ReactantKineticPair [*]
-kcat:DoubleParameter [*]
-products:Reactant [*]

**Figure 2.35 There are several regulatory functions supported by xCellerator. A regulation indicates that a particular Reactant modifies the kinetics of a reaction equation but is not consumed in the process. The corresponding data classses appearing in the schema with all requisite xCellerator input parameters are, from left to right, Hill (regulatory) and its subclass CatalyticViaHill, GRN (Genetic Regulatory Network), NHCA (Non-heirarchical, Cooperative Activation), and SSystem. Also, a UserDefinedRegulation exists, and a subsection of allosteric regulations is composed of three classes of the Monod-Wyman-Changeux (MWC) model.**

Several classes of RegulatoryFunctions are available in the ODEReaction portion of the schema. Two forms of Hill equations are available, a purely regulatory form in which no substrate is consumed (Hill) and CatalyticViaHill version where the substrate is consumed. A class for Genetic Regulator Network (GRN) reactions is available. Three forms of Monod-Wyman-Changeux are present for allosteric regulations. The generalized form of MWC (GMWC) reaction equation is used in the Yang_2005_Ile-Val-Leu model in the Sigmoid database [Najdi *et al.* 2005] A regulatory function for Non-heirarchical Cooperative Activation (NHCA) is available as well as SSystem (Savageau 1969) (Savageau 1970), and a UserDefinedRegulation.

#### 2.9.1.2 *Compound MathematicalReactions.*

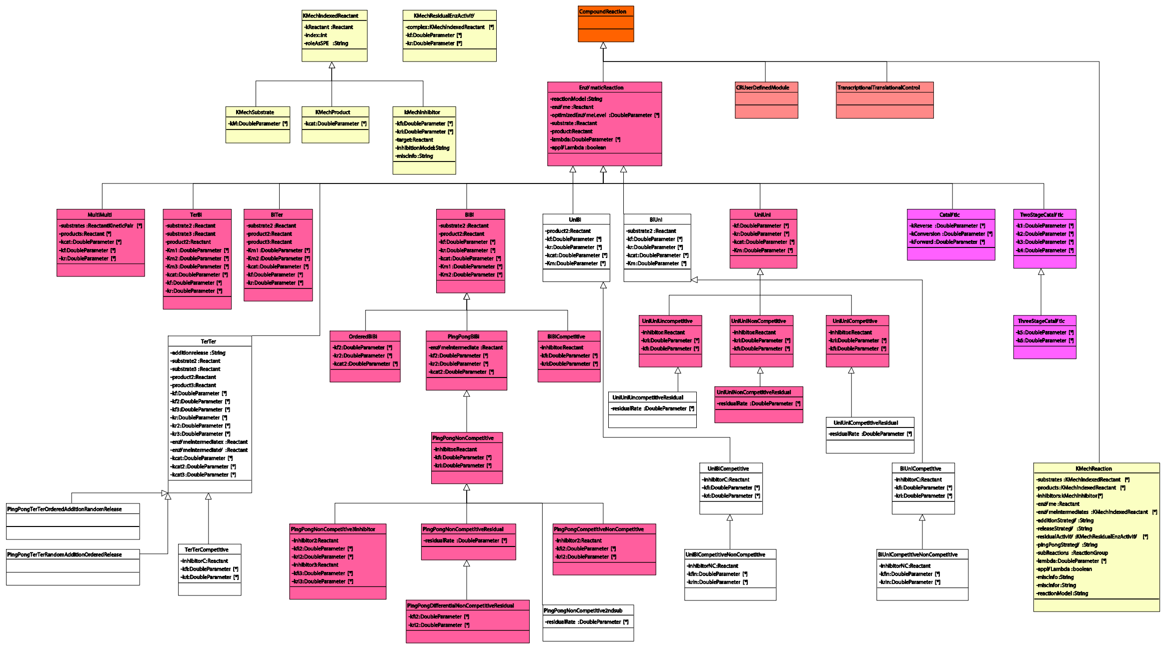


**Figure 2.36 The Compound Reaction section of the Reaction hierarchy. Compound Reactions are conceptually single Reactions that are composed of sets of ODEReactions. The majority are enzymatic processes. Reactions shown in purple on the right are catalytic reactions that have direct analogs with xCellerator reactions. Reactions illustrated in either red or white are direct representations of existing kMech reaction equations. Classes shown in while are classes to support more recent additions to kMech. Reactions shown in yellow are part of the KMechReaction representation which is a parameterized version of the kMech reactions shown in white and red. The KMechReaction set of classes is processed by the SigMech Utility to produce all of the associated sub-reactions.**

The Compound Mathematical Reactions portion of the MathReaction hierarchy featured in Figure 2.36 consists of enzymatic Reactions that are composed of sets of sub-Reactions. The ThreeStageCatalytic reaction featured previously in Figure 1.2 is an example of an xCellerator compound reaction. It is a single parameterized reaction like process that possesses six sub-reactions. All Compound Reaction class instances have corresponding xCellerator or kMech reactions. The majority of CompoundReactions are kMech enzymatic reactions.

The Compound section of the schema began has grown as new compound reactions have been added to kMech and xCellerator. It has become apparent that the additional kMech reactions are built on a few general principles and that storing instances of enzyme mechanisms as a hierarchy of classes would be a cumbersome process for the design cycle of Sigmoid. Each instance of a new kMech reaction case would yield a requirement to process a schema, generate the Java API classes, set up a development environment for the particular schema release, update the middleware Sigmoid-to-Cellerator code, and retask SME with the new Java API. To alleviate this coding burden, the new KMechReaction Classes featured in Figure 2.37 are built to replace the previous strategy of adding new enzyme mechanism reactions as they are requested by modeling biologists.

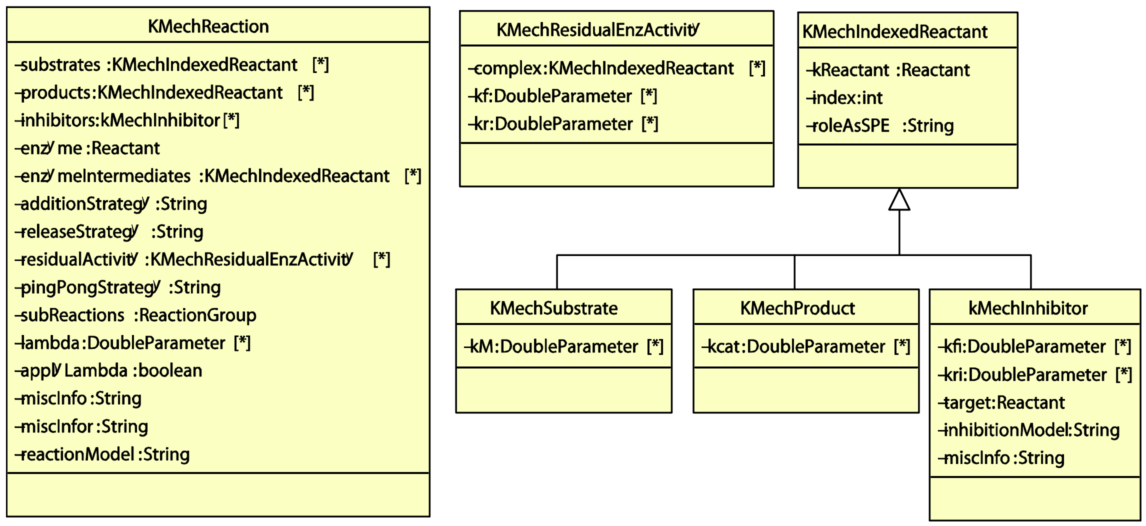


**Figure 2.37 The KMechReaction classes are designed to be a replacement representation for all (self excluded) existing compound enzymatic reactions featured in Figure 2.36. The representation is a generalized form of previous enzymatic reaction mechanism classes within the Sigmoid schema. The KMechReaction class must be processed by SigMech (0) in order to generate associated sub-reactions.**

The KMechReaction class has been designed as a replacement representation for all existing compound enzymatic reactions in the CompoundReaction portion of the MathReaction hierarchy of Reaction classes. These reactions consist of three xCellerator enzymatic compound reactions and over 30 kinds of kMech reaction classes.

Furthermore, all potential new kMech reaction classes that follow previous motifs will be encompassed by the new representation. The KMechReaction representation relies upon a parameterized set of inputs and a set of procedures (SigMech) that generate the sub-reactions of the enzyme mechanism. In this representation Substrate, Product, kMechInhibitor, and enzymeIntermediate Reactants are stored as vectors along with associated kinetic parameters. A set of "strategies" is stored to designate sub-reaction generation procedures and process models of inhibitions are specified for kMechInhibitors. SigMech is described in detail in 0.

### *2.9.1.3 AlgebraicPassthrough MathematicalReactions*

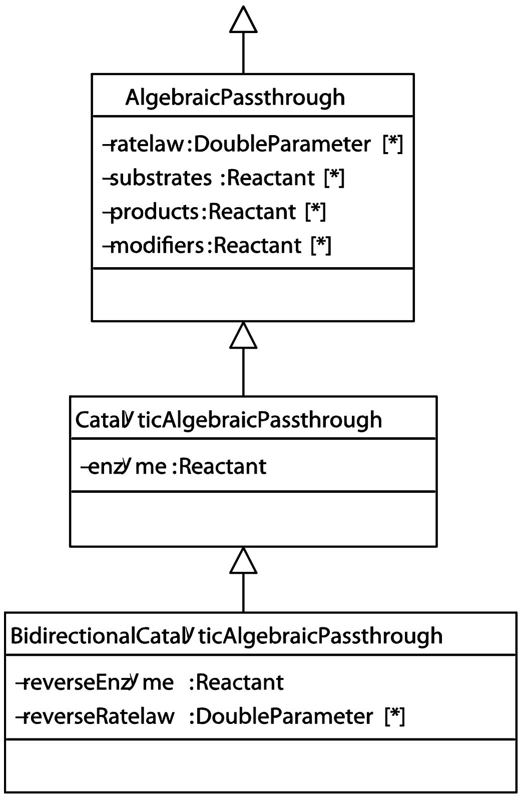

AlgebraicPassthrough
-ratelaw:DoubleParameter [*]
-substrates :Reactant [*]
-products:Reactant [*]
-modifiers:Reactant [*]
CatalyticAlgebraicPassthrough
-enzyme :Reactant
BidirectionalCatalyticAlgebraicPassthrough
-reverseEnzyme :Reactant
-reverseRatelaw :DoubleParameter [*]

**Figure 2.38 The AlgebraicPassthrough subclasses of MathematicalReaction were constructed to pass algebraic rate laws directly to Mathematica for processing. Three classes noncatalytic, a catalyzed version that includes a field for an enzyme (or catalyst) and a bidirectional version that includes a reference for an enzyme that catalyzes an additional reverse reaction.**

The AlgebraicPassthrough reactions provide a mechanism by which raw algebraic expressions can be passed through the middleware to Mathematica for processing. The "rateLaw" attributes store instances of RateFunction which inherits from DoubleParameter. A catalytic version of the passthrough class and a bidirectional passthrough reaction are present in the schema as illustrated by Figure 2.38. The AlgebraicPassthrough reactions were incorporated into the schema for two primary reasons. First, the representation allows for user defined functions that may not correspond directly to existing xCellerator or kMech functions. Secondly, pathway models stored by the SBML markup language have their rate laws in algebraic form. These SBML functions do not necessarily map directly to xCellerator. As will be explained in more detail in chapter 4, automated populator programs have been developed to automatically load SBML models into the Sigmoid database. Since the SBML format does not directly translate to xCellerator functions, a mechanism was needed to pass these SBML rate functions through to Mathematica for processing. Also, the same limitation applied to SBML models coded by hand from the Biomodels model repository. Some of these models contain expressions that do not easily translate to xCellerator functions.

### 2.9.2 *Reaction Utility Classes.*

Classes other than Reactions are present in the UML Reaction class diagram. A few utility classes exist that are necessary to support the Reaction classes and their handling of parameters. Other classes are designed to group Reactions for organizational and display purposes. These support classes are described in this section.

#### *2.9.2.1 Parameter Wrapper Classes*

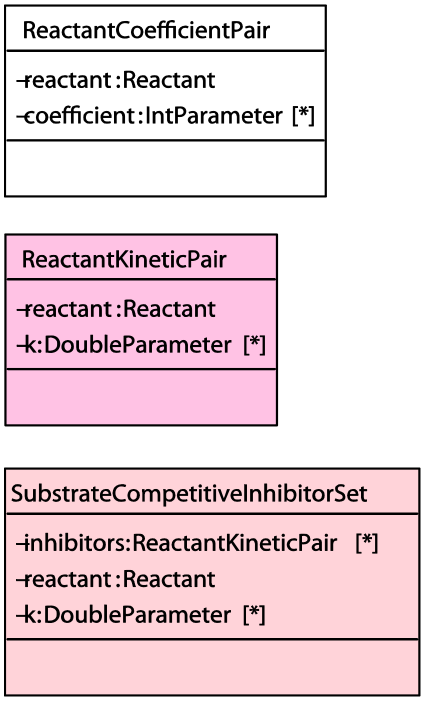


**Figure 2.39 There are three frequently used parameter wrapper classes in the Reaction class diagram. ReactantCoefficientPairs are used to specify the stoichiometry of reaction equations. ReactantKineticPairs are a mechanism to store values such as substrate affinity for substrate binding of an enzyme or, rates for inhibitor binding. SubstrateCompetitvieInhibitorSets were provided to store parameters for enzymatic inhibitors specific to a particular substrate.**

A few parameter wrapper classes were designed to coordinate with the Reaction classes. In cases where stochiometry is involved in reaction equations, integer coefficients are required. The ReactantCoefficientPair class handles stoichiometric coefficients for particular reactants in reaction equations. ReactantKineticPairs store parameter information specific to a particular Reactant within a Reaction. For instance a Reactant may have a particular binding rate to an enzyme. SubstrateCompetitiveInhibitor sets store substrate and activator parameter information for the GeneralizedMWC (MWC

Monod-Wyman-Changeux) MathReaction which implements an allosteric form of cooperativity [*Najdi et al. 2005*].

### *2.9.2.2 Reaction Groups*

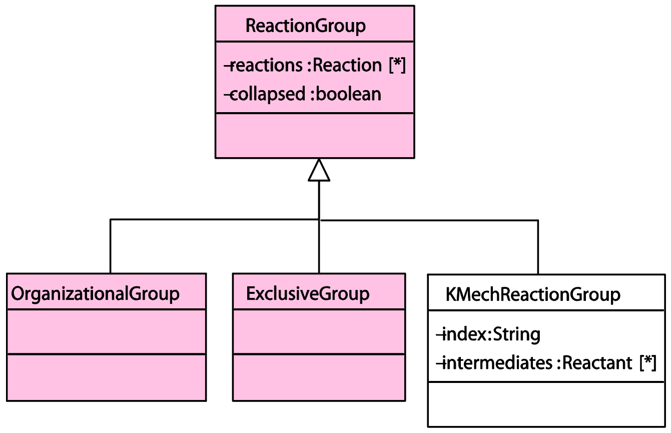
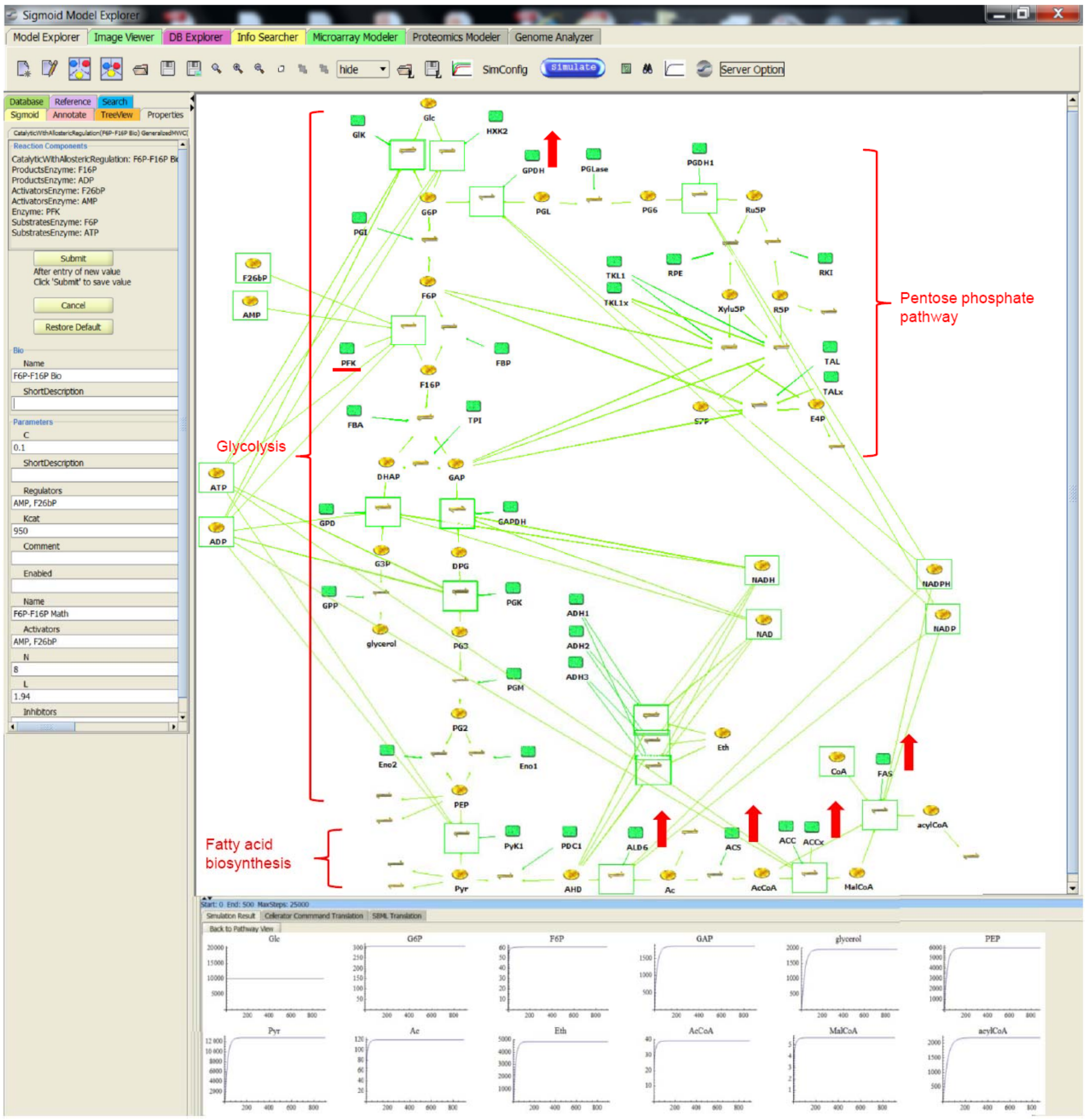


**Figure 2.40 Reaction groups serve as an organizational tool to reference groups of related reactions. They're available to the user interface (SME) to hide or expand portions of the pathway networks. The KMechReactionGroup serves an additional purpose. Since SigMech builds sets of sub-reactions from a parameterized set of inputs (a KMechReaction), The KMechReactionGroups are present to store these sets of Reactions in the database. (KMechReactionGroup could be renamed SigMechReactionGroup because it was designed to store the output of SigMech).**

Reaction groups were included in the schema as a mechanism by which sections of pathways could be partitioned. This partitioning is to assist with the organization of Reactions for of the modeling biologist. For example, enabling the collapse and expansion of relevant portions of pathways, within SME, could prove useful for large interconnected models. In theory, sections of pathways could be collapsed and iconically represented with a few remaining input and output reaction arrows remaining. In cases where pathways are large and highly interconnected the number of reaction arrows in pathway network view of SME can become overbearing. Partitioning should serve to alleviate this logistical obstacle. Larger models containing multiple related pathways can be more ideally managed. For example, the [Najdi 2011] fatty acid biosynthesis model, shown in Figure 2.41, contains portions of the pentose phosphate pathway, glycolysis, and an engineered fatty acid biosynthesis pathway. These sections could be managed separately, expanded and contracted within the SME (or other) viewer so that visualization could be more easily directed by the modeling biologist. SME implemented the reaction groups in conjunction with the LayoutNode class to enable hiding of reactions, reactants and reaction arrows. Actual visual contraction and expansion of groups hasn't been implemented.

Furthermore, compound reactions that contain multiple sub-reactions like the SigMech reactions (Chapter 3) could implement ReactionGroups to manage the sub-reactions. The class KMechReactionGroup was added for this purpose. The idea behind this design is that an enzyme mechanism consisting of multiple sub-reactions, conceptually can be seen as one reaction, and the details of the particular enzyme mechanism should be hidden. When the network of sub-reactions may be of interest, particularly when the mechanism is being modeled, the sub-reactions could be expanded. Ultimately, proper implementation of reactions groups should provide differing levels of pathway model examination.



CatalyticWithAllostericRegulation(F6P-F16P Bio) GeneralizedMWC(

Reaction Components
CatalyticWithAllostericRegulation: F6P-F16P Bi
ProductsEnzyme: F16P
ProductsEnzyme: ADP
ActivatorsEnzyme: F26bP
ActivatorsEnzyme: AMP
Enzyme: PFK
SubstratesEnzyme: F6P
SubstratesEnzyme: ATP

After entry of new value
Click 'Submit' to save value

Bio
Name: F6P-F16P Bio
ShortDescription:

Parameters
C: 0.1
ShortDescription:
Regulators: AMP, F26bP
Kcat: 950
Comment:
Enabled:
Name: F6P-F16P Math
Activators: AMP, F26bP
N: 8
L: 1.94
Inhibitors:

Start: 0 End: 500 MaxSteps: 25000

**Figure 2.41 The [Najdi 2011] fatty acid biosynthesis model contains portions of the pentose phosphate, glycolysis and fatty acid biosynthetic pathways. The implementation of reaction groups enable the hiding or highlighting of pathway sections and reaction arrows to more clearly show portions of the overall model. This particular model has been organized fairly well but other more interconnected models can become cumbersome to visualize. Commonly connected reactant molecules such as ATP can generate many edges in a network that can quickly clutter network visualization.**

## 2.9.3 ***Post Translational Modification Hierarchy***

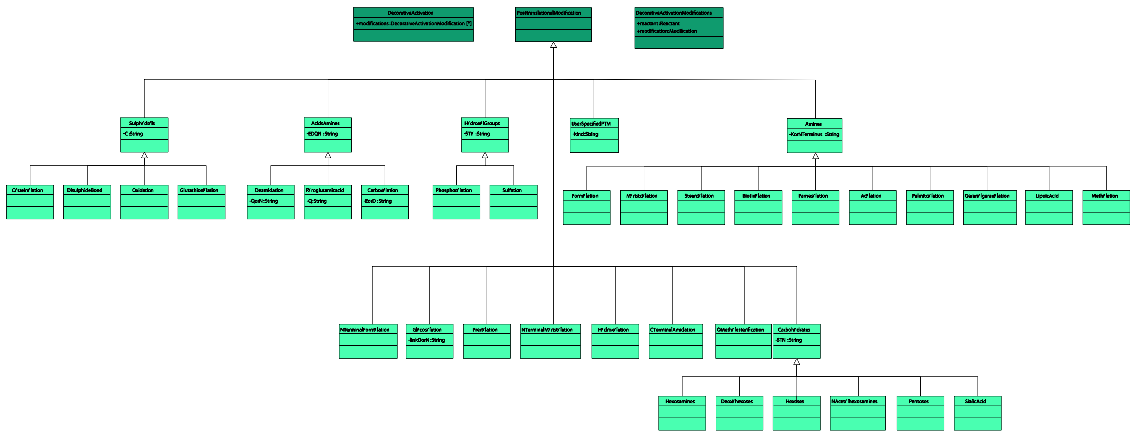


**Figure 2.42 The post translational modification hierarchy contains a structured organization of common chemical modifications of protein amino acid residues. Since this figure is difficult to read a comprehensive list of these modification classes is provided in Appendix 0.**

The Post-translational hierarchy consists of common chemical modifications of protein amino acid residues. Since Figure 2.42 is difficult to read, a list of posttranslational modification classes is provided in Appendix 0. These modifications are important as they have functional impact upon the protein. This domain of modifications was included in the schema because two sequence-identical proteins can have many functional differences. Saving separate instances for the same protein with different modification states in the database wouldn't be feasible, as a single protein may have multiple modification types over numerous locations. The cross product of modification states becomes enormous quickly, and they can't be stored in the database as individual proteins. A more feasible approach would be to store a protein base form and to "decorate" the protein with modifications. A corresponding table of states could then be attached to the particular reactant. The post translational modification hierarchy of modification should integrate with Section 2.6.1 concerning state vectors and modifications.

Although this hierarchy appears in the reaction diagram, this set of modifications does not inherit class attributes of the Reaction base class. Post-translational modifications can be modeled as either a state property of a polypeptide or a biological reaction process. For instance, examination of the phosphorylation of a protein by a particular enzyme, can take differing perspectives. If our goal is to address the functional and regulatory properties of the protein itself by examining the state of the protein, in this case the quantity and location of the phosphorylation sites, will yield information about the enzymes functional activity. Another perspective is to view the biological process of phosphorylation as a biochemical event with other Reactants participating and playing various roles such as catalyst and substrate i.e. a kinase attaching a phosphate group to the protein. Since the Gene Ontology (GO) classes in Sigmoid (for greater detail see Section 2.11.1) can already provide a gateway to compilations of data concerning cellular processes and functional events, developing an independent ontology of these events may not be the most efficient use of resources for Sigmoid development. If however, we wish to develop a system of examining the state of a particular polypeptide, in terms of the product of multiple post-translational modifications, a system of recording the state of the polypeptide must be implemented. The modifications in terms of state are treated as decorative properties of proteins. If this is to be the approach, the PostTranslationalModification hierarchy either should be more closely tied to the Reactant diagram or, deserves its own diagram and should be relocated to a

"Modification" class diagram, perhaps to be shared with other potential macromolecule type decorations.

## *2.10 Knowledge Sources*

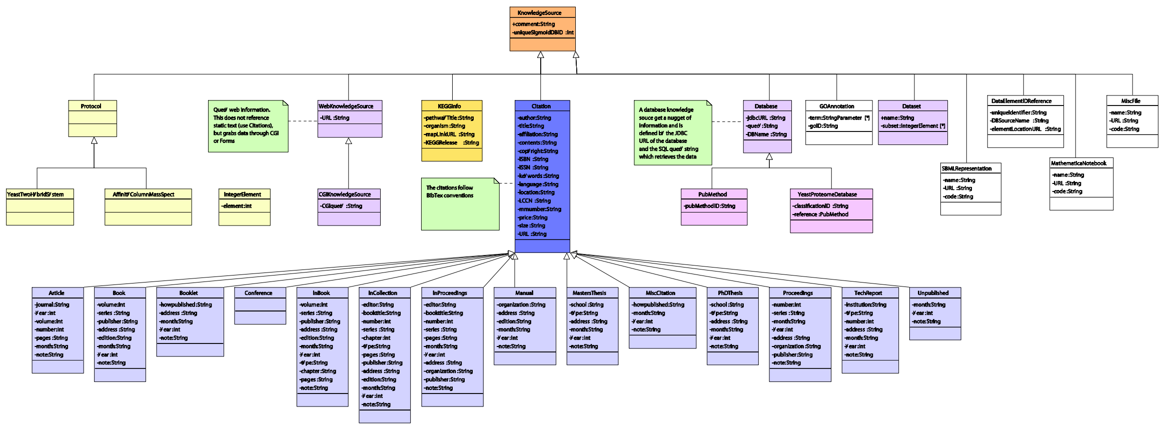


**Figure 2.43 The entire KnowledgeSource class diagram consists of data classes designed to provide citation information and documentation references for Sigmoid Models, Reactants and Reactions. The Citation domain of subclasses (Blue and purple classes) provide references to model research publications. The Knowledgesource classes are expanded for legibility in subsequent figures.**

Knowledge sources are designed to provide investigators with references to relevant Model, Reactant, and Reaction documentation. The KnowledgeSource class diagram is illustrated in Figure 2.43. A large subclass of KnowledgeSource is the Citation class (illustrated in more detail in Figure 2.45). The Citation class and its subclasses follow BibTex conventions for document citation. Storing relevant model citation information provides investigators with valuable access to previous research. In addition, proper population of the Sigmoid database with publication information, and some integration with word processing utilities like Latex could facilitate research publication of pathway models. No features of this sort have been implemented for

Sigmoid so its integration with word processing utilities remains an open problem. Some KnowledgeSource classes such as the Citationn MathematicaNotebook and MiscFiles classes are accessed and displayed by the model browse feature of the sigmoid website at www.sigmoid.org as shown in Figure 2.46.

The Citation class (shown in Figure 2.45) has been implemented for storing relevant publication information associated with user constructed Sigmoid models [Cheng *et al.*2005], and has been integrated with the Sigmoid web search utility at www.sigmoid.org. Relevant model publication reference information for Sigmoid models is available in the model browsing section illustrated in Figure 2.46. The MathematicaNotebook class includes URLs and file information for associated Model notebooks. This is an important reference since all of the persistent models currently in the database possess associated Mathematica notebooks. These notebooks are also linked through www.sigmoid.org.

Classes designed to access external pathway markup language documents are present in the KnowledgeSource tree (See Figure 2.44). Kyoto Encyclopedia of Genes and Genomes (KEGG) pathways are referenced through the KEGGInfo class. Attributes for KEGG pathways include Universal Resource Locators (URLs), pathway titles, associated organisms and the KEGG release number. Similar classes are available for, Gene Ontologies (GOAnnotation class) and the Systems Biology Markup Languge (SBML, SBMLRepresentation class).

WebKnowledgeSources are simple URL wrapper classes to reference web data. A subclass of WebKnowledgeSource is CGIKnowledgeSource and contains fields to perform CGI queries. The MiscFile class is a useful class for storing miscellaneous files associated with pathway models or model components.

The Database Class includes a “jdbcURL” and fields for associated database queries. Database opens up queries to other bioinformatics databases that implement the Java Database Connectivity Application Programming Interface (JDBC API). The JDBC API is the designed for database-independent connectivity between Java applications and a wide range of databases including SQL databases (Oracle 2012).

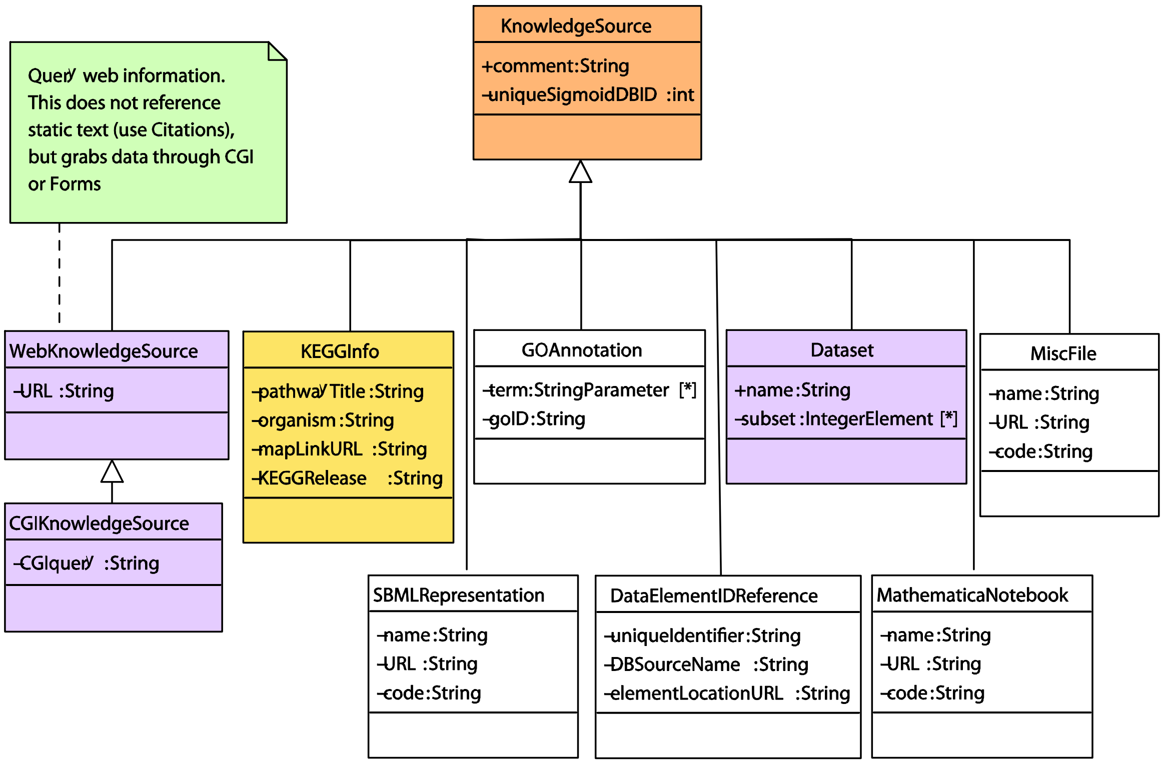


**Figure 2.44 A subset of KnowledgeSource (KS) subclasses are designed for references to external databases, files and markup languages. KEGGInfo is a KS for the Kyoto Encyclopedia of Genes and Genomes. SBMLRepresentation is a KS forthe Systems Biology Markup Language. KSs are provided for Universal Resource Locators (WebKS). Miscellaneous files and Mathematica notebooks associated with pathway models stored on the Sigmoid servers are referenced by the MiscFile and MathematicaNotebook classes. These MiscFiles and MathematicaNotebooks are accessed and displayed by the Sigmoid website.**

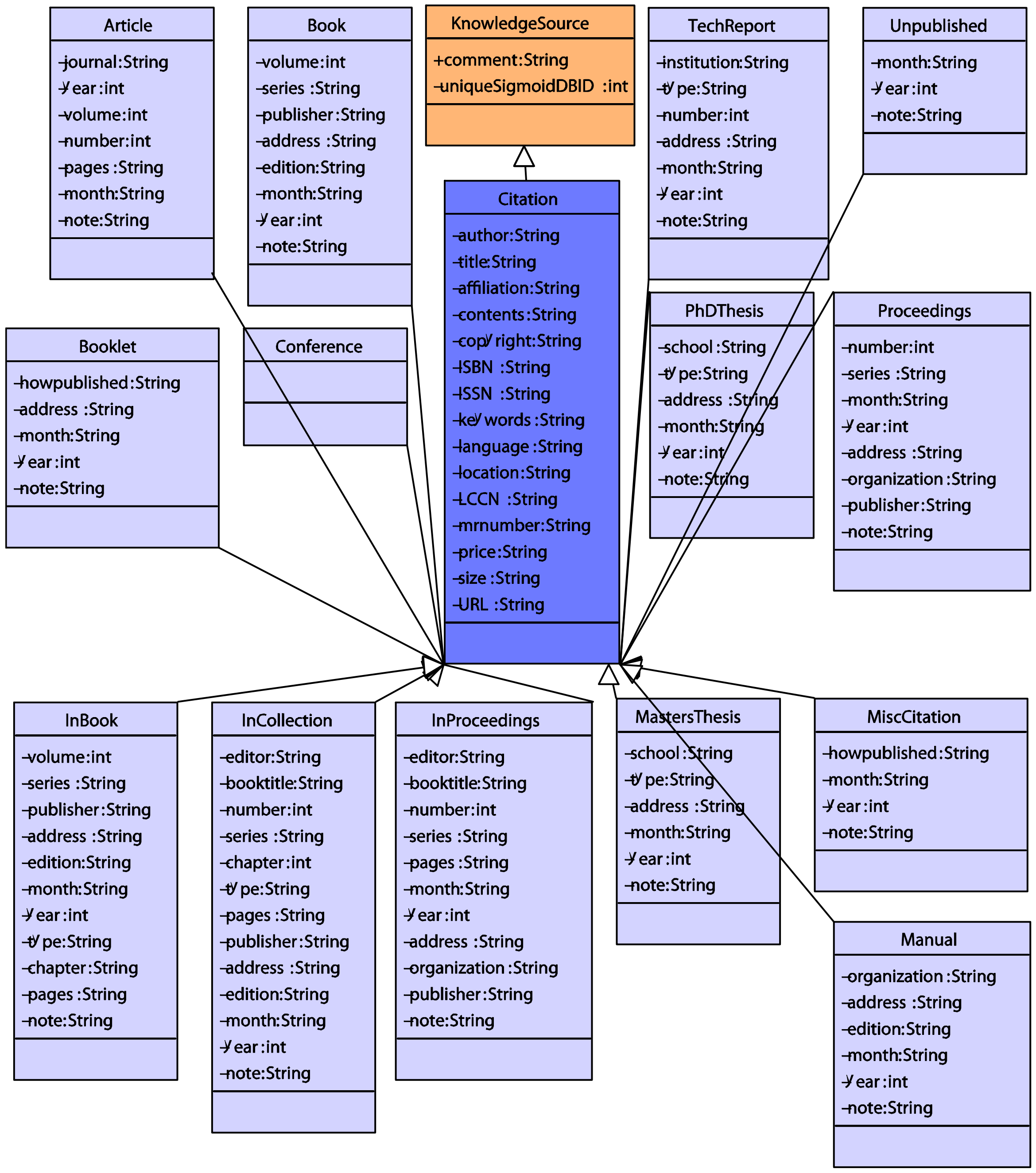
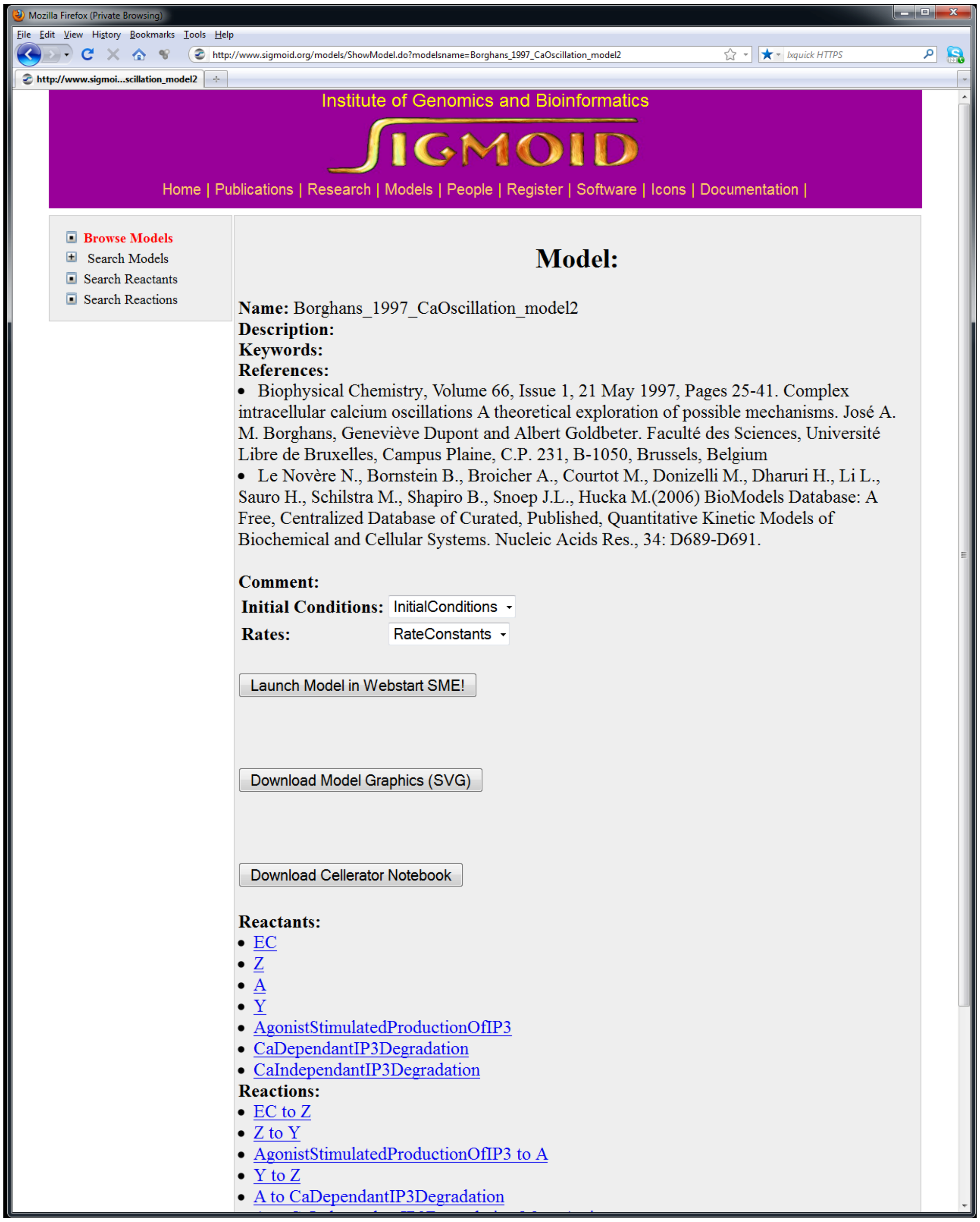


**Figure 2.45 The Citation KnowledgeSource provides a set of subclasses designed for publication citation information that follow the BibTex conventions [Cheng *et al.*2005]. Citations for model pathways are accessed and displayed by the Sigmoid website the browse models section.**


Mozilla Firefox (Private Browsing)
File Edit View History Bookmarks Tools Help
http://www.sigmoid.org/models/ShowModel.do?modelsname=Borghans_1997_CaOscillation_model2
http://www.sigmoi...scillation_model2
Institute of Genomics and Bioinformatics
SIGMOID
Home | Publications | Research | Models | People | Register | Software | Icons | Documentation |
Browse Models
Search Models
Search Reactants
Search Reactions

**Model:**

**Name:** Borghans_1997_CaOscillation_model2
**Description:**
**Keywords:**
**References:**

- Biophysical Chemistry, Volume 66, Issue 1, 21 May 1997, Pages 25-41. Complex intracellular calcium oscillations A theoretical exploration of possible mechanisms. José A. M. Borghans, Geneviève Dupont and Albert Goldbeter. Faculté des Sciences, Université Libre de Bruxelles, Campus Plaine, C.P. 231, B-1050, Brussels, Belgium
- Le Novère N., Bornstein B., Broicher A., Courtot M., Donizelli M., Dharuri H., Li L., Sauro H., Schilstra M., Shapiro B., Snoep J.L., Hucka M.(2006) BioModels Database: A Free, Centralized Database of Curated, Published, Quantitative Kinetic Models of Biochemical and Cellular Systems. Nucleic Acids Res., 34: D689-D691.

**Comment:**
**Initial Conditions:** InitialConditions
**Rates:** RateConstants

Launch Model in Webstart SME!
Download Model Graphics (SVG)
Download Cellerator Notebook

**Reactants:**
- EC
- Z
- A
- Y
- AgonistStimulatedProductionOfIP3
- CaDependantIP3Degradation
- CaIndependantIP3Degradation

**Reactions:**
- EC to Z
- Z to Y
- AgonistStimulatedProductionOfIP3 to A
- Y to Z
- A to CaDependantIP3Degradation

**Figure 2.46 Some KnowledgeSource information such as Model Citations are accessible through the "Browse Models" utility at www.sigmoid.org. Model xCellerator notebooks and model graphic files, which are also referenced by KnowledgeSource, are also accessible there.**

## *2.11 Additional supporting classes*

### 2.11.1 *Gene Ontologies*

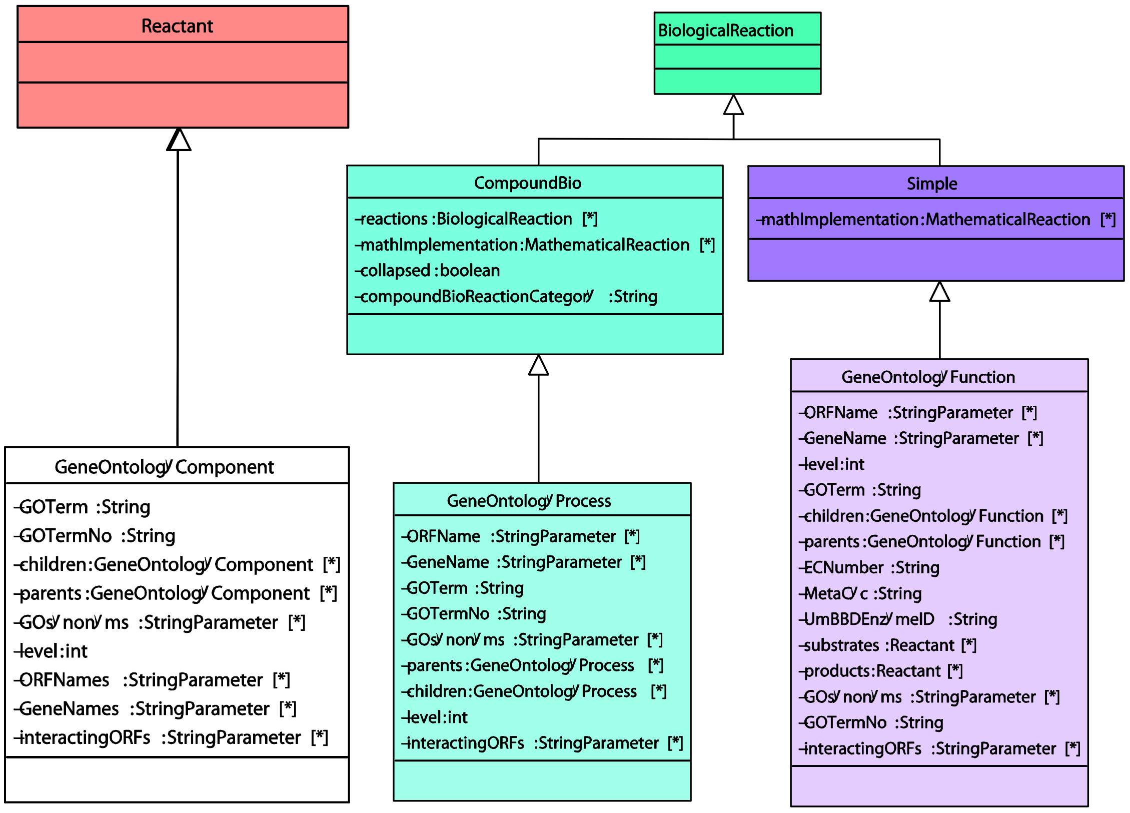

Reactant
BiologicalReaction
CompoundBio
-reactions :BiologicalReaction [*]
-mathImplementation:MathematicalReaction [*]
-collapsed :boolean
-compoundBioReactionCategory :String
Simple
-mathImplementation:MathematicalReaction [*]
GeneOntologyComponent
-GOTerm :String
-GOTermNo :String
-children:GeneOntologyComponent [*]
-parents :GeneOntologyComponent [*]
-GOSynonyms :StringParameter [*]
-level:int
-ORFNames :StringParameter [*]
-GeneNames :StringParameter [*]
-interactingORFs :StringParameter [*]
GeneOntologyProcess
-ORFName :StringParameter [*]
-GeneName :StringParameter [*]
-GOTerm :String
-GOTermNo :String
-GOSynonyms :StringParameter [*]
-parents :GeneOntologyProcess [*]
-children:GeneOntologyProcess [*]
-level:int
-interactingORFs :StringParameter [*]
GeneOntologyFunction
-ORFName :StringParameter [*]
-GeneName :StringParameter [*]
-level:int
-GOTerm :String
-children:GeneOntologyFunction [*]
-parents :GeneOntologyFunction [*]
-ECNumber :String
-MetaCyc :String
-UmBBDEnzymeID :String
-substrates :Reactant [*]
-products:Reactant [*]
-GOSynonyms :StringParameter [*]
-GOTermNo :String
-interactingORFs :StringParameter [*]

**Figure 2.47 The Sigmoid schema implements classes for a Gene Ontology representation. The GeneOntologyComponent class (On left) has been implemented as a subclass of Reactant because the GO component most closely correlates to biological objects. The GeneOntologyFunction (On right) has been implemented as a subclass of Simple BioReactions because of its close correlation to simple biological reaction processes. The GeneOntologyProcess class (middle) has been implemented as a subclass of CompoundBio reactions because GO processes, which are composed of sets of GOFunctions, most closely parallel the complex biological processes represented by CompoundBio reactions.**

Classes have been incorporated into the Sigmoid data structures to build compatibility with the Gene Ontologies Project (GO). The GO project is intended to unify a controlled-vocabulary representation of annotated gene and gene product attributes across species. In addition, the GO project provides a set of tools to access and process GO data. The GO project provides ontology over three domains: 1) The Cellular Component domain represents cellular components or elements of cellular structures. Cellular components are not only (single or conglomerates) gene products, but the gene products existing within the context of a larger cellular structure thus establishing it being a component. 2) Molecular functions exist to represent molecular level activities of generally single gene products. Gene product functions are what tasks the gene products perform or what roles they play within a cell. Catalysis events and protein activities are included within this scope. 3) Biological processes in GO represent series of events that are composed of sets of GO molecular functions. The GO definition states that a process should have a distinct beginning and end. Also, processes can be composed of processes.

GO components most closely map to Reactants in the Sigmoid schema as they are biological objects and with discretion can be treated as Sigmoid Reactants. Some care must be used because some GO components can be large, complicated structures. For example a cell membrane is a potential GO component, that is a child to a cell, and a parent to cell membrane proteins in the GO hierarchy. Sigmoid's mathematical reaction equations are designed mostly for molecular biochemical reactions, so it is left to the expert user to either determine applicability or extend the computational features of the simulator to fit the context with which a GO component would be applied. GO functions which are molecular activities most closely resemble Simple reactions therefore have been implemented as a subclass of Simple BioReaction. There are many protein activities available in the GO datasets that could be represented by the set of existing

Cellerator functions. The GO processes have been set to inherit from the CompoundBio Reaction class as they consist of sets of GO functions just as CompoundBio Reactions consists of sets of BioReactions. Again discretion must be applied by the expert user as some GO processes do not directly map to the simulation capabilities of xCellerator. For example, a muscle fiber contraction is a potential GO process, which of course involves a network of biochemical reactions. A fiber contraction has other process attributes such as spatial dynamics. These dynamics within the context of the GO term could be the primary focus of the term. As spatial modeling in Sigmoid is in its infancy, modeling considerations and development would have to be made to model a muscle fiber contraction. None the less, the three GO categories of classification share enough commonalities with their positions within the schema that they can be of use for a modeling biologist.

### 2.11.2 ***Spatial and Compartmental Modeling***

We have added to Sigmoid a representation intended to support the development of compartmental pathway models. This representation may be used for developmental modeling and simulation, or for the modeling of spatially extended systems more complex than a simple “well mixed” model of the locus of a pathway. Although the representation is intended to be more comprehensive and flexible, a basic support class structure exists to produce models for the Cellzilla (www.xlr8r.info) modeling tool available with xCellerator.

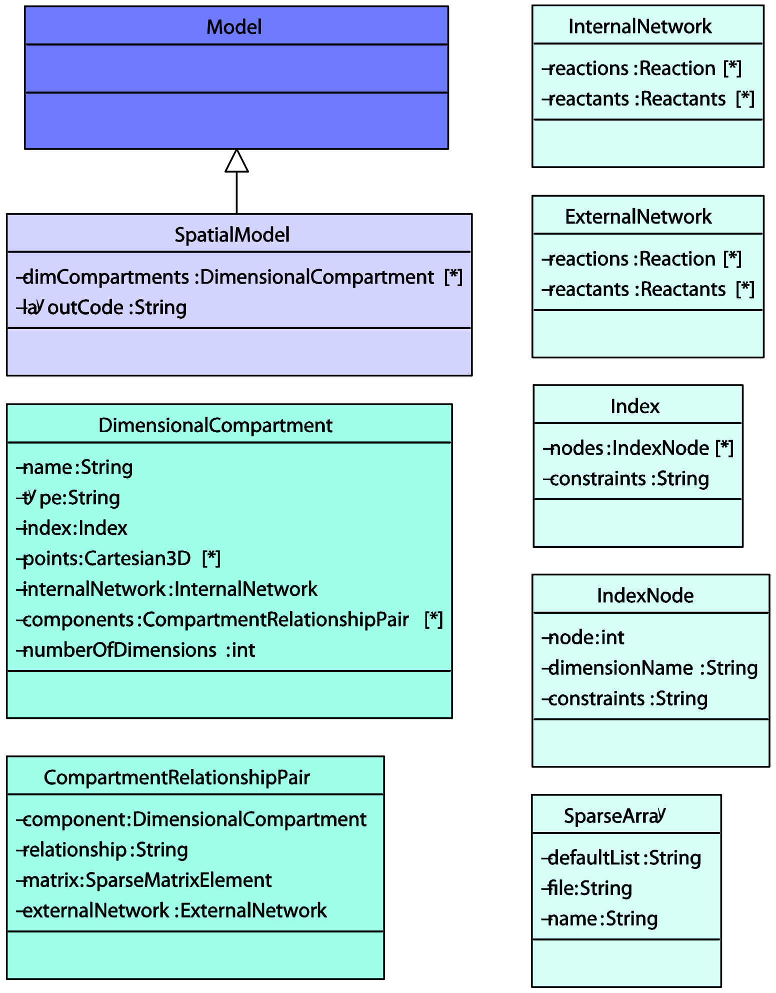


**Figure 2.48 Classes to support Spatial and Compartmental Models are located in the Model class diagram. SpatialModel is a subclass of Model and inherits all Models' attributes. A Spatial model is composed of DimensionalCompartments (DCs) that possess sets of InternalNetworks and ExternalNetworks. InternalNetworks are composed of sets of Reactants and Reactions that are contained within particular DCs. ExternalNetworks have Reactants and Reactions that participate across DCs. DCs have references for Catesian coordinates to incorporate spatial components of n dimensions to the model if necessary. DCs have CompaortmentRelationshipPairs to specify relationship specificity between DCs. Since Cellzilla utilizes indices for multi-compartmental reaction modeling, a set of Index, IndexNode and SparceArray classes are associated with DCs to reference DCs for simulation.**

SpatialModel inherits all the class attributes of a model. The main distinction is that it contains DimensionalCompartments. The intention behind spatial models is to provide a framework to construct objects for the representation of biological structures, systems, and processes. The classes associated with SpatialModel may be used to construct geometric and multicompartmental biochemical representations for the SpatialModel.

DimensionalCompartments (DC) may be an abstract compartment or may have concrete spatial Cartesian coordinates associated with them. Dimensional compartments were constructed to be of any number of dimensions, for flexibility. DCs can consist of points, edges, surfaces, volumes and even potentially higher dimensional solids. Each DC may possess an InternalNetwork of reactions and reactants. The InternalNetwork reactions operate solely within a DimensionalCompartment. Possessing an InternalNetwork is akin to having a self contained Sigmoid Model within the compartment. Of course, independent DCs can have independent copies of the same InternalNetwork. For instance, one could build an array of cells with each possessing an independent copy of an internal set of cellular reaction processes and associated reactant species.

An attribute of DimensionalCompartiments is that each DC may have multiple CompartmentRelationshipPairs (CRPs). The CRPs are to serve as the framework for links and associations between DCs. The "relationship" attribute in the CRP class is left as a string to allow for an open-ended representation. We have defined some straightforward relationships that will be used in SpatialModels. A "component" relationship serves to define possession for an object for building a geometrical hierarchy of possession. For instance, let us assume we're building a rectangular solid to represent a cell. Each edge, surface and point within the spatial (geometric) construction of the object will be related to the cell (rectangular solid) as a "component" of that cell. The component relationship should allow us to construct a hierarchy of spatial objects.

Another type of “relationship” will be a “neighbor” relationship to indicate adjacency between neighboring DCs. Neighbor relationships will be useful for modeling biochemical phenomena such as diffusion. Let us extend our example of a cell to include several internal three-dimensional DCs to represent the cytosol of the cell. Diffusion inside the cell could be modeled by establishing neighbor relationships between these internal cellular compartments. Moreover, the neighbor relationship between adjacent cells could also serve to model intracellular diffusion across an array of cells. Also, “boundary” relationships will indicate a junction between neighboring DCs where potential biochemical interactions can take place. A hypothetical biological application of this would be to establish the exterior plane surfaces of our hypothetical cell as boundaries where intracellular signaling could take place. Every CRP has a “matrix” attribute which references a SparseMatrix to store references that specify adjacency. Each CRP also possesses an ExternalNetwork (EN) attribute field, although not all relationships between compartments, for example “component” relationships, will have an associated EN.

ExternalNetworks consist of a network of reactions each of which spans more than one compartment. Biological processes such as transport, diffusion, osmosis, and signaling can be expressed with ExternalNetworks. As with InternalNetworks, each ExternalNetwork will be attributed with its own corresponding set of Reactions and Reactants.

The InternalNetwork and ExternalNetwork theme mirrors the Cellzilla approach to model multicompartmental biochemical reaction networks. Since Sigmoid has built considerable support for the xCellerator platform, supporting Cellzilla, an xCellerator extension, is a logical choice in creating a multicompartmental simulator. Cellzilla uses indices on the symbolic reactants of xCellerator to represent compartment identities. In order to implement multiple compartments and reaction networks across compartments in Sigmoid, we required a scheme of implementing indices for our Reactant and Reaction classes.

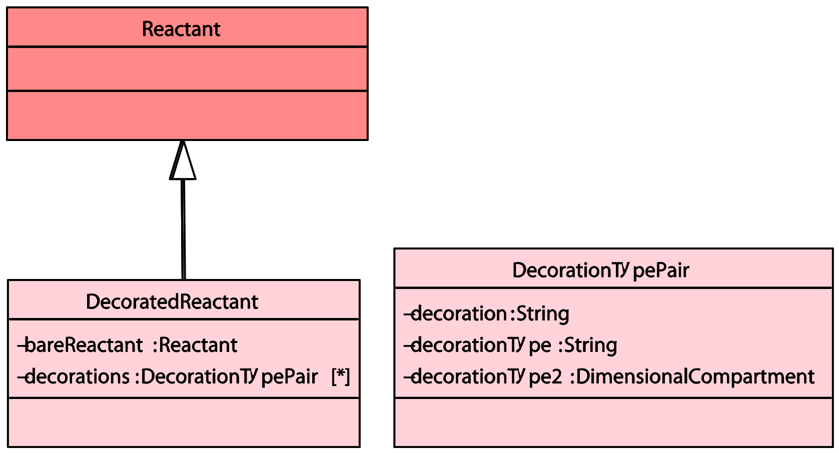


**Figure 2.49 DecoratedReactants contain a Reactant and a DecorationTypePair. The DecorationTypePair class attribute "decoration" is used to store indices to be processed for Cellzilla output. The "decorationType" attribute indicates the nature of the decoration for instance a "neighbor". The "decorationType2" attribute can establish a reference to a DimensionalCompartment.**

In order to represent compartment indices for reactants, we constructed a DecoratedReactant class that contains a Reactant and a DecorationTypePair (DTP). The DecorationTypePair class has a "decoration" field that can be used to store index variables that are passed to Cellzilla and indicate which compartment a Reactant is in. The "decorationType" attribute of DTP stores the nature (or relationship) of the decoration. An example would be a "neighbor" relationship for adjacent compartments. DecoratedReactants are used by the BiReplicatedReaction class to implement intercompartmental reactions.

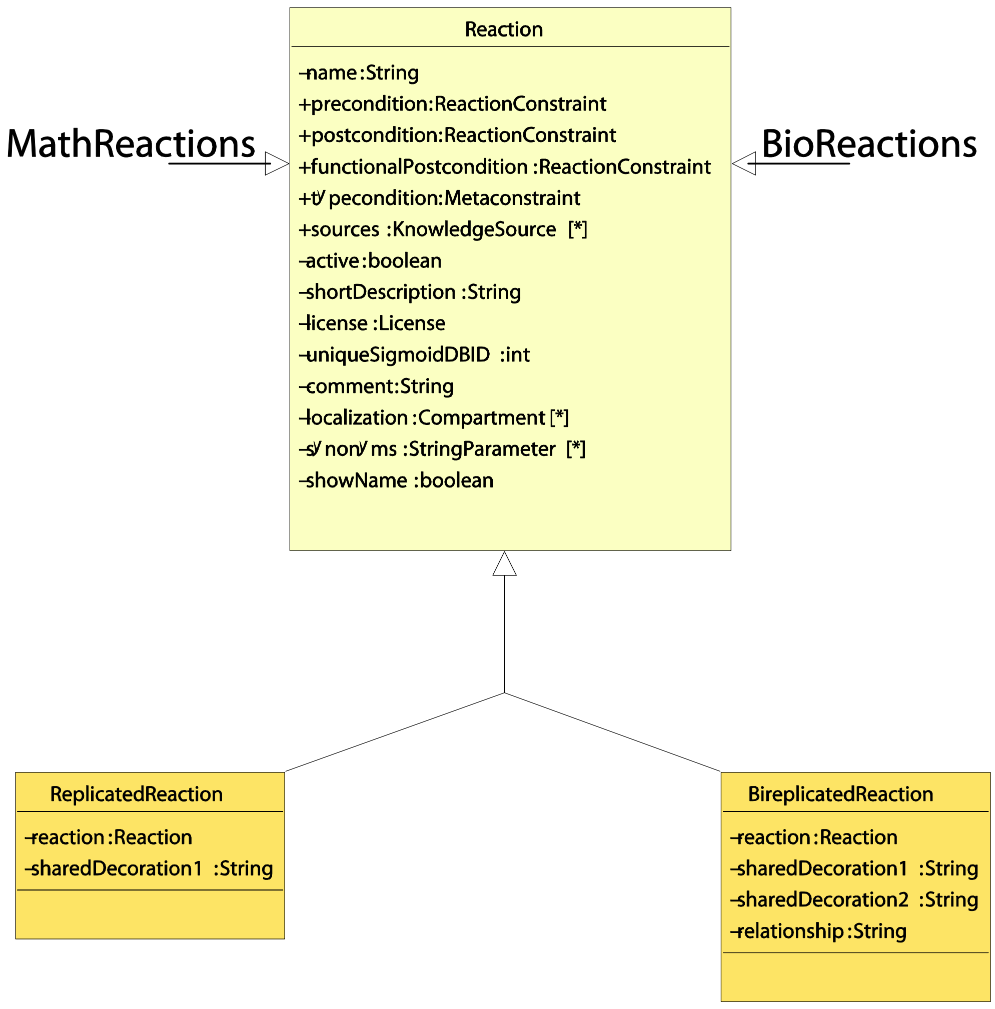

Reaction
-name:String
+precondition:ReactionConstraint
+postcondition:ReactionConstraint
+functionalPostcondition:ReactionConstraint
+typecondition:Metaconstraint
+sources:KnowledgeSource [*]
-active:boolean
-shortDescription:String
-license:License
-uniqueSigmoidDBID :int
-comment:String
-localization:Compartment [*]
-synonyms:StringParameter [*]
-showName:boolean
MathReactions
BioReactions
ReplicatedReaction
-reaction:Reaction
-sharedDecoration1 :String
BireplicatedReaction
-reaction:Reaction
-sharedDecoration1 :String
-sharedDecoration2 :String
-relationship:String

**Figure 2.50 The ReplicatedReaction and BireplicatedReactions were created to handle indices for multicompartmental SpatialModels. The "sharedDecorations" attributes are used to store indices for Cellzilla that correspond to compartments. For instance the "sharedDecoration" attributes of a BireplicatedReaction could be decorated with indices "i" and "j". The BireplicatedReactions' associated MathReaction could implement diffusion with a DecoratedReactant Rea[i] as a substrate getting converted to product Rea[j]. The change of index indicates a change in compartment.**

The ReplicatedReaction and BiReplicatedReaction classed were designed to implement indices for multicompartment Reactions. A BireplicatedReaction contains a Reaction and has fields for "sharedDecorations" that indicate the indexes for that particular reaction. The Reaction should possess DecoratedReactants as participants. The "sharedDecoration" fields store index variables that are to be passed to Cellzilla.

For instance the "sharedDecoration" attributes of a BireplicatedReaction could be decorated with indices "i" and "j". The BireplicatedReactions' associated MathReaction could implement diffusion with a DecoratedReactant Rea[i] as a substrate getting converted to product Rea[j]. The change of index indicates a change in compartment.

We have coded our first SpatialModel, the Shapiro_2008_WUS model, using the Sigmoid API. Shapiro_2008_WUS is discussed in Section 4.1.7.

## *2.12 Other Systems*

Other packages, such as VCell, Sigpath, and JDesigner for example, have functionalities that are similar to some of the features contained in Sigmoid. One notable example that shares similarity with the Sigmoid system is the Biomodels database (www.biomodels.net). The Biomodels database is maintained by the European Bioinformatics Institute (EBI). EBI is part of the part of the European Molecular Biology Laboratory (EMBL). Biomodels is an online database with a repository of SBML models, support for Gene Ontologies, web based visualization of models, and possess Mathematica simulation support for their SBML based models through MathSBML.

While it is sound to have a number of parallel efforts across multiple research groups, there are several features of the Sigmoid architecture that, in aggregate, position it uniquely within realm of the currently available systems biology software systems. Sigmoid introduced the web services framework [Cheng *et al.*2005] to create a truly distributed system. This flexible framework offers powerful modularity that, in conjunction with the generative nature of the Sigmoid coding cycle, offers a significantly reduced development time for integration of new components and data structures. Also, the OJB object relational bridge offers the advantages of oriented programming in conjunction with a relational database. Sigmoid capitalizes on the robust mathematical software tools and problem solving environment that Mathematica offers (along with the xCellerator/kMech packages designed to facilitate biological modeling via automated equation generation which other systems lack) while remaining open to other simulation and analysis tools.  Sigmoid defines its own set of biologically relevant classes as a foundation for its database and modeling capabilities. These classes are not constrained by conforming to exterior modeling formats such as SBML.  The synthesis of these features yields a flexible scalable architecture that not only allows for manageable, cost effective, adoption of new system components, but may open the ability to play within yet larger bioinformatics frameworks.

## *2.13 Conclusion*

We have successfully extended the Sigmoid schema to implement an object-oriented representation of parameters and parameter sets that work in conjunction with a set of MathematicalReaction classes to produce, when translated by the middleware, executable xCellerator notebooks.  We have expanded the Reactant hierarchy to provide a useful variety of biologically relevant objects that participate in biological processes across several levels of scale and complexity. We have extended and reorganized the variety of supported Reaction classes into sub-domains of Mathematical and Biological Reactions. The new hierarchy of Reactions allows for a flexible abstraction between the representation of biological processes and the mathematical functions that are used to simulate them. There are new KnowledgeSource classes that allow us to store model associated files and reference elements of external databases.

We have added schema support for Gene Ontologies in the form of a GOComponent class in the Reactant hierarchy, and the GOFunction and GOProcess Reactions in the Reaction hierarchy. These new classes allow us to support and incorporate specialized ontological external datasets (see section 4.2.4) into Sigmoid.

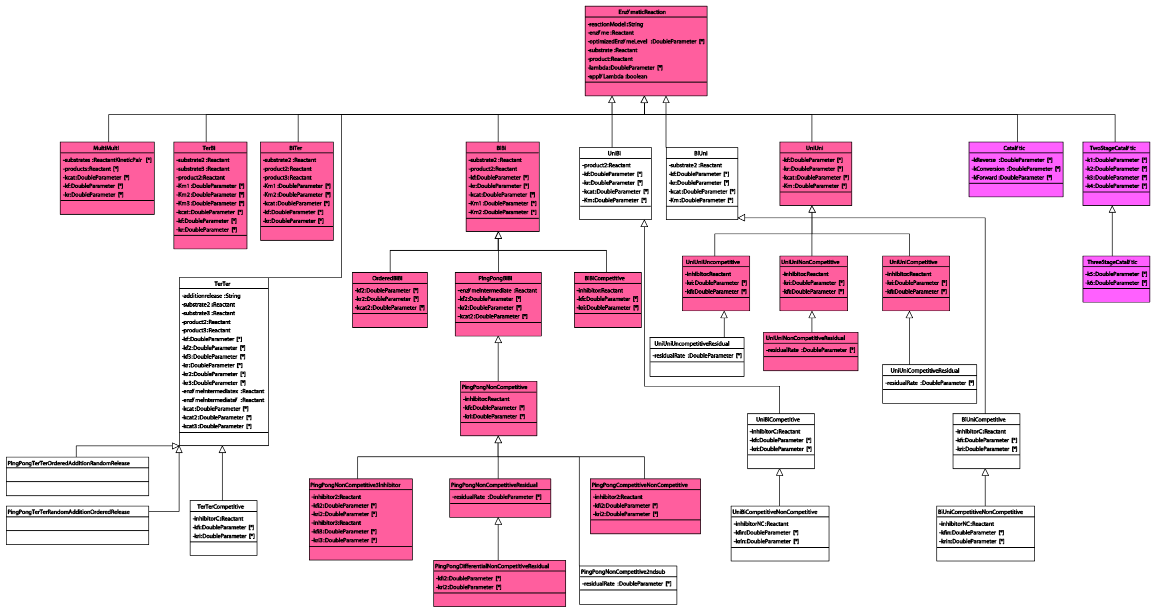
Finally, we've constructed the SpatialModel and associated classes that allow for the construction of geometric and compartmental objects to be used in Sigmoid. Multicompartmental models can be constructed that have internal reaction networks and intercompartmental reaction networks. As a first step for compatibility with Cellzilla, these internal and external networks can be sent through a next version of the middleware for simulation in Cellzilla.

# Chapter 3. Enzyme Mechanism Representation: SigMech

The set of currently available kMech reaction mechanisms has grown from just a few initial reactions to a large array of descriptive enzymatic mechanisms. Over the years, 35 enzymatic reaction mechanisms have been explicitly created. One version of the enzymatic class hierarchy of the schema is shown for illustrative purposes in Figure 3.1.

In order to function within the Sigmoid framework, a request for a kMech reaction case would yield a requirement to process a schema, generate the Sigmoid Java API classes, instantiate a new postgreSQL database, repopulate the database with model code, set up a development environment for the particular schema release within Eclipse (or equivalent), update the middleware Sigmoid-to-Cellerator code with the new kMech translation, retask SME with the new Java API and test all these steps. This development cycle, although designed for “rapid” development, still requires considerable personnel resources.

Common motifs can be seen in the Compound Enzyme reactions that express how biologists approach the construction of these mechanisms. For instance, a biologist might first describe and enzymatic reaction in terms of how many substrates and products it has. For a simple case we will construct a reaction with two substrates and two products. This is commonly called a BiBi reaction. The convention for this nomenclature follows uni, bi, ter, tet etc. This is typically how a reaction is first described. The convention has expanded to describing reactions as UniBi, BiUni, TerTer, and so forth. As more concepts are added to visualize or model a potential reaction, the more parameters of description will apply. For instance the reaction might be of type ping-pong (described later in this chapter.) We will show that since these enzymatic reaction mechanisms are built upon concepts, their descriptors can be identified as a set of motifs.



EnzymaticReaction
MultiMulti
TerBi
BiTer
BiBi
UniBi
BiUni
UniUni
Catalytic
TwoStageCatalytic
ThreeStageCatalytic
TerTer
OrderedBiBi
PingPongBiBi
BiBiCompetitive
UniUniUncompetitive
UniUniNonCompetitive
UniUniCompetitive
UniUniUncompetitiveResidual
UniUniNonCompetitiveResidual
UniUniCompetitiveResidual
PingPongNonCompetitive
UniBiCompetitive
BiUniCompetitive
PingPongTerTerOrderedAdditionRandomRelease
PingPongTerTerRandomAdditionOrderedRelease
TerTerCompetitive
PingPongNonCompetitive3Inhibitor
PingPongNonCompetitiveResidual
PingPongCompetitiveNonCompetitive
UniBiCompetitiveNonCompetitive
BiUniCompetitiveNonCompetitive
PingPongDifferentialNonCompetitiveResidual
PingPongNonCompetitive2ndsub

**Figure 3.1 The Enzymatic MathematicalReactions section of the Sigmoid Schema consists of compound MathReactions that can be represented more compactly. The three reactions in purple on the right are xCellerator compound enzyme reactions**

We have developed a parameterized input notation for enzymatic mechanisms that encompasses the motifs seen in the mathematical enzymatic reactions section of the Sigmoid schema. In addition, a set of computational processes should produce the requisite set of sub-reactions that compose the corresponding mechanism. The parameterization and set of corresponding processes to generate mechanism reactions should achieve a few things. First it should most accurately represent the concepts with which a biologist constructs his enzymatic model and reflect a simple syntactic compatibility to his/her description. That way the selection of a reaction mechanism will be straight forward for the modeling biologist.  Second, a general form, the ”parameterized representation”, and a corresponding set of reaction generation procedures reduces coding iterations in the development cycle of our software system. The development effort savings should prove considerable. In addition, the ever growing number of classes required to represent enzymatic mechanisms in the schema can be reduced to a more elegant representation, as demonstrated in Figure 3.2. The new classes effectively solve the problem of the representation, within the scope of certain parameters that will be discussed below, thus eliminating the need for frequent updates. Such a generalization also increases the level of abstraction and allows us to focus our efforts to expand the representation on the motif level.  There may be more or new patterns of biological relevance that may extend this approach.

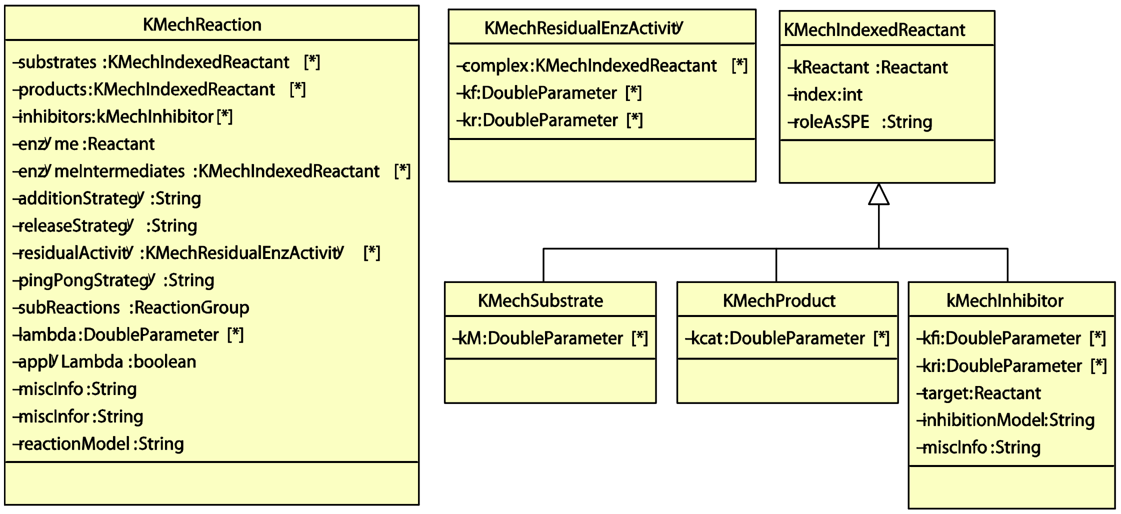


**Figure 3.2 The new KMech representation for enzymatic mechanisms.  This set of data classes should encompass all previous kMech compound reaction classes and many new cases of kMech reactions that follow previous patterns.**

A set of motifs that can be used as parameters of an input notation for procedurally generating a family (or set) of reaction mechanisms will be described.  All of the existing kMech reactions mechanisms and correlate Sigmoid schema representations thereof (as of this writing) are encompassed by this set. We will name this parameterized set of inputs, data structure, and corresponding set of Reaction generation procedures as SigMech. “SigMech” is a contraction of Sigmoid enzyme Mechanism. The data classes currently are termed with kMech terms. This reflects the fact that these classes were constructed to encompass all the kMech classes. The utility as a whole, providing additional functionality, will be called SigMech. The SigMech utility, including all the procedures it uses to generate reaction mechanisms, has been written with the Java language using the Sigmoid API.

We can construct an enzyme mechanism with the SigMech utility by implementing procedures for an enzymatic mechanism in three steps. First we should look at how we wish to add substrates to an enzyme to form an enzyme substrate complex. Second, we should determine how these bound substrates are converted to product. Third, we should determine how products are to be released from complex.

## *3.1 Addition Strategy*

The patterns apparent in the kMech utility classes in the schema fall into three categories. The three categories of reactant addition to enzymatically bound complex are full addition, an ordered addition or, random addition strategy.

The first motif we will address is the way in which the enzyme binds its substrates into a complex. From here, depending on the context, the pattern with which substrates are added to complex will be either referred to as an addition “strategy” in cases where we’re describing the manner in which we’re constructing the elementary reaction equations, or an addition “phase” when referring to that strategy as a portion of the entire reaction mechanism.

Consideration of previous kMech enzyme mechanisms revealed the three distinct patterns of substrate enzyme complex formation that were just introduced. Parallel and independent research also revealed these patterns of addition and applied them to an oxidation reduction software supplement to kMech notebooks called RedoxMech [Chang 2011]. It appears that these patterns weren’t implemented in any form of a release phase or conversion strategy (to be introduced in Sections 3.2 and 3.3) in RedoxMech.

These patterns or strategies developed for implementation in SigMech will be detailed below in a kMech notation to illustrate the elementary reactions:

### 3.1.1 ***Full Addition***

A full addition strategy indicates that all available substrate types are bound by the enzyme in one reaction step. This motif will be termed “full addition”. For this example the kMech notation [Yang 2005] for a BiBi reaction would appear as:

$$Enz[\{\mathrm{S1\_}, \mathrm{S2\_}\} \overset{\mathrm{En\_}}{\rightleftarrows} \{\mathrm{P1\_}, \mathrm{P2\_}\}, \mathrm{BiBi}[\mathrm{kf\_}, \mathrm{kr\_}, \mathrm{k\_}]] := \{\{\mathrm{S1} + \mathrm{S2} + \mathrm{En}$$

$$\rightleftarrows \{\mathrm{NC}[\mathrm{En}, \mathrm{S1}, \mathrm{S2}], \mathrm{kf}, \mathrm{kr}\}, \{\mathrm{NC}[\mathrm{En}, \mathrm{S1}, \mathrm{S2}] \rightarrow \mathrm{En} + \mathrm{P1} + \mathrm{P2}, k\}\};$$

The kMech notation NC[En,a,b] indicates an enzyme complexed with a and b. kf, kr and k are rate parameters.

The addition phase of this reaction is:

$$\{\mathrm{NC}[\mathrm{En}, \mathrm{S1}, \mathrm{S2}], \mathrm{kf}, \mathrm{kr}\}$$

where all available substrates are bound in one reaction.

If this were a TerBi reaction with an input such as the following,

$$Enz[\{\mathrm{S1\_}, \mathrm{S2\_}, \mathrm{S3\_}\} \overset{\mathrm{En\_}}{\rightleftarrows} \{\mathrm{P1\_}, \mathrm{P2\_}\}, \mathrm{TerBi}[\mathrm{kf\_}, \mathrm{kr\_}, \mathrm{k\_}]] :=$$

Then the addition reaction (product generation omitted) would appear as

$$\{\{\mathrm{S1} + \mathrm{S2} + S3 + \mathrm{En} \rightleftarrows \mathrm{NC}[\mathrm{En}, \mathrm{S1}, \mathrm{S2}, \mathrm{S3}], \mathrm{kf}, \mathrm{kr}\}$$

For any number n of substrates the full addition pattern of substrates translates to:

$$Enz[\{\mathrm{S1}_{,\mathrm{S2}_,}.,.,\mathrm{Sn}\ \} \overset{\mathrm{En}}{\rightleftarrows} \{\mathrm{P1}_{,\mathrm{P2}}\ \}, \mathrm{NBi}[\mathrm{kf\_}, \mathrm{kr\_}, \mathrm{k\_}]] := $$

$$\{\{\mathrm{S1} + \mathrm{S2} + \ldots + Sn + \ \mathrm{En} \rightleftarrows \mathrm{NC}[\mathrm{En}, \mathrm{S1}, \mathrm{S2}, ., ., \mathrm{Sn}], \mathrm{kf}, \mathrm{kr}\},$$

## 3.1.2 ***Sequential Addition***

Sequential (ordered) addition is the case in which the enzyme binds a particular substrate followed by sequentially specific substrates until all available substrates are bound.

A BiBi example with ordered addition (conversion and release reactions omitted) would appear as:

$Enz[\{\mathrm{S1\_}, \mathrm{S2\_}\} \overset{\mathrm{En}}{\rightleftarrows} \{\mathrm{P1\_}, \mathrm{P2\_}\}, \mathrm{OrderedBiBi}[\mathrm{kf1\_}, \mathrm{kr1\_}, \mathrm{kf2\_}, \mathrm{kr2\_}, \mathrm{k\_}]] :=$

$\{\{\mathrm{S1} + \mathrm{En} \rightleftarrows \mathrm{NC}[\mathrm{En}, \mathrm{S1}], \mathrm{kf1}, \mathrm{kr1}\},$

$\{\mathrm{S2} + \mathrm{NC}[\mathrm{En}, \mathrm{S1}] \rightleftarrows \mathrm{NC}[\mathrm{En}, \mathrm{S1}, \mathrm{S2}], \mathrm{kf2}, \mathrm{kr2}\},$

The number of reactions necessary to fully complex all the substrates sequentially with the enzyme will be equal to the number of substrates. The sequential addition yields n number of reactions for n number of substrates.

Expanding out the example for n substrates,

$$Enz[\{\mathrm{S1\_}, \mathrm{S2\_}, ., \mathrm{Sn}\} \overset{\mathrm{En}}{\rightleftarrows} \{\mathrm{P1\_}, \mathrm{P2\_}\ \},$$

$$\mathrm{OrderedAdditionNBi}[\mathrm{kf1\_}, \mathrm{kr1\_}, \mathrm{kf2\_}, \mathrm{kr2\_}, \mathrm{k\_}.\,.\,\_, ., \mathrm{kfn\_}, \mathrm{krn\_}\ ]] :=$$

$\{\{\mathrm{S1 + En \rightleftarrows NC[En, S1], kf1, kr1}\},$

$\{\mathrm{S2 + NC[En, S1] \rightleftarrows NC[En, S1, S2], kf2, kr2}\},$

$\{\mathrm{S.. + NC[En, S1, S2] \rightleftarrows NC[En, S1, S2, S..], kf.., kr..}\}, \ldots$

$\{\mathrm{Sn + NC[En, S1, S2, S \ldots] \rightleftarrows NC[En, S1, S2, S\ldots, Sn], kfn, krn}\}$

## 3.1.3 ***Random Addition***

In the case of a random strategy, the ordering of substrates is disregarded. The enzyme will bind any available substrate for the first subreaction. From that state, the enzyme/substrate complex will randomly choose an available substrate progressively until all substrates are bound. In the example below we have an example of a kMech reaction expressed by xCellerator input notation and kMech expansion.

$$\mathrm{Enz[\{S1\_, S2\_, S3\_\} \overset{En}{\rightleftarrows} \{P1\_, P2\_, P3\_\}, RandomTerTer[kf1\_, kr1\_, kf2\_, kr2\_, kf3\_, kr3\_, k\_]]}$$
$$:= \{$$
$$\begin{gathered}
\{\mathrm{S1 + En \rightleftarrows NC[En, S1], kf1, kr1}\},\\
\{\mathrm{S2 + En \rightleftarrows NC[En, S2], kf2, kr2}\},\\
\{\mathrm{S3 + En \rightleftarrows NC[En, S3], kf3, kr3}\},\\
\{\mathrm{S1 + NC[En, S2] \rightleftarrows NC[En, S1, S2], kf1, kr1}\},\\
\{\mathrm{S1 + NC[En, S3] \rightleftarrows NC[En, S1, S3], kf1, kr1}\},\\
\{\mathrm{S2 + NC[En, S1] \rightleftarrows NC[En, S1, S2], kf2, kr2}\},\\
\{\mathrm{S2 + NC[En, S3] \rightleftarrows NC[En, S2, S3], kf2, kr2}\},\\
\{\mathrm{S3 + NC[En, S1] \rightleftarrows NC[En, S1, S3], kf3, kr3}\},\\
\{\mathrm{S3 + NC[En, S2] \rightleftarrows NC[En, S2, S3], kf3, kr3}\},\\
\{\mathrm{S1 + NC[En, S2, S3] \rightleftarrows NC[En, S1, S2, S3], kf1, kr1}\},\\
\{\mathrm{S2 + NC[En, S1, S3] \rightleftarrows NC[En, S1, S2, S3], kf2, kr2}\},\\
\{\mathrm{S3 + NC[En, S1, S2] \rightleftarrows NC[En, S1, S2, S3], kf3, kr3}\}\}
\end{gathered}$$

This example is of a TerTer “Random Addition” reaction case added to kMech [Najdi 2010] that generated a support request for Sigmoid. (As we are looking at only the addition phase for this compound reaction, the reactions that generate product have been omitted.)

The "random addition" strategy can be implemented for any number of substrates. Of course the requisite number of intermediate complexes and reaction steps quickly grows with the number of substrates. The number of reactions generated by a set of substrates follows the sum of the coefficients of the binomial expansion.

## 3.2 *Conversion Strategy*

The next motif we'll define is how an enzyme converts substrates into product form. The predominant form we've witnessed in the models we've built is the classic $k_{cat}$ type conversion where conversion of the complexed substrate and subsequent release of product are modeled as a one rate step. This has expanded to a bidirectional step and then a two-step bidirectional process. The general form and strategy we're adopting for SigMech separates the release process of products from the conversion phase of complexed substrate to product and will be termed the "conversion" phase or strategy. Two forms of conversion have been implemented.

### 3.2.1 *Instant Conversion*

The first form of conversion is an instant conversion where enzymatically-complexed substrates are the entry point to the release phase; this is analogous to the $k_{cat}$ example or the TwoStage catalytic reaction in the schema.

The kMech notation would appear as:

$$\mathrm{Enz}[\{\mathrm{S1}\} \overset{\mathrm{En}}{\rightleftarrows} \{\mathrm{P1}\}, \mathrm{UniUniIC}[\mathrm{kf\_}, \mathrm{kr\_}, \mathrm{kf1\_}, \mathrm{kr1\_}]] :=$$

$$\{\{\mathrm{S1} + \mathrm{En} \rightleftarrows \mathrm{NC}[\mathrm{En}, \mathrm{S1}], \mathrm{kf}, \mathrm{kr}\},$$

$$\{\mathrm{NC}[\mathrm{En}, S1] \rightleftarrows \mathrm{En} + \mathrm{P1}, kf1, kr1\}\};$$

The last reaction equation in the example above converts the enzyme/substrate complex to free enzyme and product in one step.

The conversion phase can be modeled independently of number of substrates and products independently, translating to:

$$\text{Enz}[\{\text{S1\_}, \text{S2\_}.., \text{Sn\_}\} \overset{\text{En}}{\rightleftarrows} \{\text{P1\_}, \text{P2\_}, .., \text{Pn\_}\}, \text{MultiMultiIC}[\text{kf\_}, \text{kr\_}, \text{kf1\_}, \text{kr1\_}]] \coloneqq$$

$$\{\{\text{S1} + \text{S2}+..+Sn + \text{En} \rightleftarrows \text{NC}[\text{En}, \text{S1}, \text{S2}, ..\text{Sn}], \text{kf}, \text{kr}\},$$

$$\{\text{NC}[\text{En}, \text{S1}, \text{S2}, ..\text{Sn}] \rightleftarrows \text{En} + \text{P1} + \text{P2}+..+\text{Pn}, kf1, kr1\}\};$$

### 3.2.2 ***OneStep Conversion***

The next form, adds a reaction step that converts complexed substrate into enzymatically bound product. This form is analogous to the ThreeStageCatalytic reaction in the schema. If written in kMech notation, the ThreeStageCatalytic reaction would appear as:

$$\text{Enz}[\{\text{S1}\quad\} \overset{\text{En}}{\rightleftarrows} \{\text{P1}\quad\}, \text{UniUniOSC}[\text{kf\_}, \text{kr\_}, \text{kf1\_}, \text{kr1\_}, \text{kf2}, \text{kr2}]] \coloneqq$$

$$\{\{\text{S1} + \text{En} \rightleftarrows \text{NC}[\text{En}, \text{S1}], \text{kf}, \text{kr}\},$$

$$\{\{\text{NC}[\text{En}, \text{S1}] \rightleftarrows \text{NC}[\text{En}, \text{P1}], \text{kf2}, \text{kr2}\},$$

$$\{\text{NC}[\text{En}, P1] \rightleftarrows \text{En} + \text{P1}, kf1, kr1\}\};$$

OneStep Conversions can be modeled independently of the number of substrates and products independently translating to:

$$\text{Enz}[\{\text{S1\_}, \text{S2\_}.., \text{Sn\_}\} \overset{\text{En}}{\rightleftarrows} \{\text{P1\_}, \text{P2\_}, .., \text{Pn\_}\}, \text{MultiMultiOSC}[\text{kf\_}, \text{kr\_}, \text{kf1\_}, \text{kr1\_}]] \coloneqq$$

$$\{\{\text{S1} + \text{S2}+..+Sn + \text{En} \rightleftarrows \text{NC}[\text{En}, \text{S1}, \text{S2}, ..\text{Sn}], \text{kf}, \text{kr}\},$$

$$\{\text{NC}[\text{En}, \text{S1}, \text{S2}, ..\text{Sn}] \rightleftarrows \text{NC}[\text{En}, \text{P1}, \text{P2}, ..\text{Pn}], \text{kf}, \text{kr}\},$$

$$\{\text{NC}[\text{En}, \text{P1}, \text{P2}, ..\text{Pn}] \rightleftarrows \text{En} + \text{P1} + \text{P2} + \cdots + \text{Pn}, kf1, kr1\}\};$$

## 3.3 *Release Strategy*

Just as there is a strategy for addition of substrates into complex, there can be a strategy for release of products from intermediate complex. The same algorithms that we use for addition of substrates can be applied to the release of products.

### 3.3.1 ***Full Release***

"Full release" names the case where all products are released in one reaction step.

The [Yang 2005] BiBi reaction is an example of a full release reaction where all products are released in one step.

$$Enz[\{\mathrm{S1\_, S2\_}\} \overset{\mathrm{En\_}}{\rightleftarrows} \{\mathrm{P1\_, P2\_}\}, \mathrm{BiBi}[\mathrm{kf\_, kr\_, k\_}]] := \{\{\mathrm{S1 + S2 + En} \rightleftarrows \mathrm{NC}[\mathrm{En, S1, S2}], \mathrm{kf, kr}\}, \{\mathrm{NC}[\mathrm{En, S1, S2}] \to \mathrm{En + P1 + P2}, k\}\};$$

A SigMech implementation general form with bidirectional reactions for the release phase would appear as:

$$\{\mathrm{NC}[\mathrm{En, S1, S2, ., ., Sn}] \rightleftarrows \mathrm{P1 + P2 +} \ldots + Pn + \mathrm{En, kcf, kcr}\}.$$

The previous equation assumes an instant conversion. This assumption should be relieved to become independent of the conversion phase. SP indicates an unknown (either substrate or product.) Full release strategies translate to:

$$\{\mathrm{NC}[\mathrm{En, SP1, SP2, ., ., SPn}] \rightleftarrows \mathrm{P1 + P2 +} \ldots + Pn + \mathrm{En, kcf, kcr}\}.$$

### 3.3.2 ***Sequential Release***

Sequential release names the case in which products are assumed to release one after another in a predetermined order. With addition and conversion phase reactions omitted, sequential release may be expanded into more elementary reactions as follows:

$$\mathrm{Enz}[\{\mathrm{S1\_, S2\_, ., Sn}\} \overset{\mathrm{En}}{\rightleftarrows} \{\mathrm{P1\_, P2\_, ., Pn}\},$$

$$\text{OrderedRelease}[\text{kf1\_}, \text{kr1\_}, \text{kf2\_}, \text{kr2\_}, \text{kcf1\_}.., \text{kcr1\_}.., \text{kcfn\_}, \text{kcrn\_}]] :=$$

$$\{\{\text{NC}[\text{En}, \text{SP1}, \text{SP2}, ., ., \text{SPn}] \rightleftarrows \text{P1} + \text{NC}[\text{En}, \text{SP2}, ., ., \text{SPn}], \text{kcf1}, \text{kcr1}\}$$

$$\{\text{NC}[\text{En}, \text{SP2}, ., ., \text{SPn}] \rightleftarrows \text{P2} + \text{NC}[\text{En}, \text{SP}.., \text{SPn}], \text{kcf2}, \text{kcfr2}\}$$

$$\{\text{NC}[\text{En}, \text{SP}.., \text{SPn}] \rightleftarrows \text{P}.. + \text{NC}[\text{En}, \text{SPn}], \text{kcf}.., \text{kcr}..\}$$

$$\{\text{NC}[\text{En}, \text{SPn}] \rightleftarrows \text{Pn} + \text{En},, \text{kcfn}, \text{kcrn}\}.$$

### 3.3.3 *Random Release*

Random release refers to the case in which products may disassociate from complex in any order. The algorithm for generating the reactions is the same as it is for random addition, except that the reaction equations are reversed and products are generated from the intermediate complex.

The strategies for addition, conversion and release can be independent of one another. This is one characteristic that leads to representing many potential enzymatic process mechanisms with a concise parameterized input.

## *3.4 Ping Pong Pattern*

Another case for enzymatic process exists: the ping pong mechanism. Enzymes that operate with ping pong mechanisms have chemically modified intermediate states. A particular substrate when converted to product leaves the enzyme chemically modified. This modified intermediate enzyme state is then able, in turn, to bind to a different substrate.

A standard BiBi ping pong reaction is illustrated in Figure 3.3. Here a substrate is bound by free enzyme. This substrate is converted to product and subsequently dissociates from the enzyme leaving the enzyme in a different state. This alternate state is required to bind a second substrate that in turn is catalyzed into product and free enzyme.

The ping pong pattern can be applied to any number of substrates so long as there is an intermediate for each additional substrate.

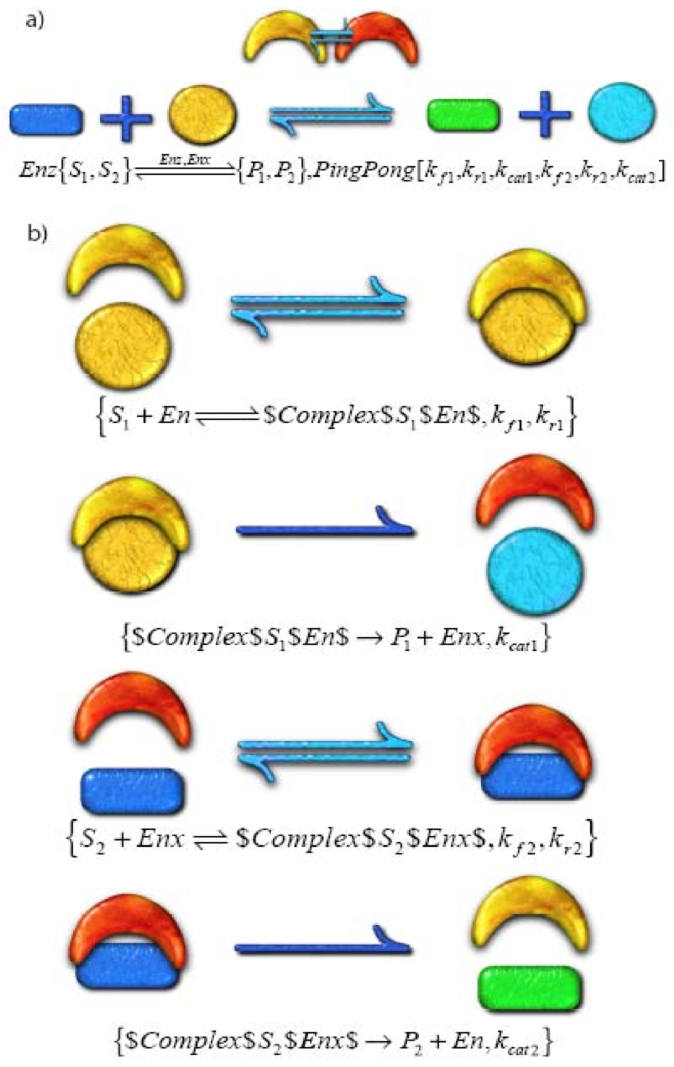

a)
Enz{S1,S2} ⇌ Enz,Enx {P1,P2},PingPong[kf1,kr1,kcat1,kf2,kr2,kcat2]
b)
{S1 + En ⇌ $Complex$S1$En$,kf1,kr1}
{$Complex$S1$En$ → P1 + Enx,kcat1}
{S2 + Enx ⇌ $Complex$S2$Enx$,kf2,kr2}
{$Complex$S2$Enx$ → P2 + En,kcat2}

**Figure 3.3 A BiBi Ping Pong kMech Reaction. a: Cellerator input notation is shown below the reaction cartoon. The enzyme has two states, free enzyme and chemically modified. The kMech compound reaction shown in a) is translated to four sub-reactions shown in b: 1) The enzyme binds to substrate S1. 2) Substrate is converted to product P1 and the enzyme is modified to a state where it can bind S2. 3) The modified enzyme binds S2. 4) S2 is converted to product and the enzyme is returned to its original state.**

## *3.5 Inhibition Models*

There are currently three inhibition models supported by kMech [Yang 2005]. The equations below are xCellerator definitions derivative of the kMech implementation with a slight difference. The difference is that the conversion of substrate-enzyme complex to product in kMech appears as a unidirectional catalytic reaction. SigMech, in order to fully generalize, currently uses bidirectional reactions for all mechanism sub-reactions. The unidirectional case can be recovered by setting the reverse reaction rate to zero. The implementation of SigMech allows for multiple inhibitors to be assigned to a particular enzymatic reaction within the schema, and in the GUI. The previous schema representation suffered logistically from having to represent differing permutations of reaction inhibitions. The schema representation shown in Figure 3.2 was built to store these forms of inhibition. The SigMech implementation, using the Sigmoid Java API, reconstructs these patterns of inhibition with bidirectional mass action reactions.

### 3.5.1 *Competitive inhibition:*

The case of competitive inhibition exists where an enzyme is bound by an inhibitor, forming an enzyme inhibitor complex, thus preventing substrate from biding to the enzyme. This process consists of one additional reaction added to an enzyme mechanism. This particular case is a SigMech adaptation of the kMech version of the reaction equations. The difference is that bidirectional rates are used for the conversion of substrate-enzyme complex to product and free enzyme. The reaction equations in kMech notation are as follows:

$$\mathrm{Enz}[\{\mathrm{S}\} \overset{\mathrm{En}}{\rightleftarrows} \{\mathrm{P}\}, \mathrm{UniUni}[\mathrm{kf\_}, \mathrm{kr\_}, \mathrm{kc1\_}, \mathrm{kcr1}], SM\mathrm{CI}[\mathrm{inh\_}, \mathrm{kfi\_}, \mathrm{kri\_}]] :=$$

$$\{\{S + \mathrm{En} \rightleftarrows \mathrm{NC}[S, \mathrm{En}], \mathrm{kf}, \mathrm{kr}\},$$

$$\{\mathrm{NC}[S, \mathrm{En}] \rightleftarrows \mathrm{En} + P, kc1, kcr2\},$$

$$\{\mathrm{En} + \mathrm{Inh} \rightleftarrows \mathrm{NC}[\mathrm{En}, \mathrm{Inh}], \mathrm{kfi}, \mathrm{kri}\}\};$$

### 3.5.2 ***Uncompetitive inhibition:***

This case exists where the substrate-enzyme complex is targeted and bound by an inhibitor thus preventing conversion or release of product.

$$\mathrm{Enz}[\{\mathrm{S}\} \overset{\mathrm{En}}{\rightleftarrows} \{\mathrm{P}\}, \mathrm{UniUni}[\mathrm{kf\_}, \mathrm{kr\_}, \mathrm{kc1\_}, \mathrm{kcr1}], SM\mathrm{UCI}[\mathrm{inh\_}, \mathrm{kfi\_}, \mathrm{kri\_}]] :=$$

$$\{\{S + \mathrm{En} \rightleftarrows \mathrm{NC}[S, \mathrm{En}], \mathrm{kf}, \mathrm{kr}\},$$

$$\{\mathrm{NC}[S, \mathrm{En}] \rightleftarrows \mathrm{En} + P, kc1, kcr2\},$$

$$\{\mathrm{NC}[S, \mathrm{En}] + \mathrm{Inh} \rightleftarrows \mathrm{NC}[S, \mathrm{En}, \mathrm{Inh}], \mathrm{kfi}, \mathrm{kri}\}\};$$

### 3.5.3 ***Noncompetitive inhibition:***

An assumption is made that the inhibitor will not interfere with substrate binding of the enzyme. This means the substrate can still be bound by the Enzyme-Inhibitor complex. This generates two possible reaction paths. Either the free enzyme can be bound by the inhibitor and then binds the substrate forming the substrate-enzyme-inhibitor complex, or the substrate-enzyme complex can be bound by the inhibitor forming a substrate-enzyme-inhibitor complex. The following is an example, in kMech notation, of noncompetitive inhibition:

$$\mathrm{Enz}[\{\mathrm{S}\} \overset{\mathrm{En}}{\rightleftarrows} \{\mathrm{P}\}, \mathrm{UniUni}[\mathrm{kf\_}, \mathrm{kr\_}, \mathrm{kc1\_}, \mathrm{kcr1}], \mathrm{NCI}[\mathrm{inh\_}, \mathrm{kfi\_}, \mathrm{kri\_}]] :=$$

$$\{\{S + \mathrm{En} \rightleftarrows \mathrm{NC}[S, \mathrm{En}], \mathrm{kf}, \mathrm{kr}\},$$

$$\{\mathrm{NC}[S, \mathrm{En}] \rightleftarrows \mathrm{En} + P, kc1, kr1\},$$

$$\{\mathrm{En} + \mathrm{Inh} \rightleftarrows \mathrm{NC}[\mathrm{En}, \mathrm{Inh}], \mathrm{kfi}, \mathrm{kri}\},$$

$$\{\mathrm{NC}[S, \mathrm{En}] + \mathrm{Inh} \rightleftarrows \mathrm{NC}[S, \mathrm{En}, \mathrm{Inh}], \mathrm{kfi}, \mathrm{kri}\},$$

$$\{S + \mathrm{NC}[\mathrm{En}, \mathrm{Inh}] \rightleftarrows \mathrm{NC}[S, \mathrm{En}, \mathrm{Inh}], \mathrm{kf}, \mathrm{kr}\}\};$$

Noncompetitive inhibition possesses a residual reaction rate for enzyme-inhibitor complexes. In some instances, bound enzyme complexes can still have some residual activity to convert substrates to product. The kMech reaction notation for noncompetitive inhibition (as implemented in SigMech) with a residual rate is as follows:

$$ENZ[\{\mathrm{S\_}\} \overset{\mathrm{En\_}}{\rightleftarrows} \{\mathrm{P\_}\}, \mathrm{UniUni}[\mathrm{kf\_}, \mathrm{kr\_}, \mathrm{kc1\_}, \mathrm{kcr1\_}], SM\mathrm{NCI}[\mathrm{inh\_}, \mathrm{kfi\_}, \mathrm{kri\_}, \mathrm{residualRate\_}, \mathrm{rRr}]]:$$
$$= \{\{S + \mathrm{En} \rightleftarrows \mathrm{NC}[S, \mathrm{En}], \mathrm{kf}, \mathrm{kr}\}, \{\mathrm{NC}[S, \mathrm{En}] \rightleftarrows \mathrm{En} + P, k, kr\}, \{\mathrm{En} + \mathrm{Inh}$$
$$\rightleftarrows \mathrm{NC}[\mathrm{En}, \mathrm{Inh}], \mathrm{kfi}, \mathrm{kri}\}, \{\mathrm{NC}[S, \mathrm{En}] + \mathrm{Inh} \rightleftarrows \mathrm{NC}[S, \mathrm{En}, \mathrm{Inh}], \mathrm{kfi}, \mathrm{kri}\}, \{S + \mathrm{NC}[\mathrm{En}, \mathrm{Inh}]$$
$$\rightleftarrows \mathrm{NC}[S, \mathrm{En}, \mathrm{Inh}], \mathrm{kf}, \mathrm{kr}\}, \{\mathrm{NC}[S, \mathrm{En}, \mathrm{Inh}] \rightleftarrows \mathrm{NC}[\mathrm{En}, \mathrm{Inh}] + P, \mathrm{residualRate} * \mathrm{kc1}, \mathrm{rRr}\}\};$$

The enzyme Inhibitor complex may still bind its' substrate and release a product in two respective reaction steps. The respective residual rate is a fraction of the base enzymatic rate for this reaction.

## 3.6 Parameter Summary

In summary, the entire set of kMech instances (together with a few xCellerator reactions) can be summarized by a set of specified input parameters.

These parameters are:

A vector of Substrates [] : [Uni, Bi , Ter, Tet, ...]

A vector of Products []: [Uni, Bi , Ter, Tet, ...]

AdditionStrategy: [Full, Ordered, Random]

ConversionStrategy: [Instant, One Step]

ReleaseStrategy: [Full, Ordered, Random]

PingPongCase: [Enzyme Intermediates] (This condition reduces the addition and release strategies to a specific ordered case).

Inhibition Model: [Competitive, NonCompetitive, Uncompetitive]

These options sweep out a large cross-product space of reaction types.

A biologist might describe a reaction as follows:

"We have an enzymatic BiBi reaction with random addition, ordered release and a residual enzyme activity of the S1E complex.  Also, we have competitive inhibition for I1 and non competitive inhibition for I2".  This can be notated as parameterized reaction, as follows [Bi, Bi, randomAddition[rates], orderedRelease,[rates][CI, Inhibitor, [rates]][NCI, Inhibitor, [rates]]]  From this description the subreactions can be algorithmically generated from parameterized reaction notation like the following:

[[subs[int], prods[int], addStrat[String], convertStrat[String], releaseStrat[String],[Inhibitions[Inhibitor, target, [kf, kr]]].

This representation includes the conversion phase.  The representation provides a data framework and with a set of procedures we can generate a biological reaction network. Each BiologicalReaction in the network will have a corresponding MathematicalReaction.   In version 1.0 of SigMech, Mass action BidirectionalMathematical reactions are constructed to provide kinetics for simulation. Ultimately, the mapping between the BiologicalReaction network and MathematicalReactions should be flexible and allow for any relevant

MathematicalReaction (or set of reactions) to be associated with a particular biological sub-reaction of the mechanism. The representation below is close to conforming to regular expression notation. It should serve as an input notation that is to be parsed to generate the mechanism.

```
[
        [
                subs[ [substrate, kf, kr]*], prods[ [product, kf, kr]*]
        ],
        [
                PingPong[ [enzymeIntermediates]* |
                [
                AddStrat [ full |ordered | random ], ReleaseStrat[ full | ordered | random ]
                ],
        [
                ConversionStrat[ instant | oneStep | multistep, [kf, kr] ]
         ],
        [Inhibitions[inhibitor, target, [ competitive | nonCompetitive
[ResidualRate[complex, substrate,     product, [residualRate]]*, kf, kr] | uncompetitive ],
[kf, kr]]*
]
```

Capitalization indicates a method to be run. Uncapitalized entities are variables . In all cases, Capitalized words are procedures that generate reactions. An asterisk (*) indicates a zero-to-many relationship. The pipe symbol ( | ) specifies an OR input condition. The variables subs, prods and enzymeIntermediates should be stored in the form of ordered lists.

## *3.7 User Interface:*

The SigMech user interface has been written in Java and may be run as a standalone application or accessed from within SME by a button on the far top left of the SME control panel as seen in Figure 3.4. After a suitable enzyme mechanism has been designed, the corresponding reaction network may be viewed in the SME network view panel by pressing the "View Enzyme Mechanism" button just to the right, also shown in Figure 3.4.

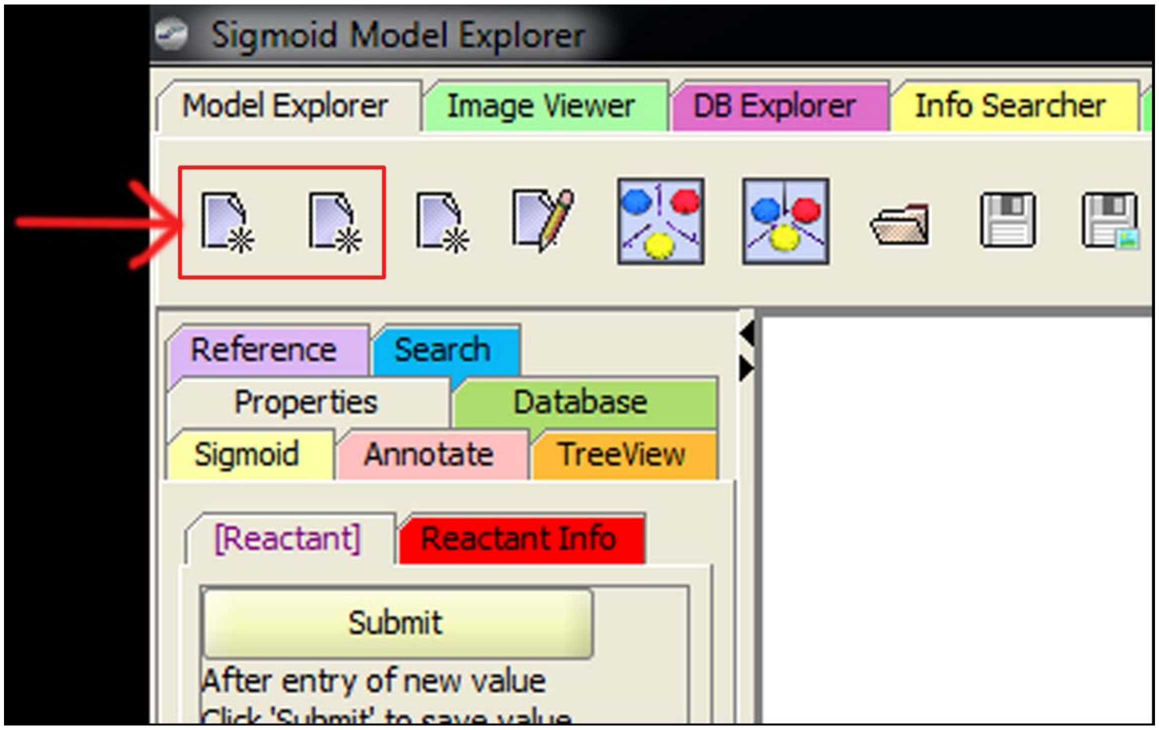


**Figure 3.4 The access buttons for SigMech from SME appear on the left of the taskbar. The button on the far left opens the SigMech utility. The button to the right displays the enzyme mechanism once it has been generated.**

### 3.7.1 *Accessing SigMech from SME*

Currently the user interface for SigMech, shown in Figure 3.5, consists of a primary single user window designed to take in all the respective parameters necessary to produce an enzyme mechanism. There are four addition panels, four drop down menus, and access buttons for mechanism and enzyme attributes. The panels from left to right are: the substrate list, the product list, the inhibitor list and the enzyme intermediate list. Under each list are addition and removal buttons. A new reactant can be instantiated by pressing the respective addition button. Version 1.0 queries for just a reactant name, but a next version could provide access to more detailed reactant specifications. The inhibitor addition process may query for inhibitor target information depending on the inhibition type.  The Enzyme Intermediates list is tied to the use of PingPong reaction types.   The console output should give an error message either when intermediates are present and PingPong is turned off, or when ping pong is on but there are no intermediates specified. Version 1.0 does not enforce these rules on generation of the mechanism.   The "Mechanism" button is used to change the mechanism name.  The "Enzyme" button does the same for the enzyme name. (The plan is for subsequent versions to have more detailed access to the enzyme's Sigmoid reactant attributes.) There is a button labeled "Report" at the bottom designed to produce console feedback on the current mechanism configuration and a button labeled "Generate" (bottom right) that instantiates all respective Sigmoid reactants and reactions for the attributes currently specified in the SigMech interface.  If the console is active, Generate also calls the model emitter class to report output of the entire mechanism model.  Typically once a mechanism has been designed, it can be viewed in the console as "Model emitter" output or the "View Enzyme Mechanism" button in the main interface can be pressed in SME to view the reactions in the SME network view panel.

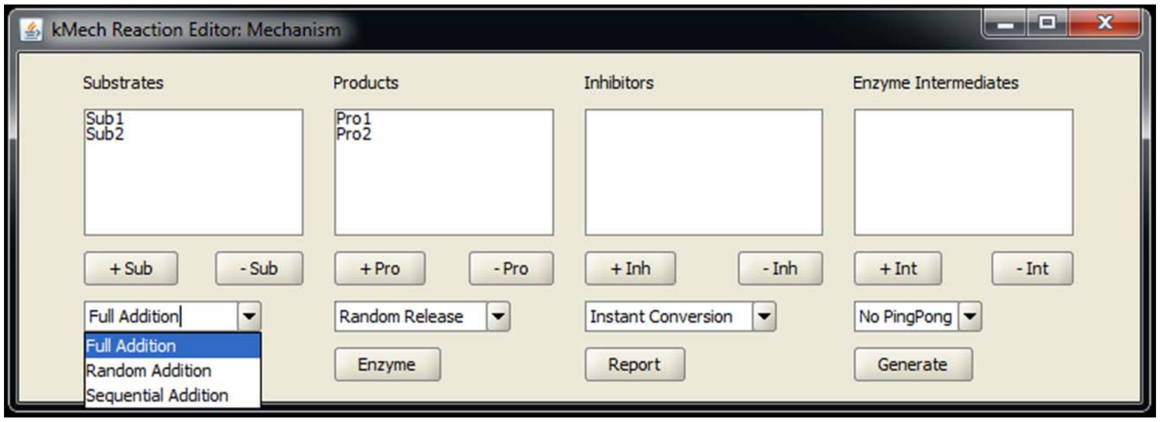


**Figure 3.5 The addition strategy selection menu appears at the far left bottom of the UI below the list of participating substrates. Three addition strategies are available, Full Addition, Random Addition or Sequential Addition.**

### 3.7.2 *Drop down Menus*

There are drop down menus available to select the strategies for building the enzyme mechanism network. Figure 3.5 illustrates the addition menu when activated. This menu is the drop down menu furthest to the left under the substrate addition panel. Substrates can be added by pressing the" +Sub" button. Currently in version 1.0 of SigMech will query the user for a substrate name and instantiates a Sigmoid reactant class with that name. The long term intention here is to open a larger panel with complete access to all the reactant attributes and subclasses. Figure 3.6 illustrates the release menu. As with the addition menu, three independent alternatives are available for generating the reactions via "Full", "Sequential", and "Random" strategies.

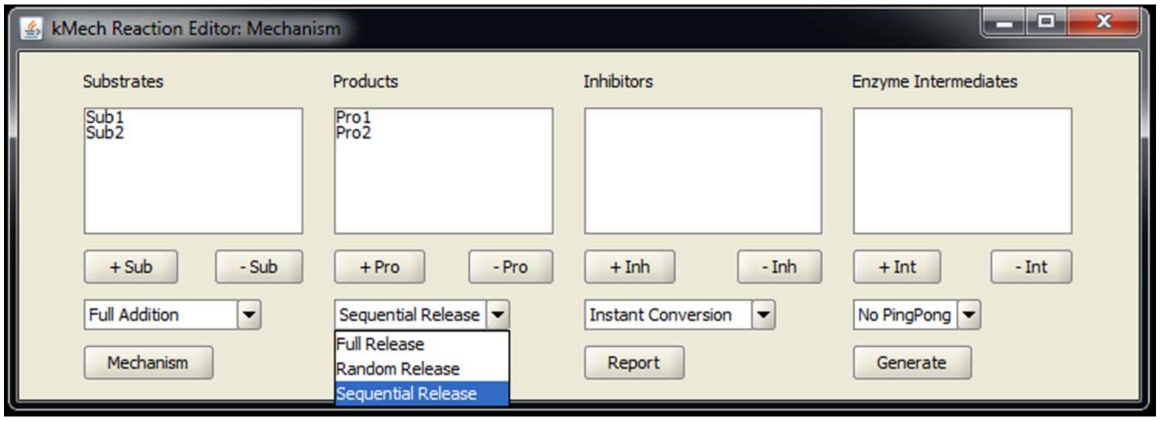
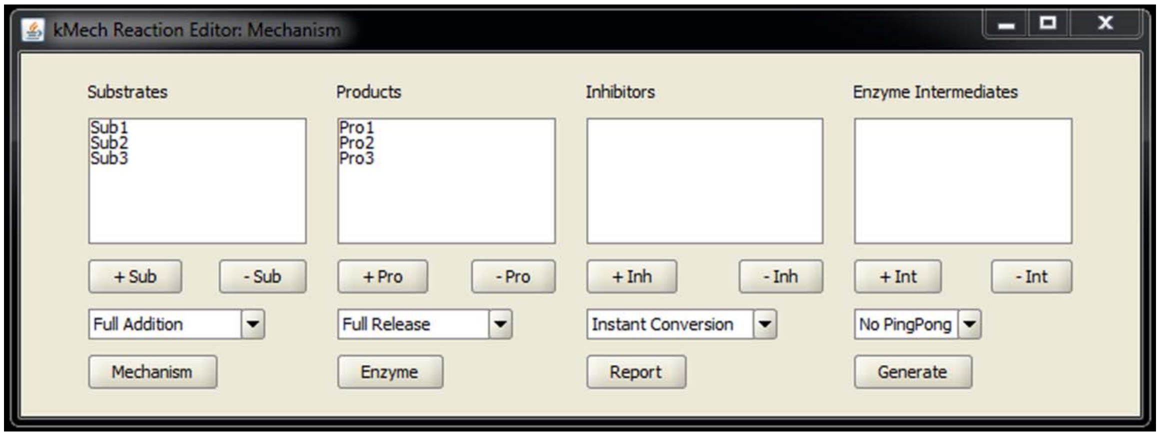


**Figure 3.6 The release strategy selection menu appears beneath the list of participating products. As with the addition strategy, three choices are available: Full Release, Random Release, and Sequential Release.**

### 3.7.3 *TerTer Full Addition Full Release Example*

For our first example we construct a mechanism with both a full addition of substrates and full release of products. The SigMech interface is shown in Figure 3.7 and the SME network view is shown in Figure 3.8. Substrates are shown in red. Products are shown in blue. In this example with "Full addition", the binding of all substrates to the free enzyme is modeled as one bidirectional mass action reaction. SME only illustrates the Biological reactions in the network layout view. The corresponding mass-action MathematicalReactions are accessible on the panel to the left.(Not shown)


kMech Reaction Editor: Mechanism

Substrates: Sub1, Sub2, Sub3
Products: Pro1, Pro2, Pro3
Inhibitors:
Enzyme Intermediates:

Full Addition | Full Release | Instant Conversion | No PingPong

**Figure 3.7 Interface view for a Ter-Ter reaction with Full Addition and Full Release.**

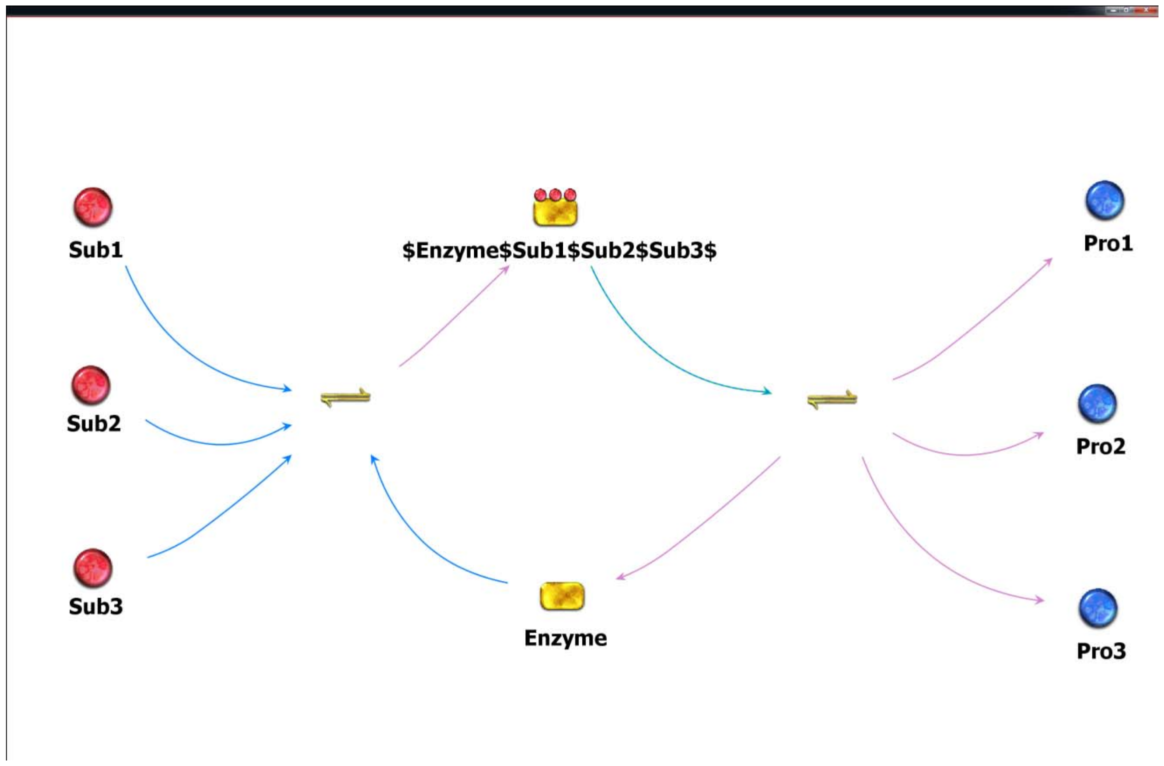
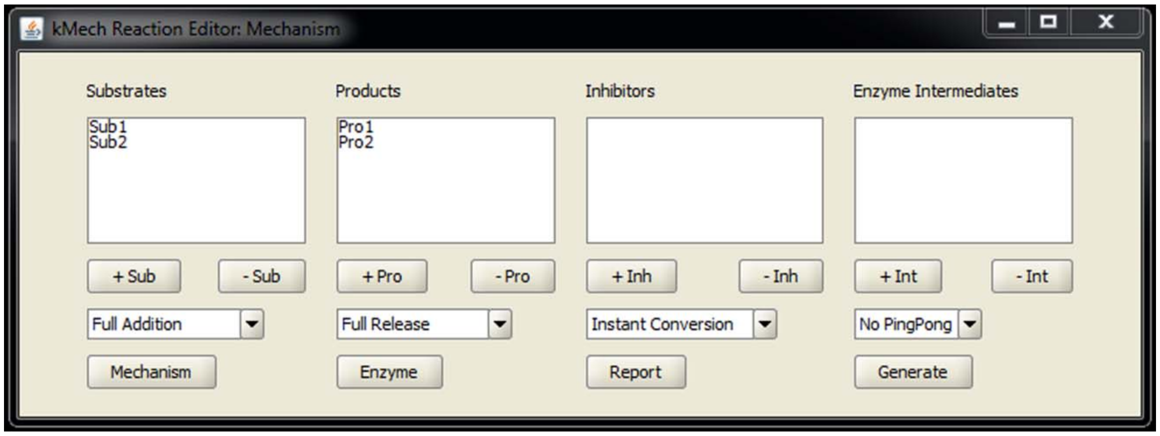


**Figure 3.8 SME Network view of the Ter-Ter Full Addition Full Release Mechanism. Substrates are on the left and products on the right.**

This mechanism also has a "Full Release" of products, so the conversion of substrates to product form, and subsequent disassociation (release) from complex are modeled as one step. It should be noted that all reactions are modeled as bidirectional mass action "MathematicalReaction" Sigmoid reactions. For a $k_{cat}$ approximation of this release reaction the reverse rate should be set to zero.

Of course, the number of reactants on each side of the equation is arbitrary, as can be seen in Figure 3.9 and Figure 3.10. There just as well could be six reactants on the substrate side and one on the product side. Addition to complex and release are completely independent of one another. This simple example illustrates how the new SigMech representation can express many previously hard-coded enzyme mechanisms and procedurally generate the required subreactions.


kMech Reaction Editor: Mechanism

| Substrates | Products | Inhibitors | Enzyme Intermediates |
|---|---|---|---|
| Sub1<br>Sub2 | Pro1<br>Pro2 | | |
| Full Addition | Full Release | Instant Conversion | No PingPong |

**Figure 3.9 SigMech Interface: BiBi Full Addition Full Release.**

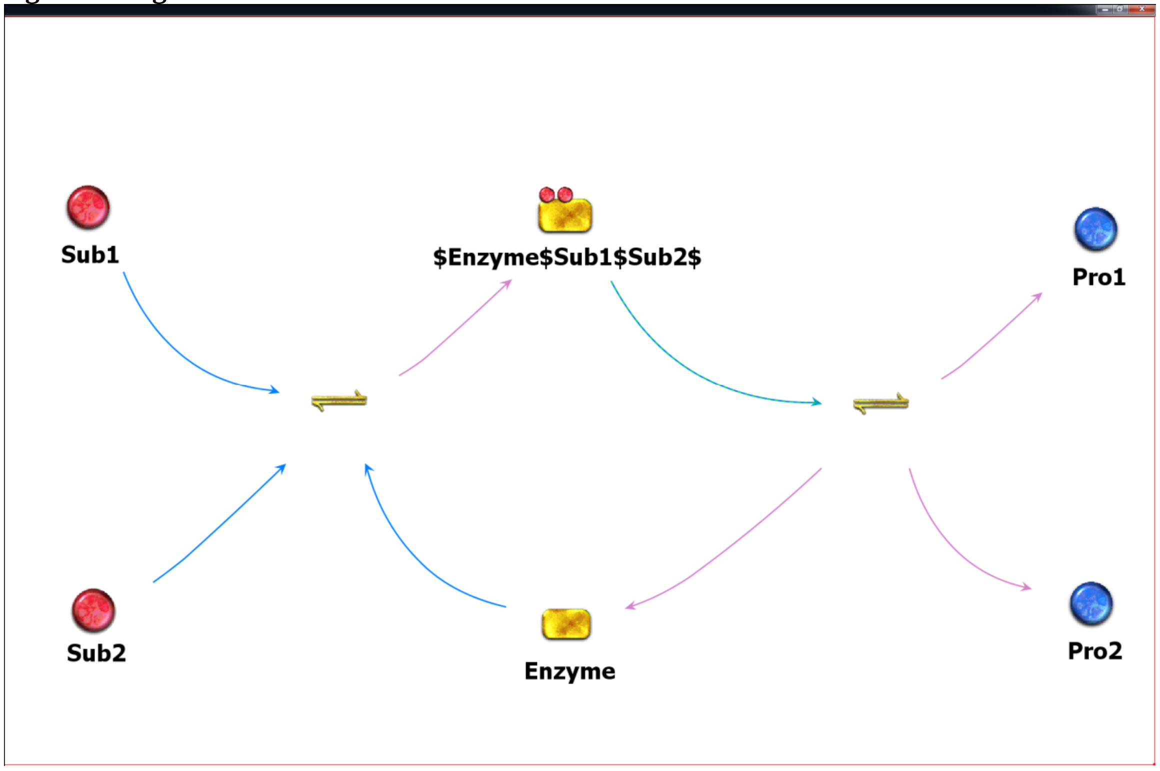
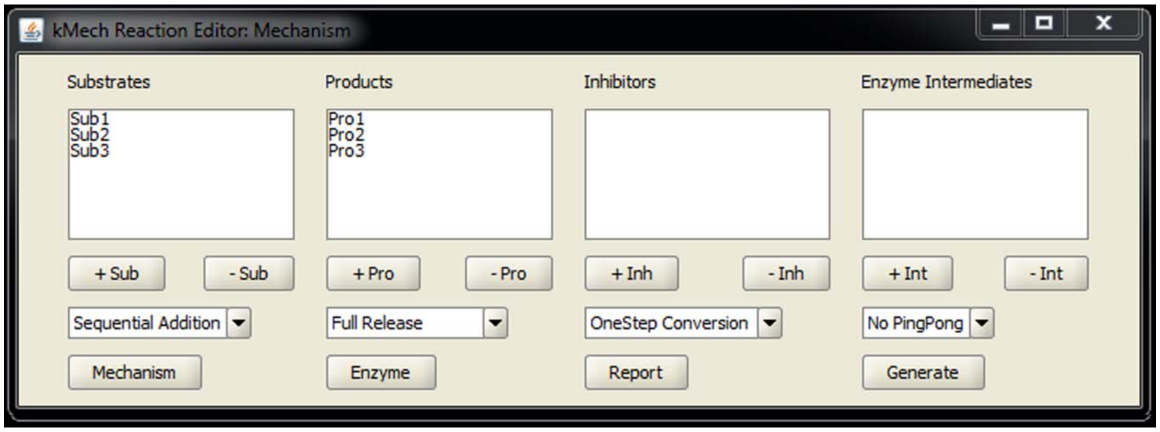


**Figure 3.10 SME view of Bi-Bi Full Addition Full Release reaction.**

### 3.7.4 *TerTer Sequential Addition Full Release Example*

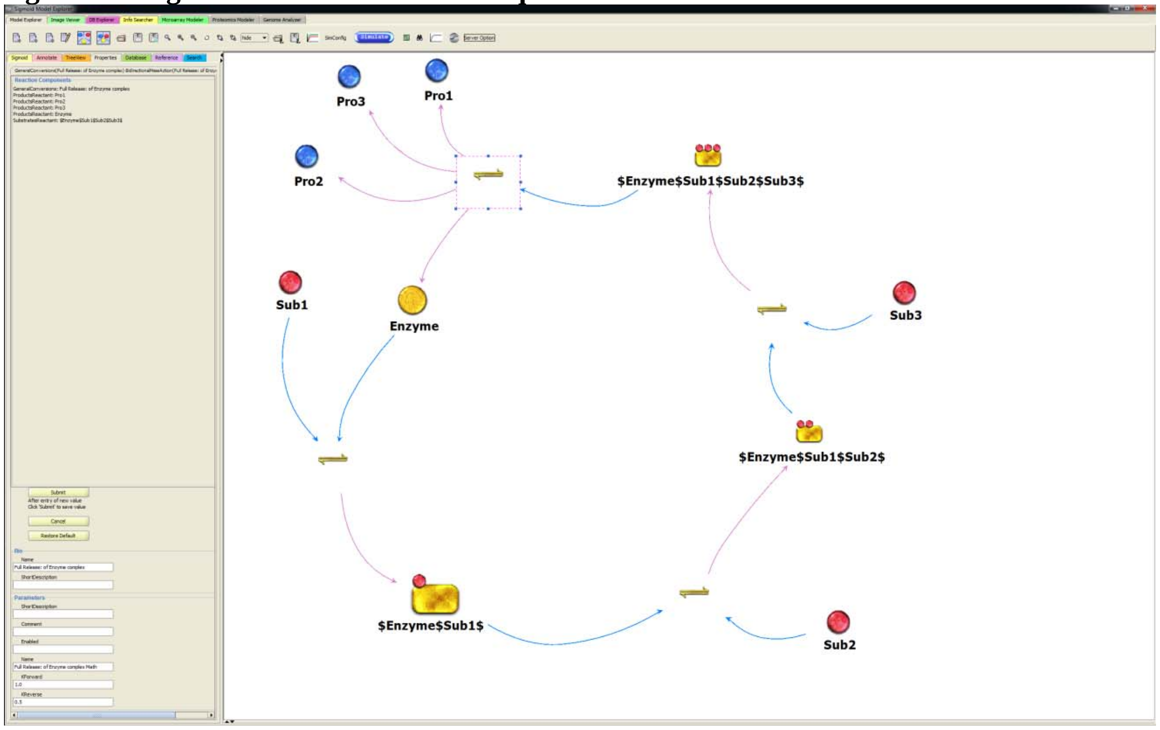
To further illustrate the independence of addition and release strategies a sequential addition and full release mechanism will serve as an example. Substrates are added to the enzyme complex in the order that they occur in the interface list shown in Figure 3.11 (From top to bottom). In this example the full SME image is shown. Substrates again are in red and products in blue. The release reaction is selected and the corresponding MathematicalReaction attributes are displayed on the left panel, including kinetic rates.



Substrates: Sub1, Sub2, Sub3
Products: Pro1, Pro2, Pro3
Inhibitors:
Enzyme Intermediates:

Sequential Addition
Full Release
OneStep Conversion
No PingPong

**Figure 3.11 SigMech Interface: TerTer Sequential Addition Full Release.**



GeneralConversions(Full Release: of Enzyme complex) BidirectionalMassAction(Full Release: of Enzy

Reaction Components
GeneralConversions: Full Release: of Enzyme complex
ProductsReactant: Pro1
ProductsReactant: Pro2
ProductsReactant: Pro3
ProductsReactant: Enzyme
SubstratesReactant: $Enzyme$Sub1$Sub2$Sub3$

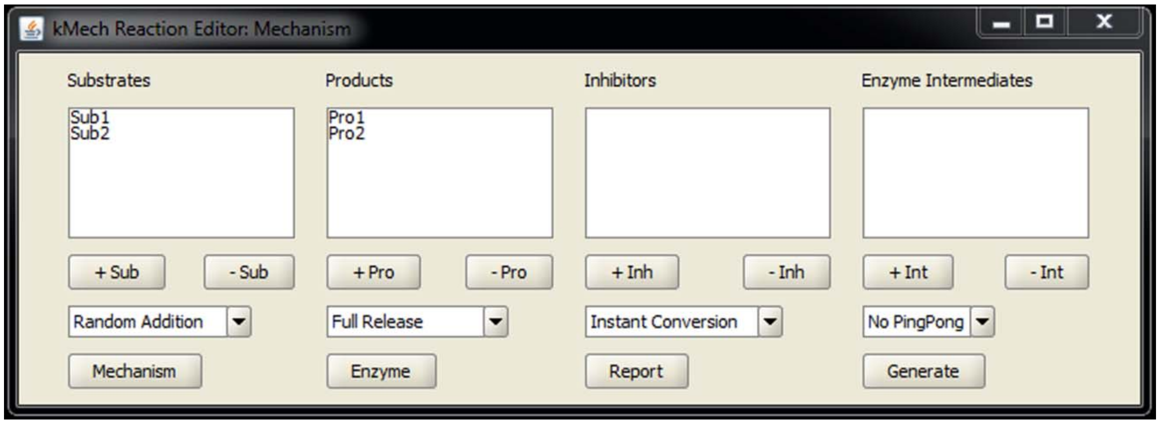
After entry of new value
Click 'Submit' to save value

Bio
Name: Full Release: of Enzyme complex
ShortDescription:

Parameters
ShortDescription:
Comment:
Enabled:
Name: Full Release: of Enzyme complex Math
KForward: 1.0
KReverse: 0.5

**Figure 3.12 SME: TerTer Sequential Addition Full Release. This illustrates that addition and release mechanisms can be independent of one another.**

Addition and release strategies can be random as well. A random selection (in the case of addition) produces reactions for every possible path to bind all available substrates. This can be used when the intention is to model a mechanism in which any substrate can bind in any order. Of course, specifying rates for such a strategy must be given particular attention because the rates act in parallel to each other and produce sums of enzymatic activity. A BiBi mechanism with random addition and full release is shown in Figure 3.14.



Substrates: Sub1, Sub2
Products: Pro1, Pro2
Random Addition
Full Release
Instant Conversion
No PingPong

**Figure 3.13 SigMech Interface: BiBi Random Addition Full release.**

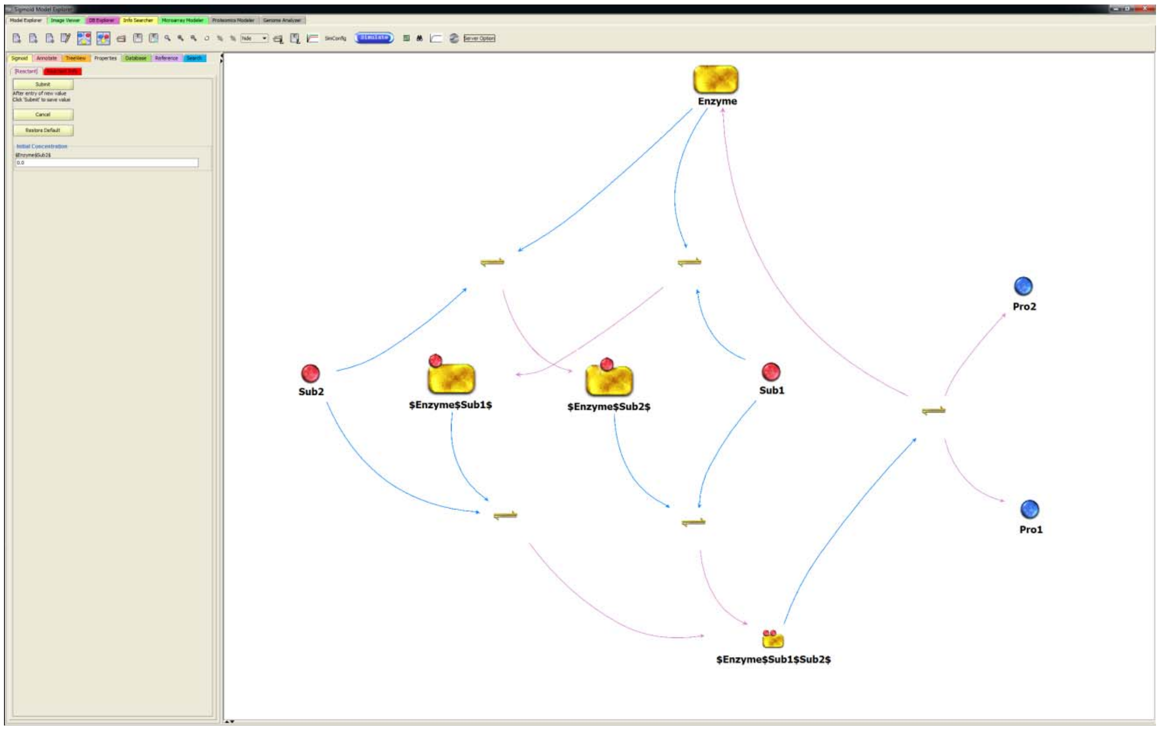
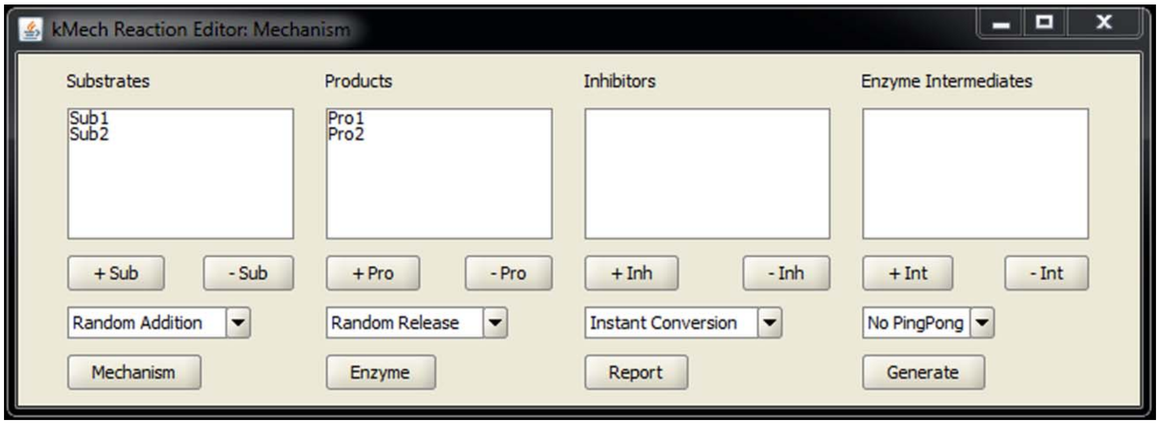


**Figure 3.14  SME: BiBi Random Addition Full release.**

The Random strategy is also available to expert users for release of product from enzyme/substrate complex.  The next example illustrates a the BiBi mechanism with a random strategy for both addition and release phases.



| Substrates | Products | Inhibitors | Enzyme Intermediates |
|---|---|---|---|
| Sub1<br>Sub2 | Pro1<br>Pro2 | | |
| Random Addition | Random Release | Instant Conversion | No PingPong |

**Figure 3.15 SigMech: BiBi Random Addition Random Release.**

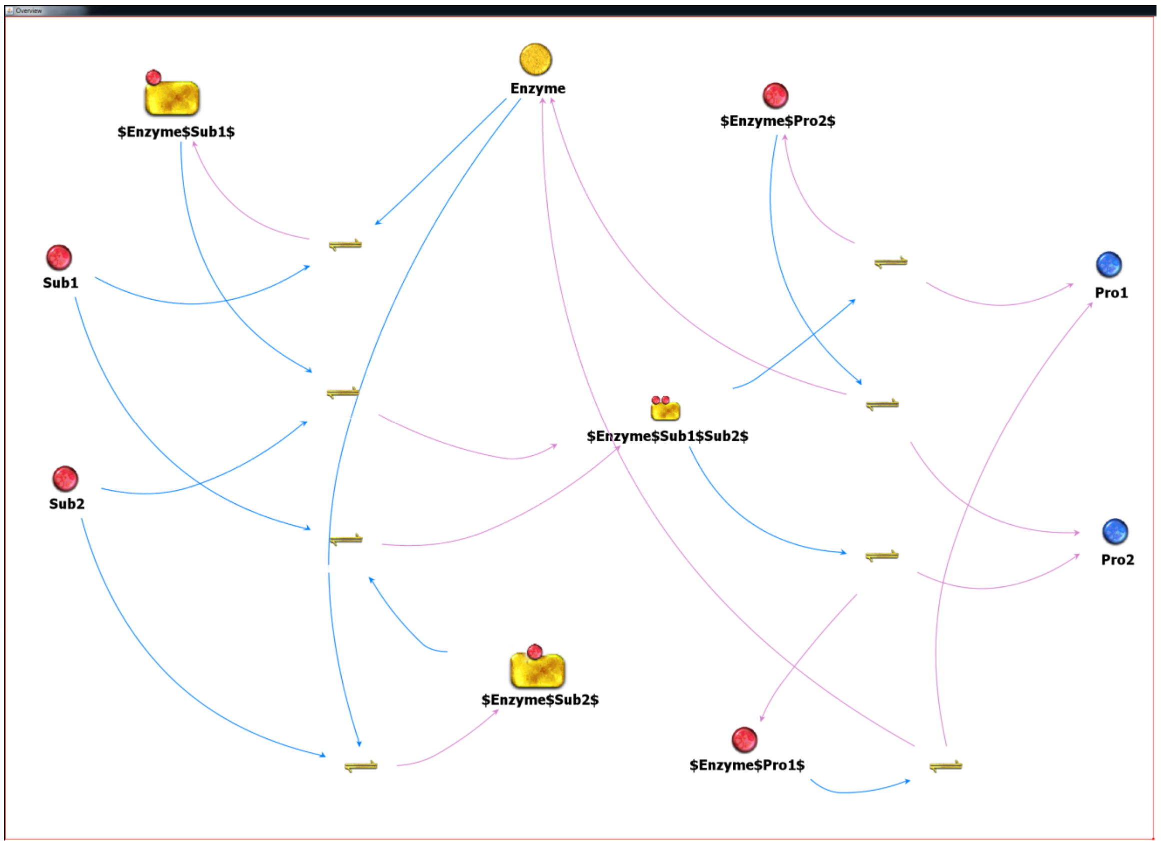


**Figure 3.16 SME Overview: BiBi Random Addition Random Release.**

The user may define any number of substrates and products with this strategy in either the addition or release phases, but the number of reactions and reactants increases rapidly with higher numbers of substrates or products. A TerTer Random Addition Random Release network is shown in Figure 3.17. Substrates are indicated by the yellow balls in the top center. Products are blue at bottom center.

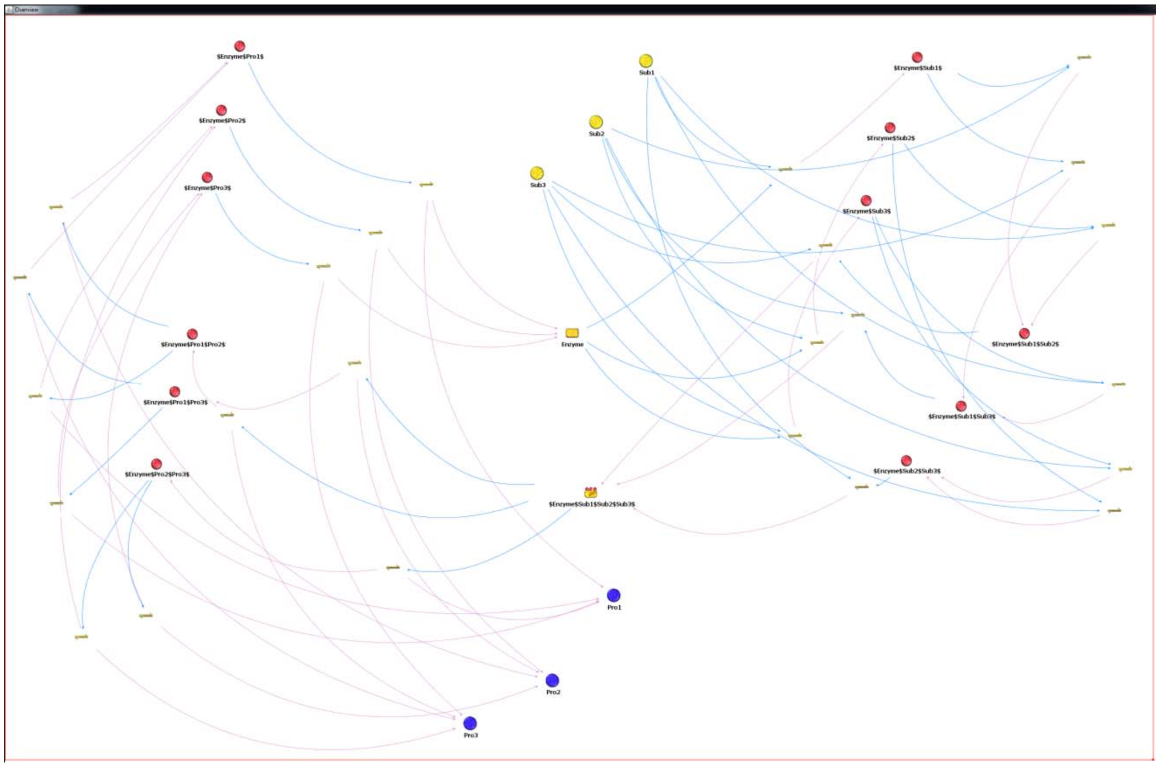
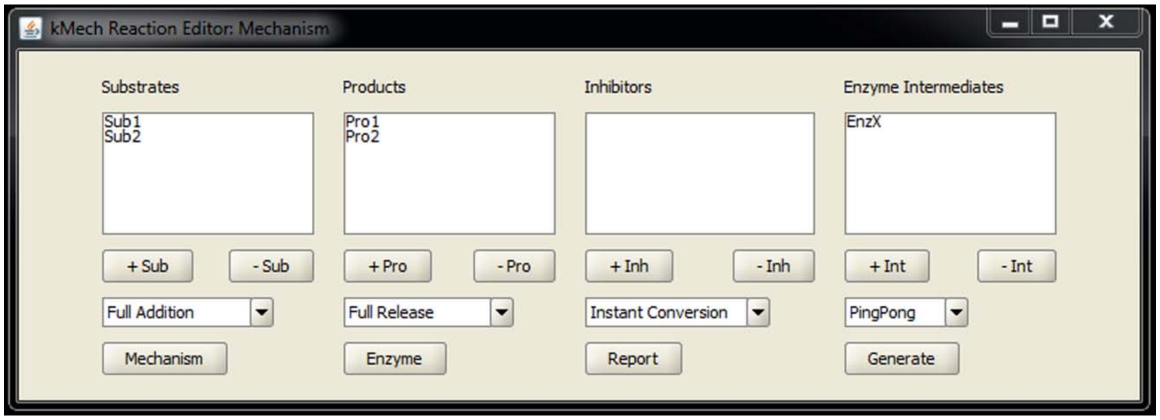


**Figure 3.17 SME Overview: TerTer Random Addition Random Release.**

## 3.7.5 **BiBi *Ping Pong Example***

Ping Pong reactions can be built with SigMech. Substrates are again shown in red and products in blue. The classic BiBi Ping Pong reaction is illustrated in Figure 3.18 and Figure 3.19.


kMech Reaction Editor: Mechanism

Substrates: Sub1, Sub2
Products: Pro1, Pro2
Inhibitors:
Enzyme Intermediates: EnzX

Full Addition
Full Release
Instant Conversion
PingPong

**Figure 3.18 SigMech Interface: Classic BiBi Ping Pong reaction.**

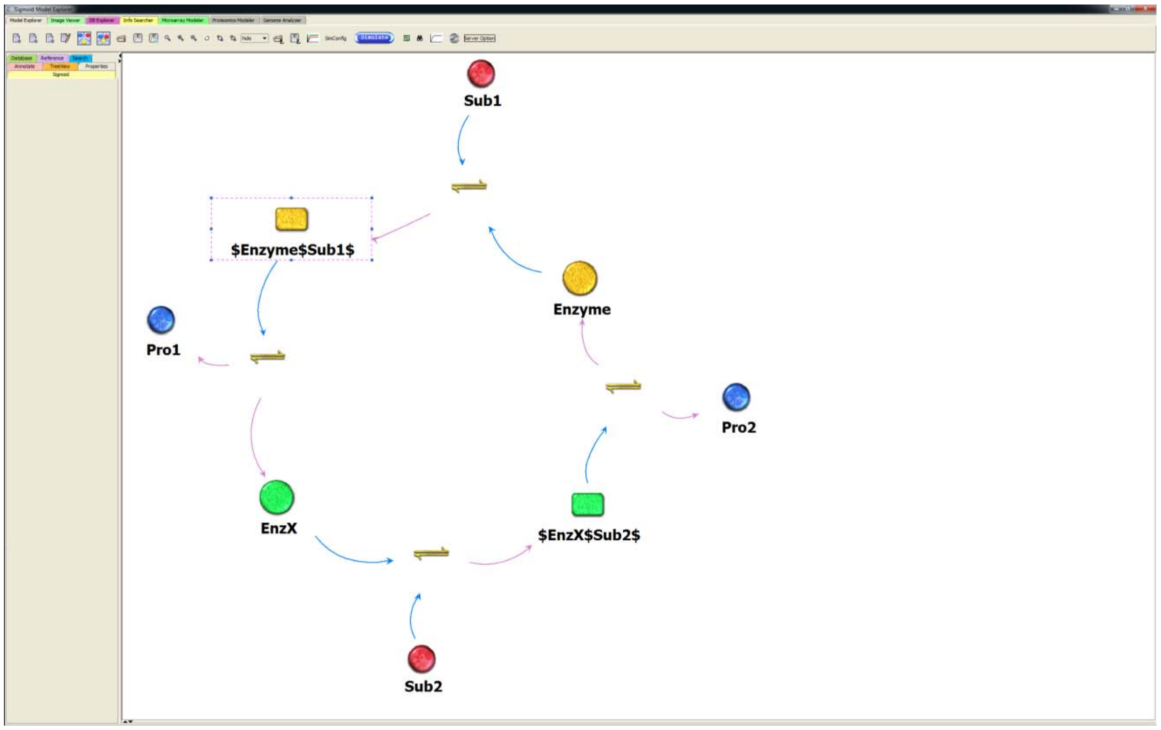
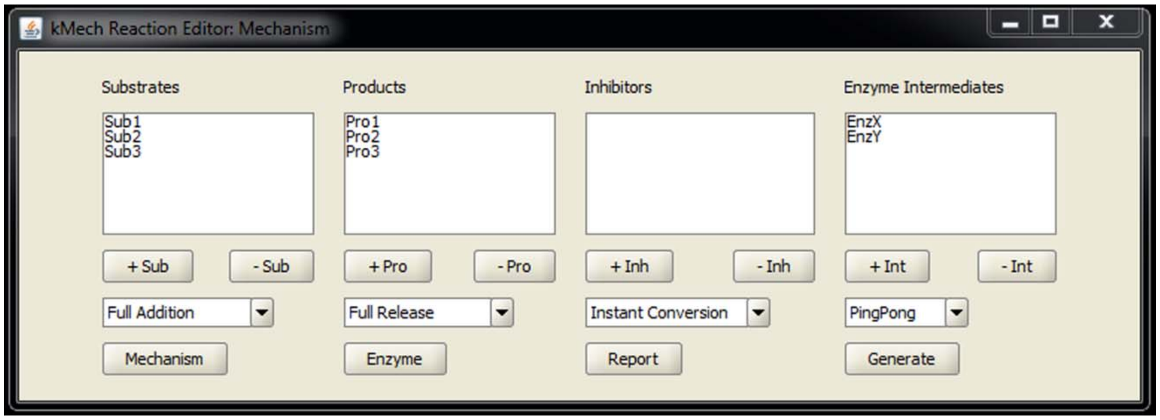


**Figure 3.19 SME: Classic BiBi Ping Pong reaction. Substrates (in red) are sequentially converted to product (in blue) leaving the enzyme in a changed state.**

Enzyme states are represented as circles and complexes as rectangles. A constraint that limits SigMech but is not intrinsic to its architecture is that an enzyme intermediate state must exist for each catalysis reaction that is present. Also, splitting or combining of substrates into product hasn't been implemented. Version 1.0 of SigMech requires that the number of substrates, products and enzyme states all be equal. This version will allow any number of sequential intermediates to function, provided they convert one substrate to one product.

## 3.7.6 *TerTer Ping Pong Example*

Note in Figure 3.20 there are two intermediates listed. The reaction is a TerTer reaction. If the free enzyme state is included, there are a total of three states. SigMech supports any number of ping pong steps provided that the number of substrates, products and total number of enzyme states is equal.



| Substrates | Products | Inhibitors | Enzyme Intermediates |
|---|---|---|---|
| Sub1 | Pro1 | | EnzX |
| Sub2 | Pro2 | | EnzY |
| Sub3 | Pro3 | | |
| Full Addition | Full Release | Instant Conversion | PingPong |

**Figure 3.20 SigMech Interface: TerTer Ping Pong.**

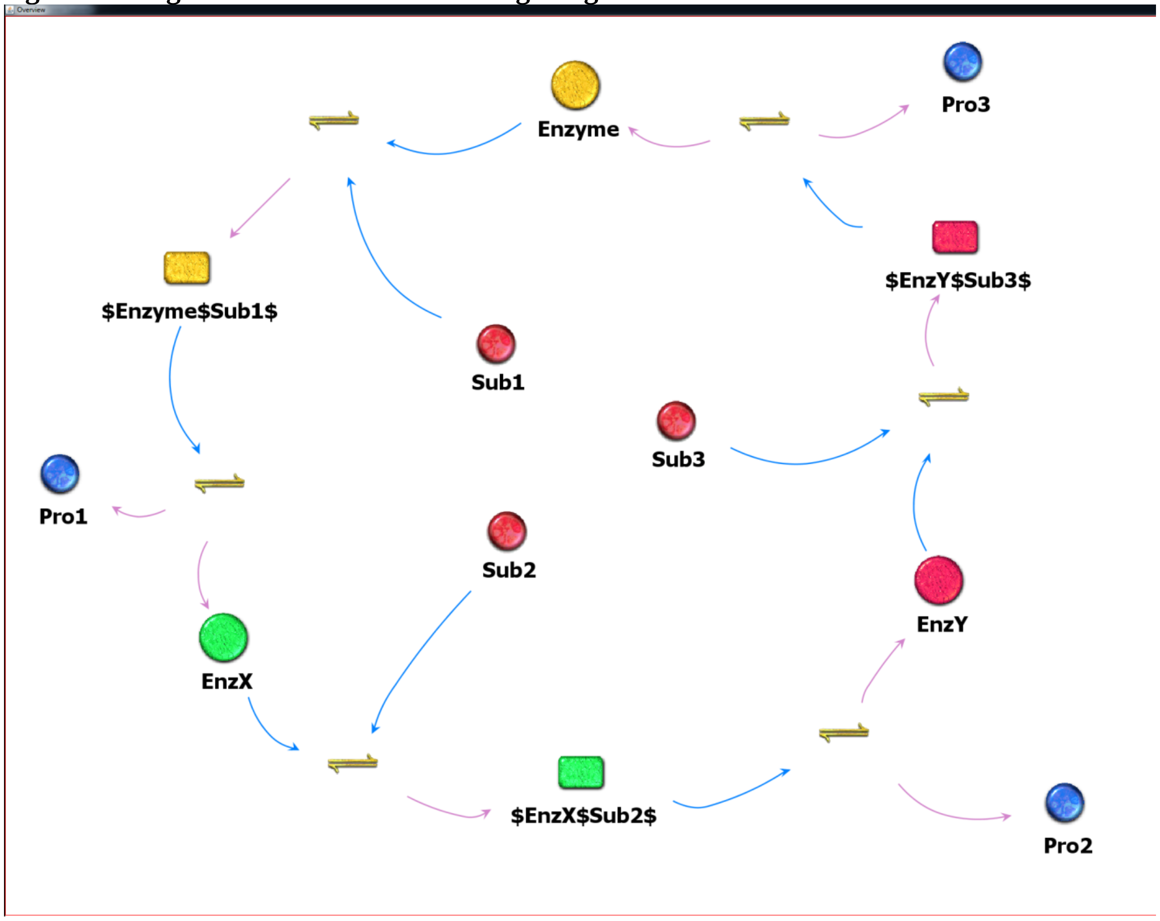
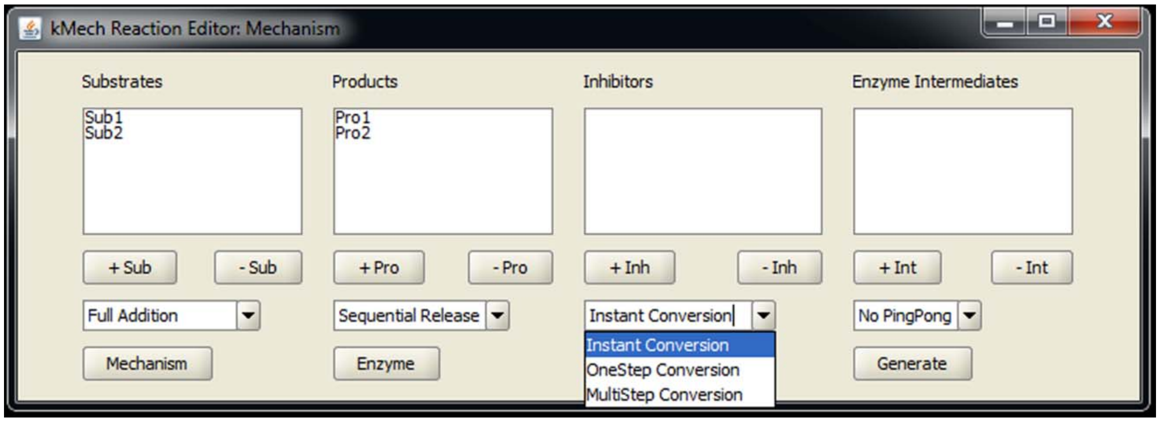


**Figure 3.21 SME Overview Panel: TerTer Ping Pong reaction mechanism. Substrates (in red) are sequentially converted to product (in blue) leaving the enzyme in a changed state. Ping pong mechanisms can be built for any number of stages provided that the number of substrates, products and enzyme states are all equal.**

### 3.7.7 *Conversion Phase*

The conversion menu has selections for instant conversion, one step, and multi-step conversion. The terminology might need to be adjusted in the user interface. "Instant Conversion" might not be the best way to name it but, SigMech Version 1.0 displays it as such.

Instant conversion is implemented in SigMech as a bidirectional reaction with corresponding forward and reverse rates. This choice of representing the catalytic process by one bidirectional reaction was made because it allows for greater flexibility. In a case where the intention is to model without a reverse reaction, the reverse rate can simply be set to zero. The simplest representation for a catalytic reaction, as reflected in the schema is to have a reaction with one forward rate. Using one forward rate to model a reaction might approximate the conditions of catalytic activity where the forward rate is many order of magnitudes faster than the reverse rate for the reaction. Even though a reverse reaction and corresponding rate exists in nature, this reverse reaction is ignored in many cases. Biologists may choose to model this reverse reaction, and reactions are available in xCellerator and the Sigmoid schema, as a TwoStageCatalytic Reaction, to facilitate such use. A reaction equation equivalent appears in Section 3.2.1.



Sub1
Sub2

Pro1
Pro2

Full Addition

Sequential Release

Instant Conversion

Instant Conversion
OneStep Conversion
MultiStep Conversion

No PingPong

**Figure 3.22 The Conversion menu.**

A one-step conversion is a modeling feature added in order to emulate the three stage catalytic modeling option available in Sigmoid and xCellerator (although not specifically termed as such in xCellerator). The three stage catalytic reaction, shown in Figure 3.23, consists of three consecutive mass action reactions, substrate binding to enzyme, conversion of the substrate to product while complexed, and then subsequent release of product from complex.

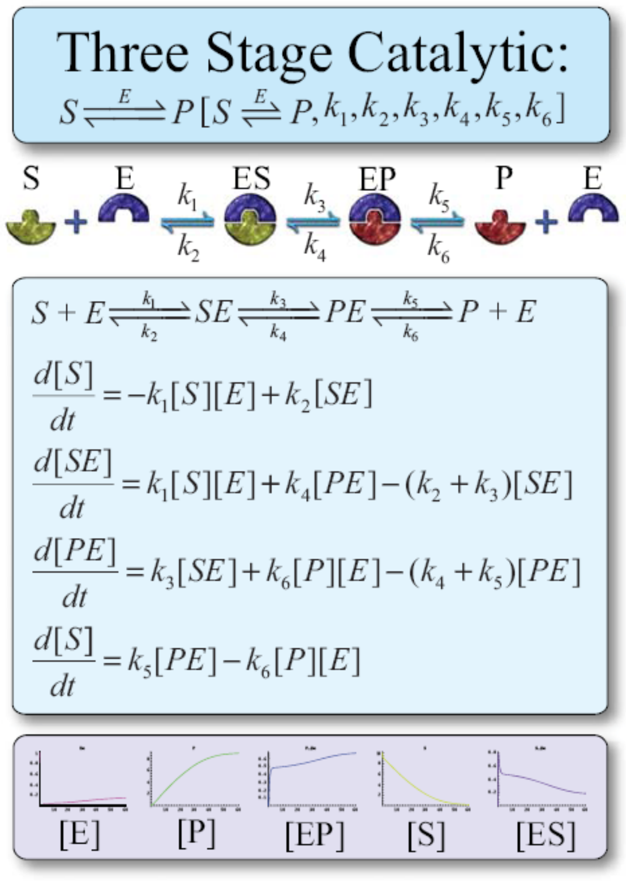
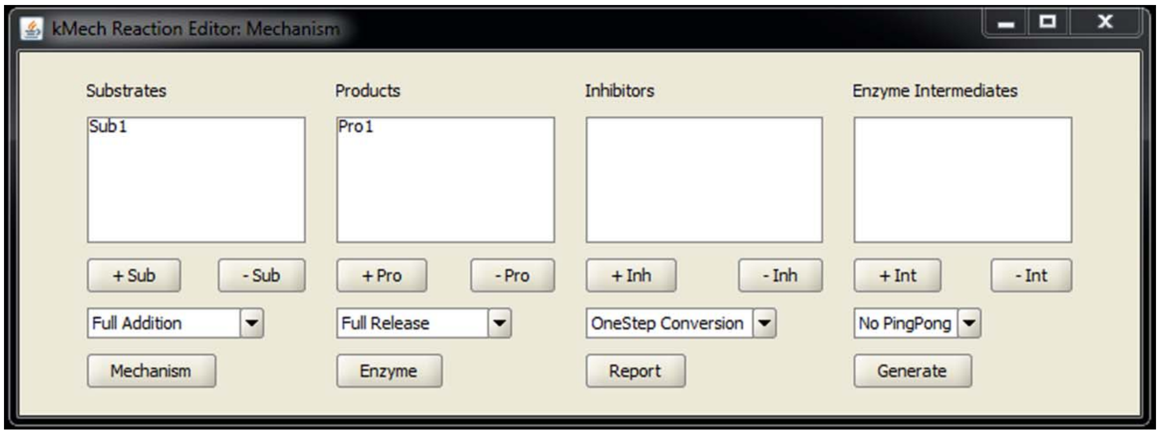

Three Stage Catalytic:
$S \overset{E}{\rightleftharpoons} P[S \overset{E}{\rightleftharpoons} P, k_1, k_2, k_3, k_4, k_5, k_6]$
S
E
ES
EP
P
E
$k_1$
$k_2$
$k_3$
$k_4$
$k_5$
$k_6$
$S + E \underset{k_2}{\overset{k_1}{\rightleftharpoons}} SE \underset{k_4}{\overset{k_3}{\rightleftharpoons}} PE \underset{k_6}{\overset{k_5}{\rightleftharpoons}} P + E$
$\frac{d[S]}{dt} = -k_1[S][E] + k_2[SE]$
$\frac{d[SE]}{dt} = k_1[S][E] + k_4[PE] - (k_2 + k_3)[SE]$
$\frac{d[PE]}{dt} = k_3[SE] + k_6[P][E] - (k_4 + k_5)[PE]$
$\frac{d[S]}{dt} = k_5[PE] - k_6[P][E]$
[E]
[P]
[EP]
[S]
[ES]

**Figure 3.23 The ThreeStageCatalytic MathReaction in the schema illustrates the implementation of a central "conversion " reaction where enzymatically bound substrates are converted to product. Release of product is modeled as a third reaction step.**

Reverse rates for each reaction are available for a total of six reaction rates, three forward and three reverse. This modeling option is toggle-able for any enzyme mechanism reaction. The conversion menu appears right under the inhibitors list as shown in Figure 3.22 . There is no conceptual relation between the conversion menu and the inhibitors. The positioning is simply coincidental.

A simple case example can be seen in Figure 3.24 and Figure 3.25 where one substrate is converted to one product though this mechanism.



Sub1

Pro1

Full Addition

Full Release

OneStep Conversion

No PingPong

Figure 3.24 SigMech UI: UniUni reaction with OneStep Conversion activated.

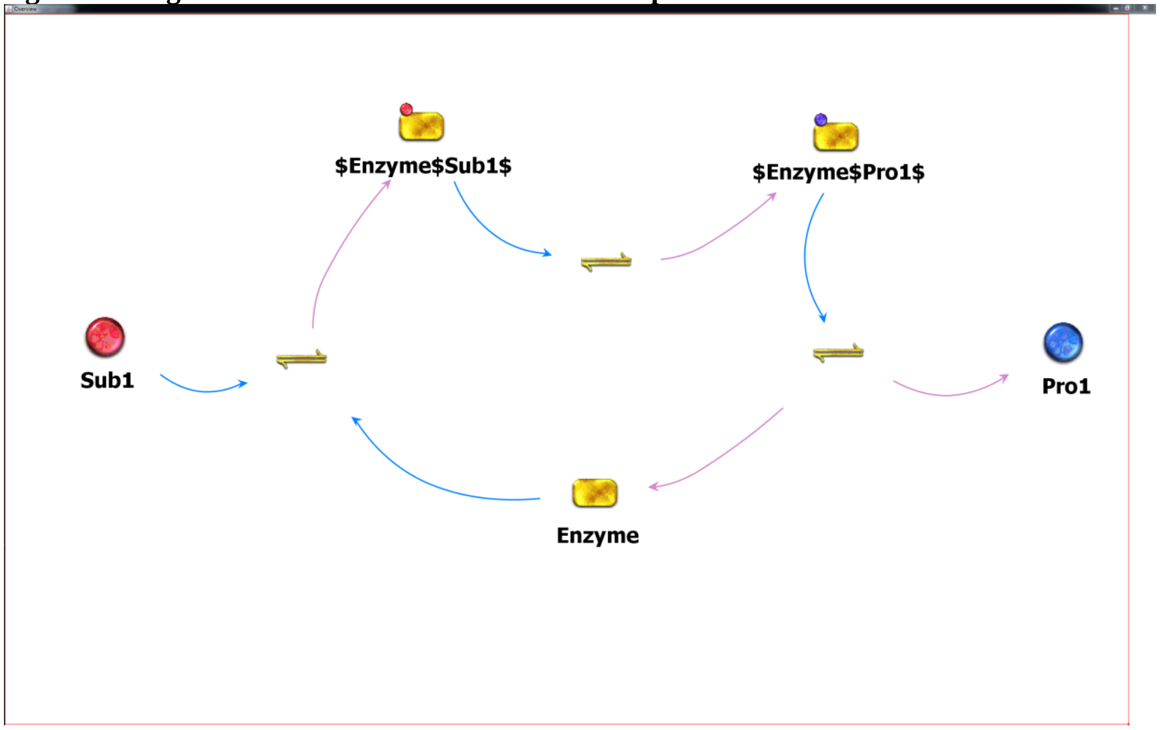
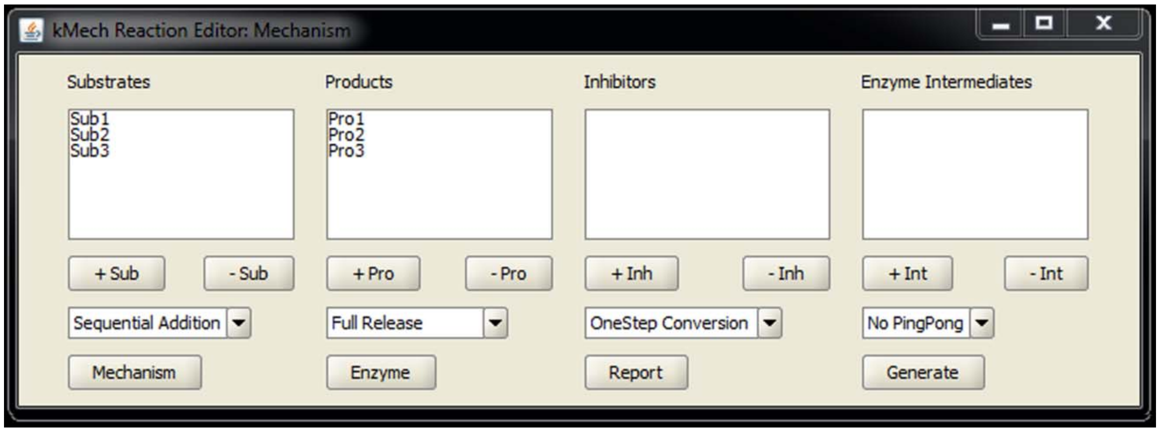


Figure 3.25 SME view: A UniUni Reaction with a OneStep Conversion phase.

This case is the same as the ThreeStageCatalytic reaction.  Since there are bidirectional mass action reactions on the attached to these BiologicalReactions, there are six overall rates, three foreward and three reverse.  More involved examples show that this feature can be implemented independently of the addition and release strategies.  For instance, Figure 3.12 illustrated an example of a Ter-Ter sequential addition and full release reaction.  A version of this reaction with a one step conversion phase is shown in Figure 3.26 and Figure 3.27.



Substrates: Sub1, Sub2, Sub3
Products: Pro1, Pro2, Pro3
Sequential Addition
Full Release
OneStep Conversion
No PingPong

**Figure 3.26 SigMech Interface: TerTer  Sequential Addition Full Release One Step Conversion.**

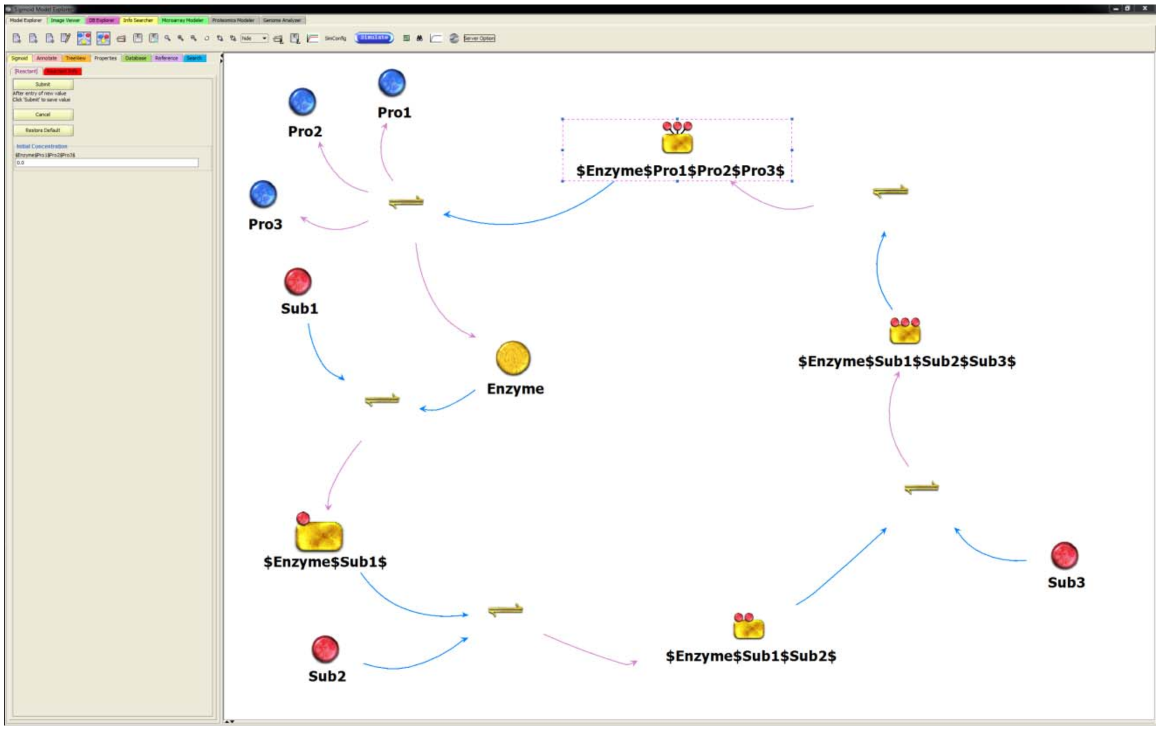
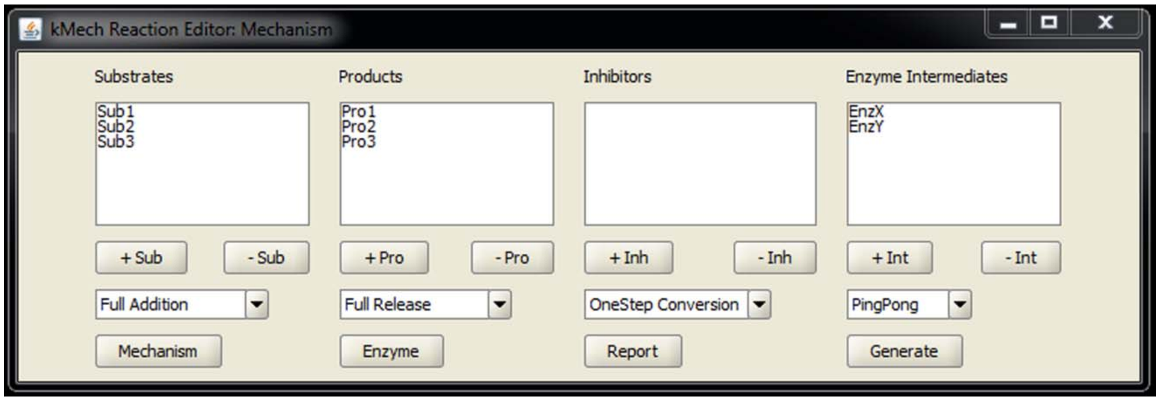


**Figure 3.27 SME: TerTer Sequential Addition Full Release with a One Step Conversion.**

Even further, the one-step conversion can be implemented in cases for Ping Pong class reactions. If the example shown in Figure 3.21 used as a starting point and a "One Step Conversion" is added we the result shown in Figure 3.29. Each binding of a substrate to an enzyme state yields a conversion reaction in which substrate is converted to product, and a subsequent release of product reaction.



| Substrates | Products | Inhibitors | Enzyme Intermediates |
|---|---|---|---|
| Sub1 | Pro1 | | EnzX |
| Sub2 | Pro2 | | EnzY |
| Sub3 | Pro3 | | |
| Full Addition | Full Release | OneStep Conversion | PingPong |

**Figure 3.28 SigMech Interface: TerTer Ping Pong with One Step Conversion.**

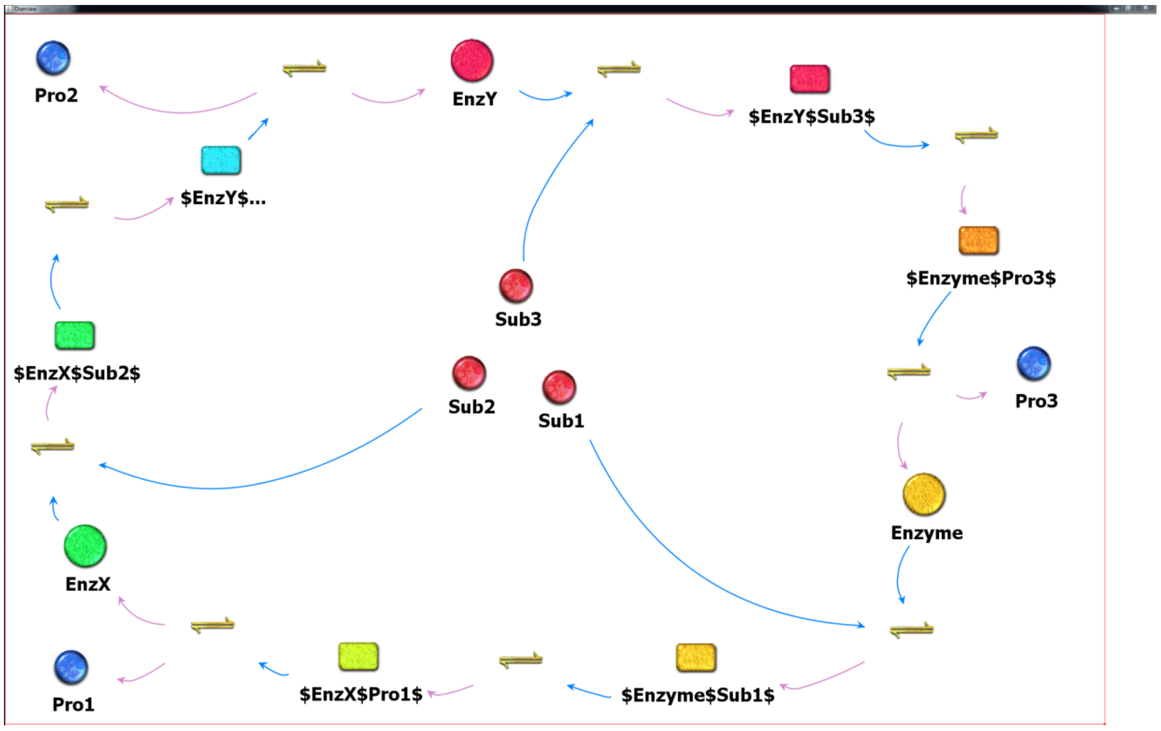


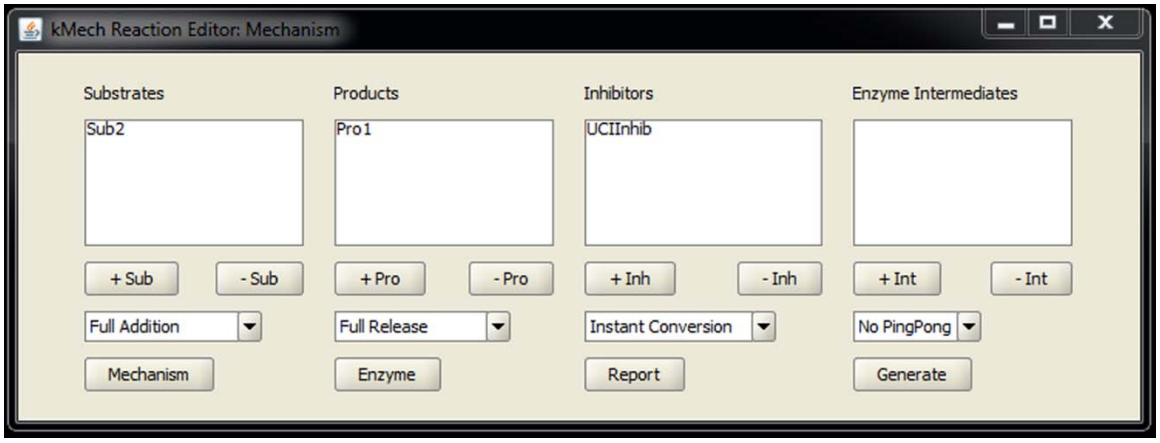
**Figure 3.29 SME Overview Panel: TerTer Ping Pong One with Step Conversion.**

In Figure 3.22, the “Multi Step Conversion” option is viewable in the GUI version 1.0 but the corresponding strategy is hypothetical and has not been implemented, so selecting this option will yield no results. This option could be used for processes that have either multiple energetic barriers to overcome or some deeper complexity, so the option was included as a possibility. Further collaboration with modeling biologists will reveal whether this is a useful option.

## 3.7.8 *Inhibition Examples:*

### *3.7.8.1 Competitive Inhibition Example*

Competitive inhibitors can be added to the inhibition list. Competitive inhibitors bind free enzyme and prevent substrate binding. (Please note in the example, the icon choice could have been better, but the network is correct in Figure 3.31). Any number of competitive inhibitors may be specified.



| Substrates | Products | Inhibitors | Enzyme Intermediates |
|---|---|---|---|
| Sub2 | Pro1 | UCIInhib | |
| Full Addition | Full Release | Instant Conversion | No PingPong |

Figure 3.30 SigMech UniUni with Competitive Inhibitor.

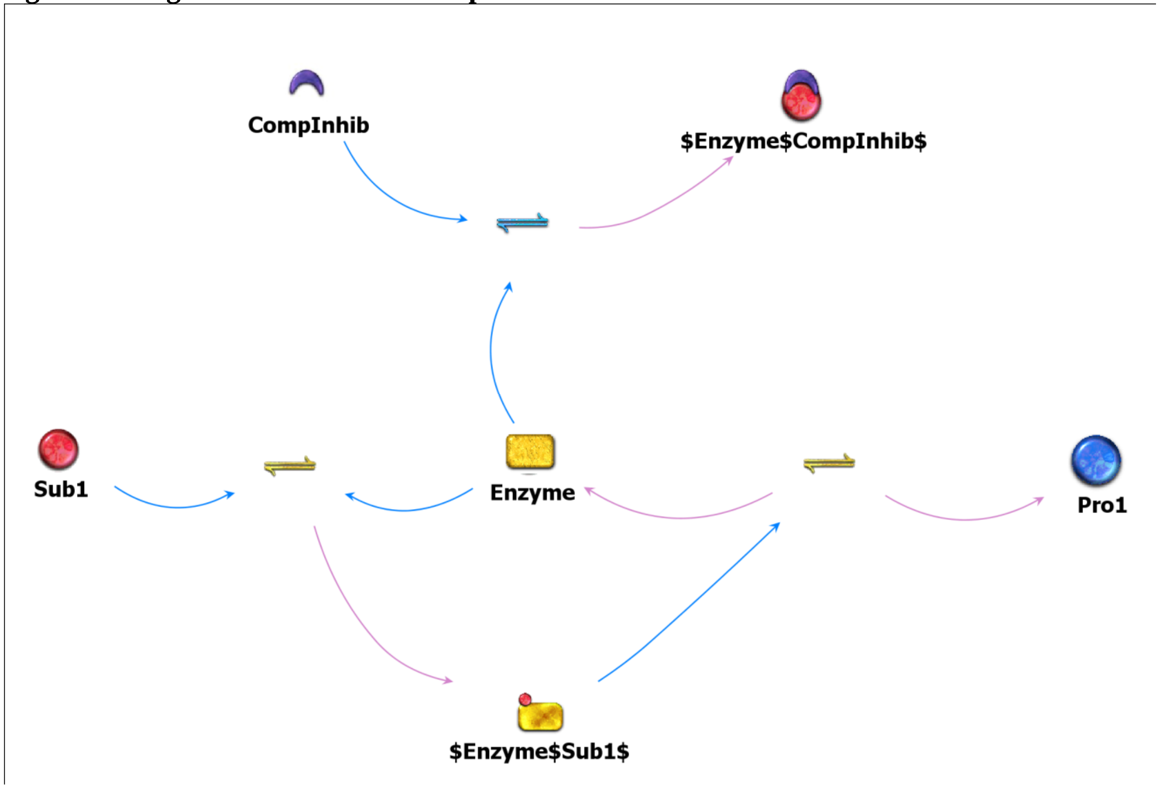
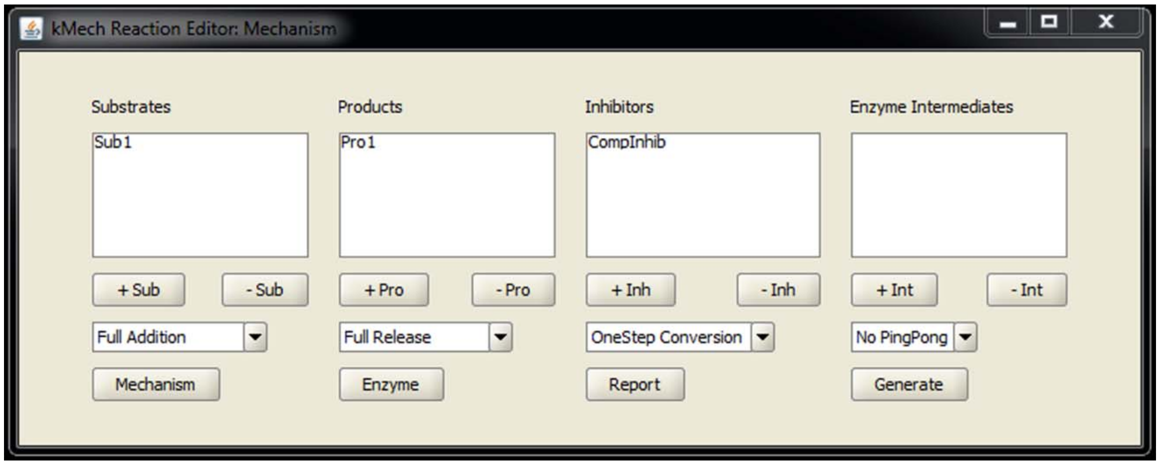


**Figure 3.31 SME Overview Panel: UniUni with Competitive inhibitor. Competitive inhibitors bind free enzyme thus preventing substrate binding.**

Competitive inhibitors, since they act on free enzyme, may operate independently of one another. Many may be present and they function irrespective of addition strategy, release strategy, conversion strategy or cases of ping pong reactions. A simple example of this independence is illustrated in Figure 3.32 where a "One Step Conversion" reaction is active. The network is illustrated in Figure 3.33.


kMech Reaction Editor: Mechanism

Substrates: Sub1
Products: Pro1
Inhibitors: CompInhib
Enzyme Intermediates:

Full Addition | Full Release | OneStep Conversion | No PingPong

**Figure 3.32 SigMech UniUni with Competitive Inhibitor and a One Step Conversion.**

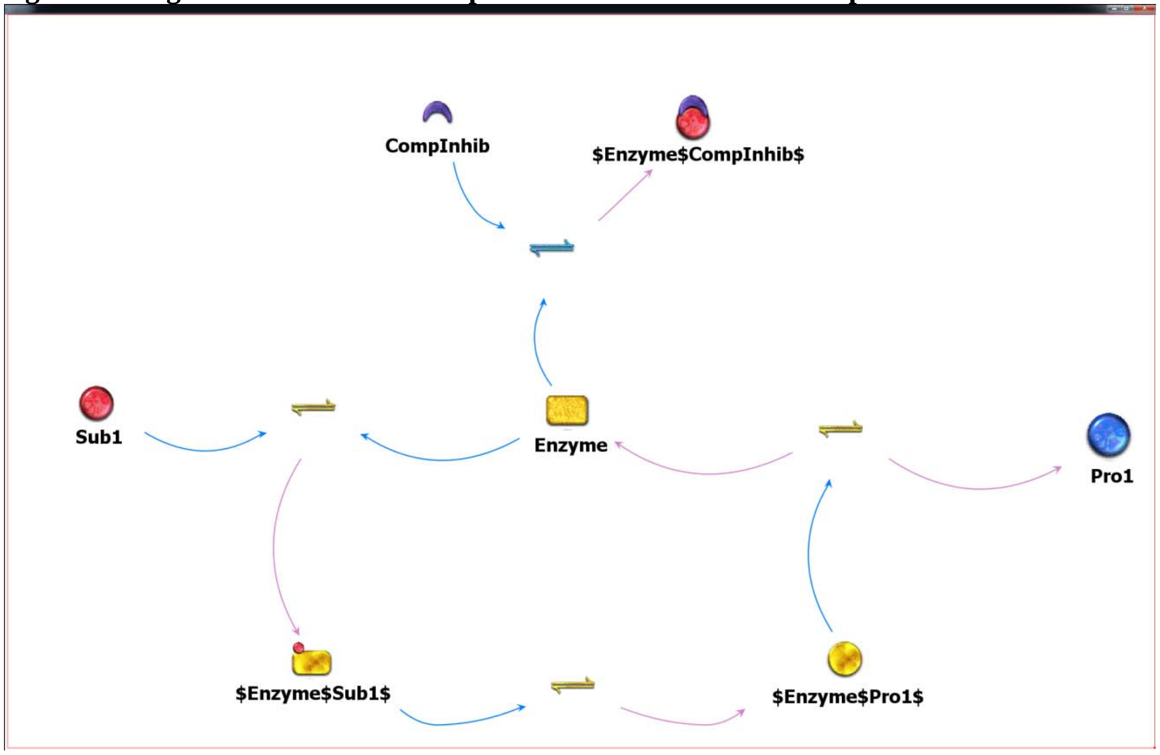
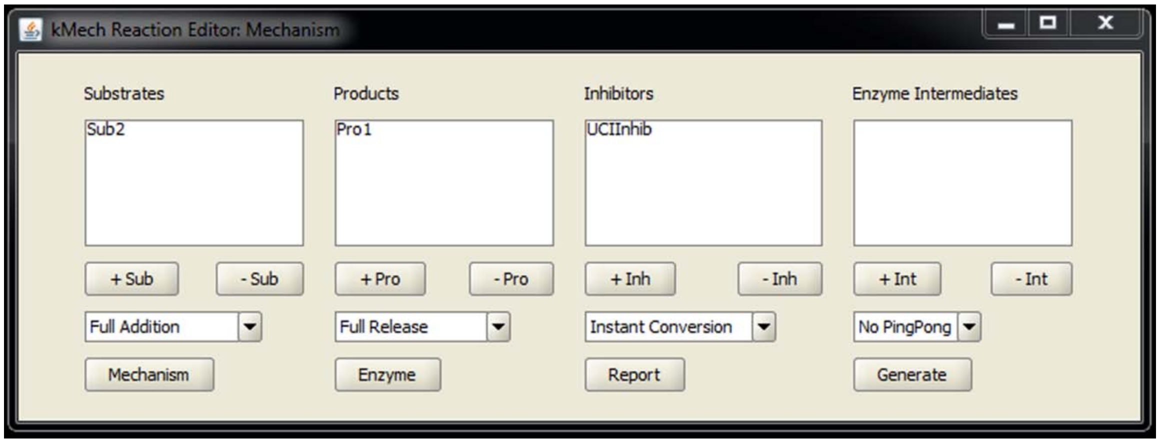


**Figure 3.33 SME Overview Panel: UniUni with Competitive inhibitor and a One Step Conversion. The enzyme-substrate to enzyme-product conversion appears in yellow at the bottom.**

### *3.7.8.2 UnCompetitive Inhibition Example*

UnCompetitive inhibitors target an enzyme-substrate complex. A simple UniUni case is exhibited in Figure 3.34 . When specifying an Un-Competitive Inhibitor, the user interface asks for a target. This target should correspond to enzyme/substrate complex that the user intends to be inhibited. SigMech does not currently enforce any rules about this specification. For instance if the user accidentally specified a product, the product would get bound by the inhibitor. Also, if the name does not match a reactant, the mechanism may be unusable. The user-assigned name of this complex must conform to the rule \$EnzymeName\$ReactantName\$ (+ AdditionalReactantName\$ +....) A next version of SigMech could pre-parse the reaction network and generate consistently named complexes that can be referenced by the software to avoid this complication.



Sub2

Pro1

UCIInhib

Full Addition

Full Release

Instant Conversion

No PingPong

**Figure 3.34 SigMech: UniUni with an UnCompetitive Inhibitor (UCI).**

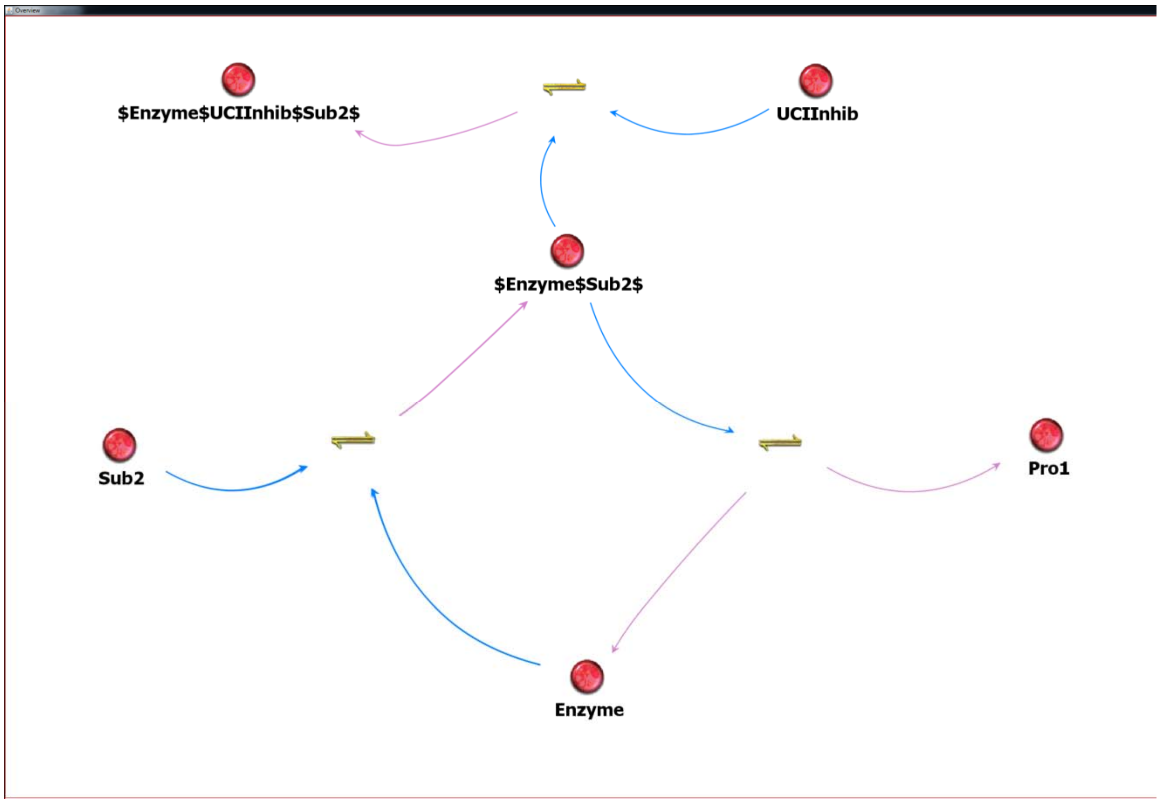
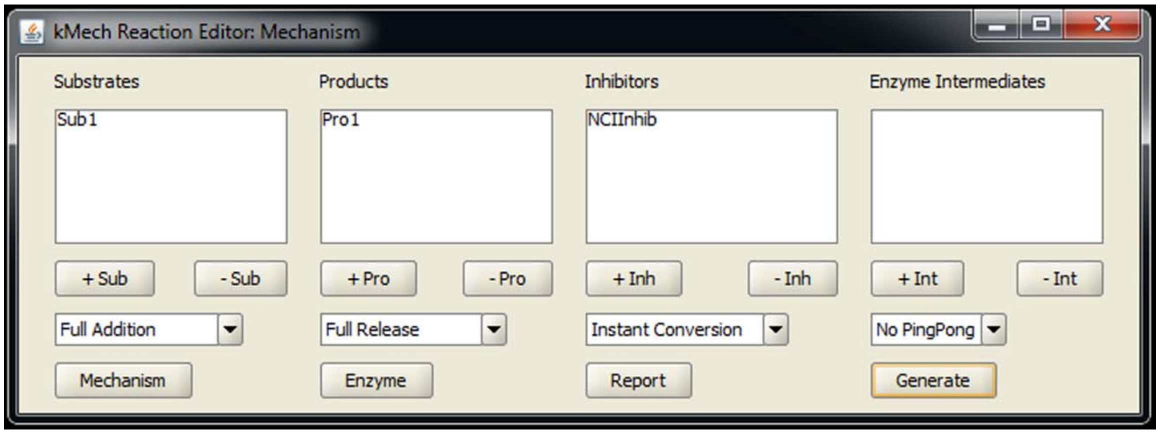


**Figure 3.35 SME Overview: UniUni with an UnCompetitive Inhibitor (UCI). UCIs bind enzyme-substrate complexes to inhibit the activity of an enzyme.**

### *3.7.8.3 NonCompetitive Inhibition Example*

NonCompetitive Inhibition is the third option for inhibitors in SigMech. A NonCompetitive inhibitor (NCI) can bind either free enzyme or enzyme already bound to substrate. A NonCompetitive inhibitor may not completely shut down the enzyme's activity, but rather it may have a residual activity. The corresponding NCI reactions are shown in Figure 3.37. . Since the GUI does not yet enforce rules on the input parameters, various addition and release strategies can produce unwanted or erroneous reaction results. Certain combinations of strategies may produce a set of reactions that have no bearing on the particular biochemistrys at hand for a particular intended enzyme mechanism if the user does not carefully construct the mechanism. It is left to the expert user to use scrutiny when applying this form of inhibition. In multiple subsrtate, multiple product reactions, the inhibition must be set up carefully as the inhibitor procedure generates reactions for the target complex formation and generates residual product from that. A next version of SigMech could check specified inputs and inhibitions to make sure the mechanism is valid.



Substrates: Sub1
Products: Pro1
Inhibitors: NCIInhib
Full Addition
Full Release
Instant Conversion
No PingPong

**Figure 3.36 SigMech UniUni reaction with a NonCompetitive Inhibitor(NCI).**

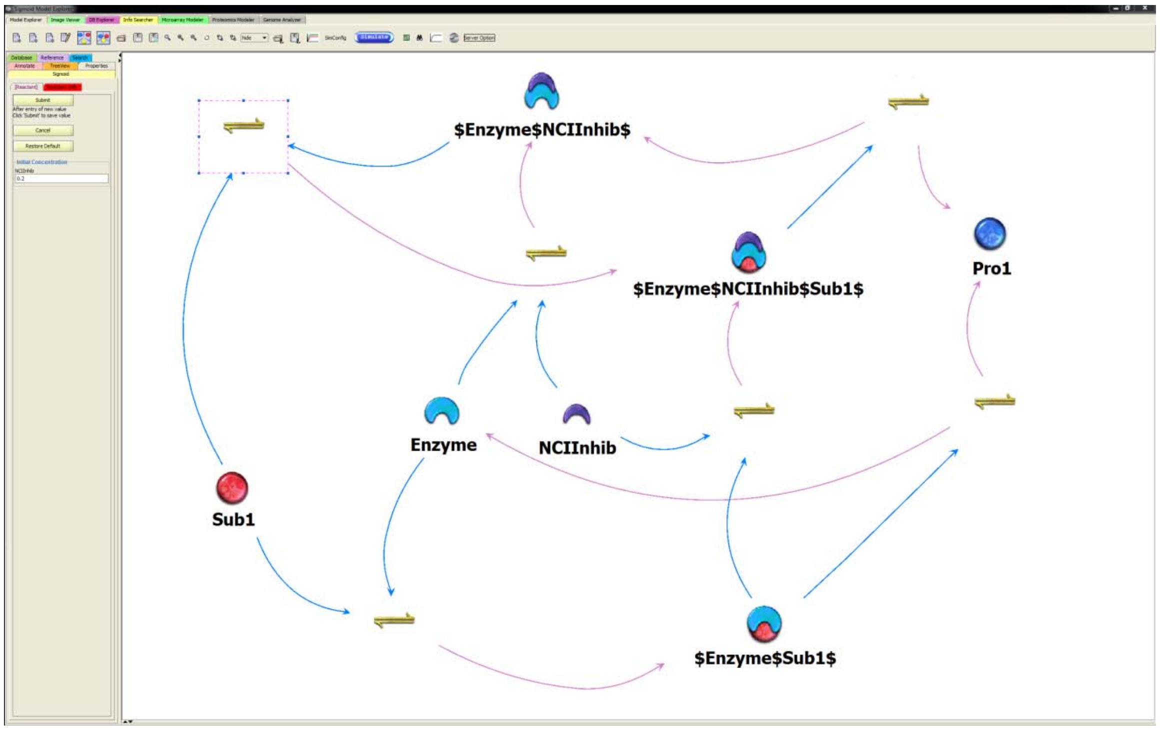


**Figure 3.37 SME: UniUni reaction with a NonCompetitive Inhibitor (NCI). NonCompetitive inhibitors may possess a residual enzymatic activity that generates product from inhibitor bound enzyme. NCIs can either bind free enzyme or enzyme-substrate complexes.**

## *3.8 Sigmoid Model Explorer integration.*

SigMech can be run as a standalone application or within SME. SigMech version 1.0 generates a Sigmoid Model class and adds the respective sub-reactions, reactants and relevant parameters to the Model. Each BiologicalReaction SigMech generates has a corresponding BidirectionalMassAction MathReaction with rates initialized to arbitrary values. When SigMech is run from SME, the reaction network is displayed as a model in the network layout view panel. The rate parameters can be edited in the reaction panel in SME (item C in Figure 1.4). As with any Sigmoid model, the model can be sent to Mathematica/xCellerator for simulation. When SigMech is run as a standalone application, the model reaction network can be examined by pressing the "Report" button on the interface (Figure 3.5). The Report button will also function when SigMech is run from within SME and reports to the Java console, if open.

The next step in SME integration would be to utilize the ReactionGroup class in the schema by adding all of a particular enzyme mechanism's sub-reactions to the group instead of instantiating a separate model. SME should then be able to hide the sub-network of reactions from a larger pathway model. Inputs and outputs of the mechanism would be viewable at the higher level pathway view but the sub-reactions would be represented by one reaction icon. In this way, the user could zoom in to view the details of the mechanism and zoom out to hide the details.

One intended feature of SigMech is to generate a BiologicalReaction network and reference MathematicalReactions for the kinetics. Presently, SigMech 1.0 assigns strictly BidirectionalMassAction Reactions with default rates to each BioReaction, as this option provides everything necessary to represent all previous Sigmoid Complex BioReactions.

Rate parameters from the resulting network of reactions can be edited in SME, but the intention is to allow a user to edit the MathReactions referenced by BioReactions from within SME so that alternate kinetics can be applied to one or many of the Mechanism subreactions.  SME does have model creation and editing capabilities but it's not clear that this feature can be implemented properly without a SME update.  Alternatively a next version SigMech could implement another feature layer providing greater access to other MathReactions, such as the AlgebraicPassthrough reactions and a matrix of initialization rates.

## *3.9 Conclusion*

The highly parameterized reaction types of SigMech, together with the kMechReaction classes, provide a parameterized representation that encompasses all previous Compound reactions that were present in the Sigmoid schema, and can generate these and other novel reaction mechanisms from the cross product of the valid SigMech input parameters.  This development effectively renders kMech integration a solved problem within the scope of previous reaction motifs, and will vastly reduce the need for frequent updates of the schema, database, middleware translation and GUI.  SigMech combines the reaction motifs of previous kMech reaction mechanisms, with the conversion phase that was implemented in the three-stage xCellerator reaction (ThreeStageCatalytic in the Sigmoid schema) yielding a greater parameter space of possible enzymatic reaction mechanisms that can be generated on the fly inside SME, and that can consequently be simulated by Mathematica/ xCellerator.

# Chapter 4. Population Strategies for the Sigmoid Database.

In chapter two we discussed the structure and schematics of the Sigmoid database and Java class API. In this chapter we will discuss populating the database pathway with models of biological significance.

Pheno [Cheng *et al.*2005] translates the Sigmoid schema diagrams into PostgreSQL database tables and a corresponding set of Java Classes that comprise the Sigmoid API Java Classes. A direct Object Relational Bridge (OJB) mapping exists between the Sigmoid Java classes and the relational database. These Java Classes can be used to compose Sigmoid pathway models. Two main strategies exist for populating the Sigmoid database with models. A straightforward method is to code Java models directly and store the models in the database via the OJB interface to the database. This method is where a user focuses on the production of one particular pathway. Another strategy is to build automated populator programs, also through OJB, that are designed to translate other pathway markup languages or database resources via web services to the Sigmoid representation. We will discuss individual models coded by users in Section 4.1. We will address the implementation of population programs in Section 4.2

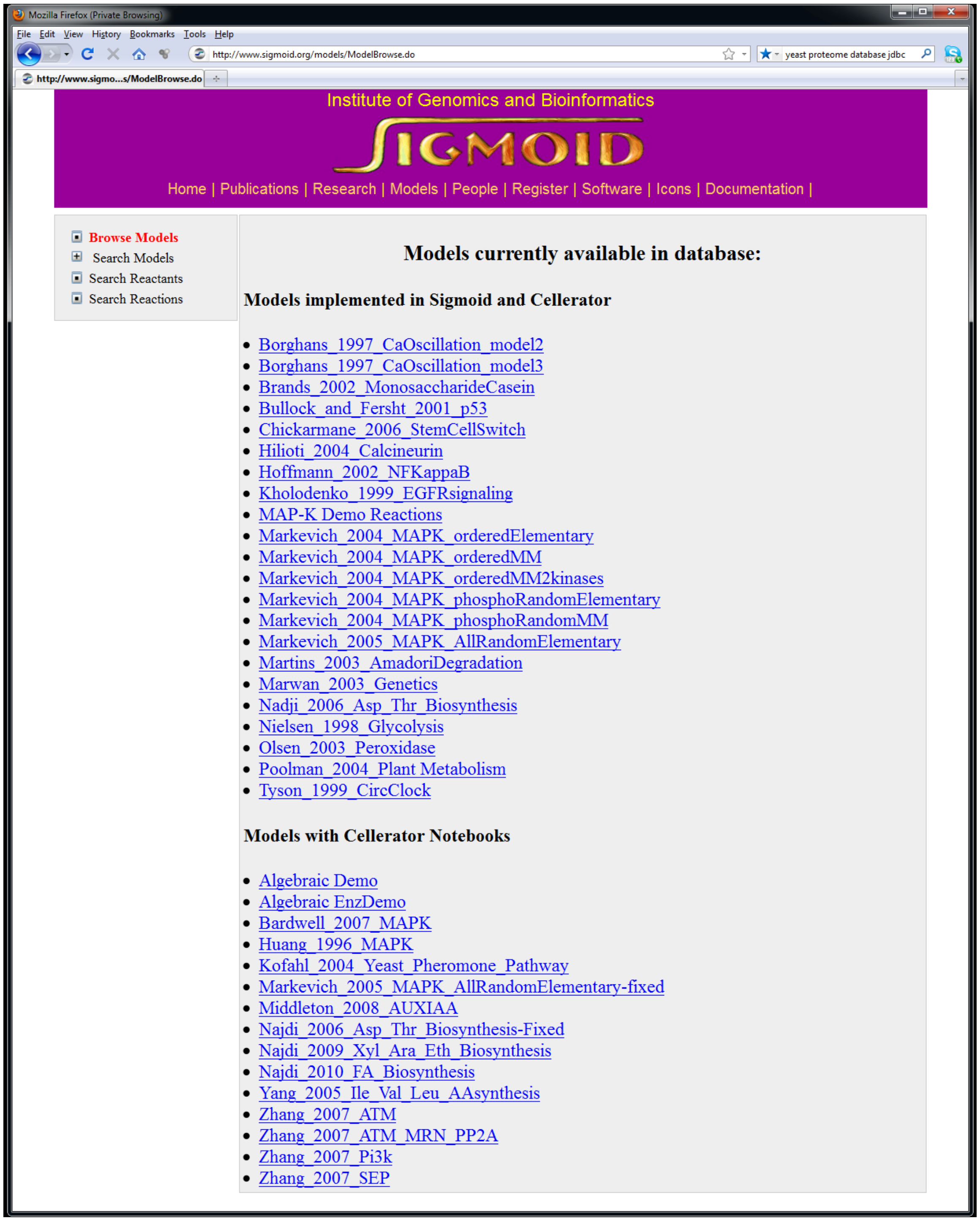
## 4.1 *User Coded Models*

Although any Java development environment would suffice, the Eclipse Java development platform (www.eclipse.org) was used for the Sigmoid population classes. Eclipse possesses built in Concurrent Versions System (CVS) interface functions that facilitated group population efforts. The CVS repository at sigmoid.sourceforge.net proved to be a useful tool for coordinating the development efforts of several team members for the set of Sigmoid pathway population programs and proved to be a viable repository for other portions of the Sigmoid code.

We developed a set of template classes (Java programs) that are designed for pathway model construction. These classes have customized methods for model, reactant, reaction, parameter set and knowledge source creation. There are administrative classes designed to delete and populate the database with sets of models. There are reporting classes designed to survey the database contents. The ModelEmitter class, which is fairly useful for model development and debugging, is designed to report model reactant and reaction attributes to the Java console. Some initial code to query the database for a model and to enter a model into the database was provided by Lucas Scharenbroich. Some methods from these classes were reworked and expanded over several iterations to comprise a portion of the template class functionality.

These template classes were used by Sigmoid team members to generate Sigmoid pathway models. Although several team members created models for Sigmoid, construction of the template classes, final debugging of pathway models, entry of models into, and administration of the database was performed by the author in the course of work for dissertation.

The generative version of Sigmoid has been successfully populated with over twenty published models that range from simple molecular interactions to complex cell fate decision networks.  A comprehensive list of the models is provided in Figure 4.1. A majority of the models in the database focus on virtual representation of intracellular pathways that include examples in signaling, metabolism, the cell cycle, and gene regulation.


Mozilla Firefox (Private Browsing)
File Edit View History Bookmarks Tools Help
http://www.sigmoid.org/models/ModelBrowse.do
yeast proteome database jdbc
http://www.sigmo...s/ModelBrowse.do
Institute of Genomics and Bioinformatics
SIGMOID
Home | Publications | Research | Models | People | Register | Software | Icons | Documentation |
Browse Models
Search Models
Search Reactants
Search Reactions

**Models currently available in database:**

**Models implemented in Sigmoid and Cellerator**

- Borghans_1997_CaOscillation_model2
- Borghans_1997_CaOscillation_model3
- Brands_2002_MonosaccharideCasein
- Bullock_and_Fersht_2001_p53
- Chickarmane_2006_StemCellSwitch
- Hilioti_2004_Calcineurin
- Hoffmann_2002_NFKappaB
- Kholodenko_1999_EGFRsignaling
- MAP-K Demo Reactions
- Markevich_2004_MAPK_orderedElementary
- Markevich_2004_MAPK_orderedMM
- Markevich_2004_MAPK_orderedMM2kinases
- Markevich_2004_MAPK_phosphoRandomElementary
- Markevich_2004_MAPK_phosphoRandomMM
- Markevich_2005_MAPK_AllRandomElementary
- Martins_2003_AmadoriDegradation
- Marwan_2003_Genetics
- Nadji_2006_Asp_Thr_Biosynthesis
- Nielsen_1998_Glycolysis
- Olsen_2003_Peroxidase
- Poolman_2004_Plant Metabolism
- Tyson_1999_CircClock

**Models with Cellerator Notebooks**

- Algebraic Demo
- Algebraic EnzDemo
- Bardwell_2007_MAPK
- Huang_1996_MAPK
- Kofahl_2004_Yeast_Pheromone_Pathway
- Markevich_2005_MAPK_AllRandomElementary-fixed
- Middleton_2008_AUXIAA
- Najdi_2006_Asp_Thr_Biosynthesis-Fixed
- Najdi_2009_Xyl_Ara_Eth_Biosynthesis
- Najdi_2010_FA_Biosynthesis
- Yang_2005_Ile_Val_Leu_AAsynthesis
- Zhang_2007_ATM
- Zhang_2007_ATM_MRN_PP2A
- Zhang_2007_Pi3k
- Zhang_2007_SEP

**Figure 4.1 A comprehensive list of pathway models stored in the Sigmoid database can be browsed in the Models section of the Sigmoid website at www.sigmoid.org. A majority of the models in the database focus on virtual representation of intracellular pathways that include examples in signaling, metabolism, the cell cycle, and gene regulation.**

### 4.1.1 *Sigmoid Database Population From xCellerator notebooks.*

Since Sigmoid leverages the numerical computational power of Cellerator and its successor xCellerator as its primary simulation platform, existing xCellerator notebooks have been an important source of pathway models. Early in the development of Sigmoid a survey of existing Cellerator notebooks was taken and an attempt was made to adapt the schematic representation of Sigmoid so that the greatest compatibility between Sigmoid and Cellerator could be achieved. The classes in the MathematicalReaction section of Chapter 2 reflect the direct compatibility between Sigmoid and xCellerator functions.

For the greatest part, all user coded models present in the public version of the Sigmoid database exist as Cellerator notebooks. These notebooks were either part of the Cellerator notebook library, or the Cellerator notebooks were created as an aid in constructing the Sigmoid Java coded pathway models. The notebooks provide a valuable testing reference for Sigmoid models and proof that the pathway model will function in xCellerator. They are not absolutely necessary, but as there is such a great parallelism between existing Sigmoid simulation capability and Cellerator, generating notebooks for every model has proven to be a useful device.

### 4.1.2 *SBML as a Source of Pathway Models.*

Another valuable source of pathway models comes from Systems Biology Markup Language SBML. SBML is an XML extended interchange format designed to represent computational models of biological processes. SBML strives to provide a common interchange format for the variety of tools designed to model biological networks. SBML has been developed in “levels” to provide stable releases of the format for developers. Currently SBML possesses three levels of development [Hucka *et al.* 2003] [Finney and Hucka 2003]. Information about SBML can be found at www.SBML.org.

The Biomodels.net database served as a valuable repository for peer-reviewed SBML models based on publications. Our team surveyed this repository in the early stages of its release and selected several models from the Biomodels database that were to be converted and stored in the Sigmoid database.

Converting SBML formatted models into Sigmoid models has presented an interesting set of challenges, both for user coded models and the automated SBML reader to be discussed in the next section. There are a few features of SBML models, such as SBML Events and Functions, which are not entirely supported by Sigmoid. Some steps have been taken to incorporate storage for them in the schema (in the form of fields to store the strings), but implementation in the form of translation to xCellerator has not commenced nor has an attempt been made to construct models with these features. A decision was made to pursue the conversion of other SBML models that had a feature set more easily incorporated into the Sigmoid classes.

A particularly interesting challenge in converting SBML to Sigmoid is that the kinetic rate laws stored in SBML are in the form of algebraic expressions. This differs from the current approach of Sigmoid. As discussed in Section 2.9, Sigmoid handles the kinetics of reaction equations by the storing sets of parameterized inputs for xCellerator. These input parameters are passed through the middleware and converted to xCellerator functions which are composed into notebooks. The notebooks are consequently sent to and processed by Mathematica/xCellerator thereby generating simulation output [Cheng 2005][Compani, Su *et al.* 2010].

The challenge lies in converting the algebraic rate law expressions into xCellerator format. For user coded models, the solution consisted of using the SBML reader that we constructed (discussed in Section 4.2) to generate text file outputs of the SBML models, and manually examining the rate law expressions. MathSBML , a Mathematica extension written by Bruce Shaprio (available at http://sourceforge.net/projects/sbml/files/mathsbml/) , also became available and provided a function to translate SBML to a human-readable form. Models that were composed of readily identifiable rate expressions such as Michaelis Menten, mass action and hill functions were selected for coding using the Sigmoid API classes.

Another fraction of the SBML models contained algebraic expressions that could be passed, with minor syntactic adjustments, directly to Mathematica for processing along with xCellerator code. In order to overcome this challenge, a set of AlgebraicPassthrough classes (see Section 2.9.1.3), a RateFunction parameter (see Section 2.3), and variable declarations field in the RateConstants class (see Section 2.4.1.1) were constructed and added to the schema. The adjustments to the RateConstants class allowed us to make variable declarations for reactants that were expressed in the algebraic expressions. The variable declarations and the algebraic expressions in the AlgebraicPassthrough MathReactions are passed though the middleware as strings to Mathmatica for processing inside the xCellerator notebook. We have several curated models in the database that use these features.

### 4.1.3 ***Sigmoid Team Population Effort.***

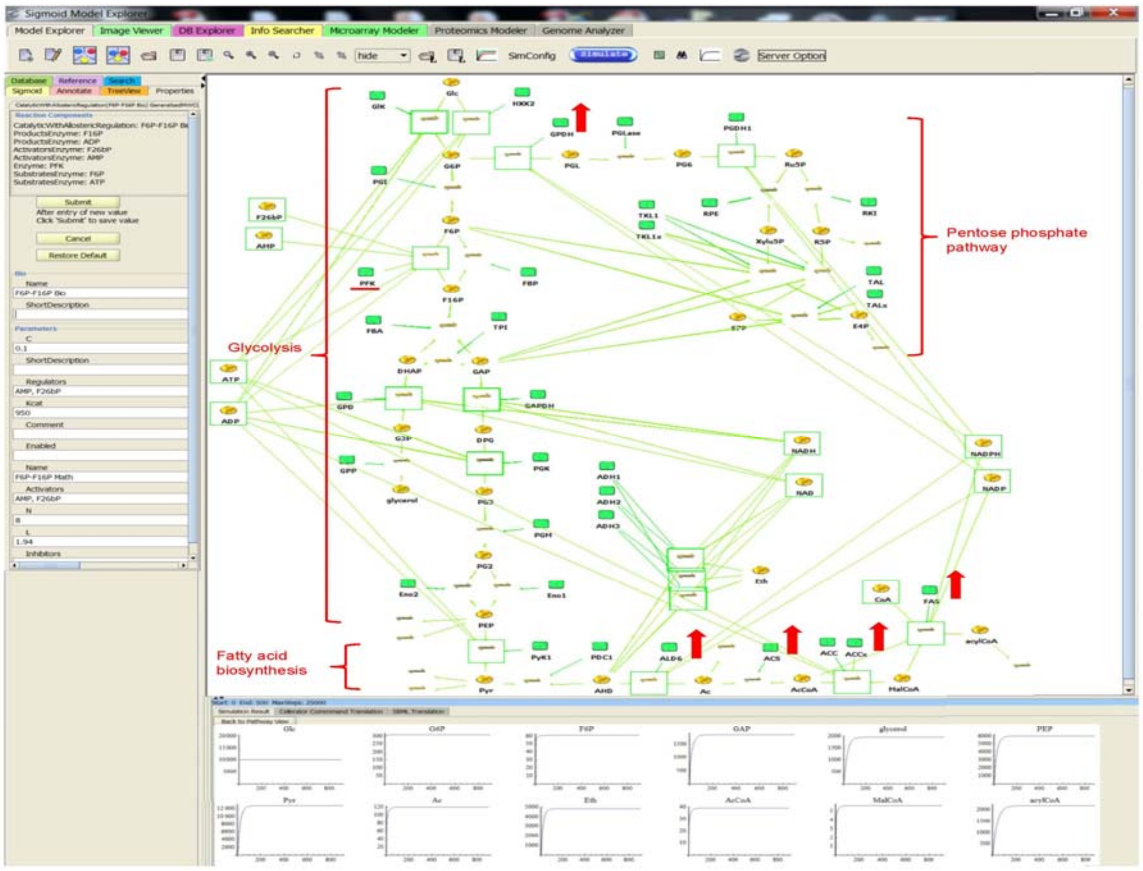
The population of the Sigmoid database with pathway models has been the product of the efforts of many individuals. Since it would be prohibitive to address authorship details throughout the text of this dissertation, a comprehensive table of member contributions with regard to database population has been provided in Appendix B. Sigmoid models derived from the Biomodels.net SBML repository are also indicated in Appendix B.

Large-scale models of the signaling pathways include the mammalian Epidermal Growth Factor Receptor (EGFR) pathway [Kholodenko *et al.*1999] and the yeast pheromone response pathway [Kofahl and Klipp2004], while other models represent common aspects of metabolism that include the anabolic Calvin cycle in plants [Poolman *et al.*2004], two models of branched chain amino acid biosynthesis in bacteria [Najdi *et al.*2006], [Yang *et al.*2005a], and catabolic glycolysis [Nielsen *et al.*1998]. Furthermore, a simple model of the circadian clock [Tyson *et al.*1999] and two models of intracellular calcium flux [Borghans *et al.*1997] demonstrate oscillating outputs. Separate models of the NFkB [Hoffmann *et al.*2002], Calcineurin [Hilioti *et al.*2004] and the p53 [Bullock and Fersht2001] regulatory networks demonstrate how transcription factors and their ability to activate or inhibit gene expression are regulated. Lastly, some models in the database represent diverse processes, including the mechanism of degradation of enzymes during industrial food processing [Brands and van Boekel2002] and the cell fate decisions of protists in the presence of far-red light under starvation conditions [Marwan2003].

Computational models of the mitogen-activated protein kinase (MAPK) cascade are also present in the Sigmoid database. Several models derived from [Markevich *et al.*2004] examine the same MAPK cascade with two separate mechanisms, mass action and Michaelis-Menten, for each of the phosphorylation and dephosphorylation events. For each of these mechanisms, the models increase in complexity as the site and order of phosphorylation are taken into account in the set of reactions. In contrast to these models, Huang_1996_MAPK and its xCellerator notebook “MAPK cascade: Huang and Ferrell 1996”, present the celebrated [1996] model that demonstrates the connection between a nonprocessive, twocollision dual-phosphorylation mechanism of kinase activation and an ultrasensitive, switch-like response. The model Bardwell_2007_MAPK_VariableFeedback and corresponding notebook “MAPK Cascade with Variable Feedback” extend this model to include a simple feedback phosphorylation of an upstream kinase by the MAPK. The effects of the feedback loop on the system depend upon the nature of the feedback: if feedback phosphorylation increases the activity of the upstream kinase (positive feedback), a bistable, all-or-none response may result [Ferrell and Machleder.1998]. In contrast, if feedback phosphorylation decreases the activity of the upstream kinase (negative feedback), then the result may be damped or sustained oscillation of the activity of the kinases in the cascade [Kholodenko2000]. The notebook contains examples of parameter values that will generate either of these outcomes, illustrating how complex, diverse and biologically useful behaviors can emerge from the combination of an ultrasensitive cascade architecture and a simple feedback loop.

### 4.1.4 ***The Najdi_2009_Xyl_Ara_Et hModel***

An unpublished xCellerator notebook model of engineered strains of yeast was constructed by Dr. Tarek Najdi. The modeling goal of the notebook was to perform mathematical modeling of proposed pathways to increase the yields of ethanol (Eth) production from *Saccharomyces cerevisiae.* A great portion of plant carbon sources consisting of the monosaccharide sugars xylulose (Xyl) and arabinose (Ara) go largely unmetabolized by wild type strains of *S. cerevisiae.* The goal of the project was to engineer metabolic pathways for yeast that would convert these sugars to ethanol with high yields, thus increasing overall yields of commercial ethanol production. The fairly large (60 reactions) notebook contained portions of the pentose phosphate pathway, glycolysis, and xylose metabolism. We constructed a Sigmoid coding of the notebook for visualization and simulation within the Sigmoid system. This pathway model required extensions to the kMech reaction equations. The requirement for these extensions also prompted the development of SigMech described in 0.



Reaction Components
CatalyticWithAllostericRegulation: F6P-F16P Bi
ProductsEnzyme: F16P
ProductsEnzyme: ADP
ActivatorsEnzyme: F26bP
ActivatorsEnzyme: AMP
Enzyme: PFK
SubstratesEnzyme: F6P
SubstratesEnzyme: ATP

After entry of new value
Click 'Submit' to save value

Bio
Name: F6P-F16P Bio
ShortDescription:

Parameters
C: 0.1
ShortDescription:
Regulators: AMP, F26bP
Kcat: 950
Comment:
Enabled:
Name: F6P-F16P Math
Activators: AMP, F26bP
N: 8
L: 1.94
Inhibitors

**Figure 4.2 The Najdi_2010_FattyAcidBiosynthesis model contains portions of the pentose phosphate pathway, glycolysis and fatty acid biosynthesis.**

### 4.1.5 *The Najdi_2010_FattyAcidBiosynthesis Model*

A one-quarter cross-training laboratory rotation was spent in Suzanne Sandmeyer's laboratory on a project that was directed toward engineering a transgenic strain of *Saccharomyces cerevisiae* with the ultimate goal of producing high yields of short chain fatty acids for use as platform chemicals for industry. Within this context a pathway optimization was being proposed to generate higher yields of fatty acids. Models were built to better understand these processes. Dr. Tarek Najdi composed Mathematica notebooks to represent the pathway model and proposed optimizations. New kMech reaction equations were developed for the notebook to address the need for enzyme models that have three substrates and three products. We constructed a Sigmoid version of this model (Najdi_2010_FattyAcidBiosynthesis), illustrated in Figure 4.2, and extended the functionality of the Sigmoid system to meet the models' requirements. These extensions consisted of new kMech reaction equations one of which is detailed in Section 3.1.3). The requirement for these extensions also prompted the development of SigMech described in 0. The model contains sections of the pentose phosphate pathway, glycolysis and fatty acid biosynthesis.

ATP-Citrate Lyase (ACL) is present in oleaginous yeasts known to accumulate fatty acids but not in *S. cerevisiae.* The goal of the project was to engineer a strain of *S. cerevisiae* that produces higher levels of cytosolic acetyl-CoA by introducing ACL. Normally, in *S. cerevisiae*, cytosolic acetyl-CoA (the precursor for *de novo* fatty acid synthesis) only comes from pyruvate decarboxylases (PDCs) acting on pyruvate. Pyruvate also goes into the mitochondria and is converted by pyruvate dehydrogenase (PDH) into acetyl-CoA that goes to the TCA cycle. If ACL were to compensate for a PDC knockout, that would prove that ACL can function in *S. cerevisiae*.

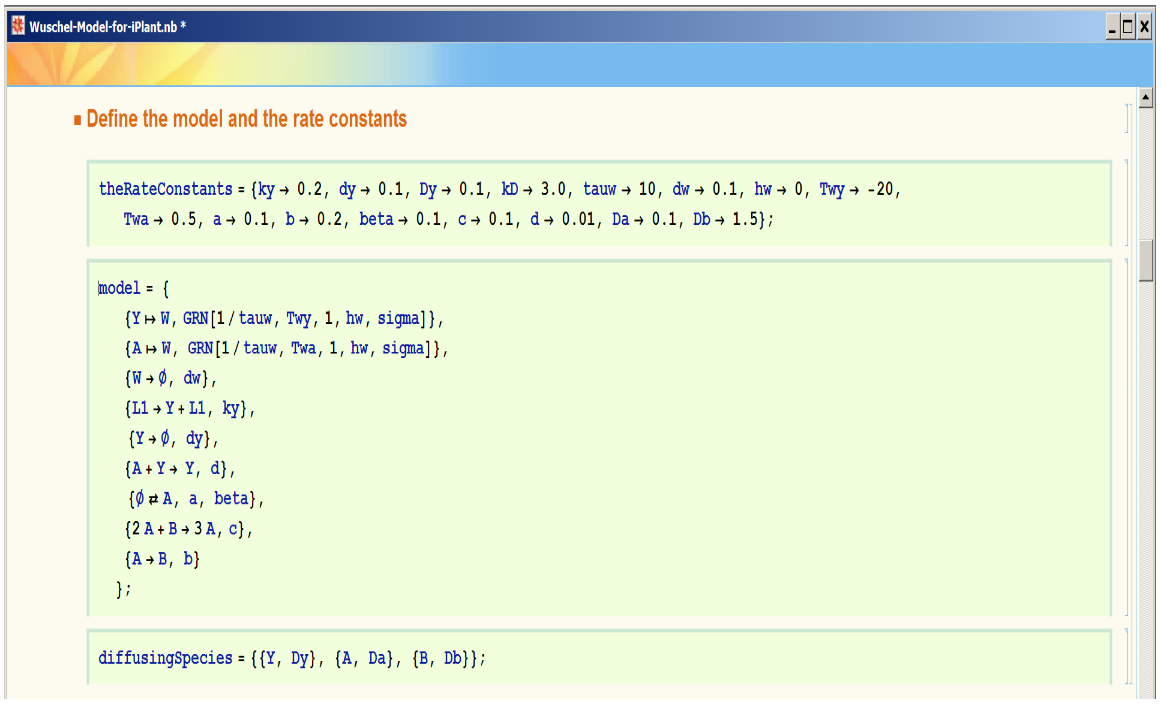
The goal of the portion of the project was to produce strains of yeast with a knockout of the PDC 1,5 and 6 genes. We attempted knocking out the PDC 5 and 6 genes from a previous strain that was PDC 1 deficient. We were not able to generate any viable strains of yeast with more than a PDC 1 deficiency. These findings suggest that knocking out both genes weakened the cell strain past the threshold of viability.

### 4.1.6 *The Middleton_2008_AuxinModel*

Auxin is a plant hormone that plays a regulatory role in developmental processes in *Arabidopsis thaliana.* Auxin signal transduction is an essential pathway modeled in

[Middleton 2010]. As part of an iPlant initiative a Sigmoid model was constructed from the AUXIAA-AM-3 xCellerator notebook for this model, and loaded into the under review section of the public Sigmoid database. The Sigmoid framework was sufficient to properly represent the regulatory network of reactions.

### 4.1.7 *The Wushel Compartmental and SpatialModel Model*


Wuschel-Model-for-iPlant.nb *
Define the model and the rate constants

```
theRateConstants = {ky → 0.2, dy → 0.1, Dy → 0.1, kD → 3.0, tauw → 10, dw → 0.1, hw → 0, Twy → -20,
   Twa → 0.5, a → 0.1, b → 0.2, beta → 0.1, c → 0.1, d → 0.01, Da → 0.1, Db → 1.5};
```

```
model = {
   {Y ↦ W, GRN[1 / tauw, Twy, 1, hw, sigma]},
   {A ↦ W, GRN[1 / tauw, Twa, 1, hw, sigma]},
   {W → ∅, dw},
   {L1 → Y + L1, ky},
    {Y → ∅, dy},
   {A + Y → Y, d},
    {∅ ⇄ A, a, beta},
   {2 A + B → 3 A, c},
   {A → B, b}
  };
```

```
diffusingSpecies = {{Y, Dy}, {A, Da}, {B, Db}};
```

**Figure 4.3 The Wuschel-Model-for-iPlant Cellzilla notebook [Shapiro 2008] implements the Wuschel pattern formation model of [Jönsson 2005]. The model = { …}; section contains the InternalNetwork of reactions and the diffusingSpecies = {…}; section contains the ExternalNetwork of reactions.**

A Sigmoid model named Shapiro_2008_WUS was built using the Wuschel-Model-For-iPlant xCellerator/Cellzilla notebook (see Figure 4.3) implementing the Wuschel pattern formation model of [Jönsson 2005]. This model differs from the other models coded for Sigmoid in that it is a computational model designed for a multicellular developmental system of plant growth in *Arabidopsis thaliana*. This model is the first multicompartmental or multicellular SpatialModel built with the Sigmoid schema classes.

The two-dimensional model uses classes discussed in Section 2.11.2. The model consists of: 1) a set of cell centers modeled as DimensionalCompartments (DC), 2) an InternalNetwork of BioReactions and MathReactions that represents biochemical and regulatory reactions that take place within each separate DC, and 3) an ExternalNetwork of BiReplicated Reactions that represent diffusion reactions between adjacent DimensionalCompartments. Certain Reactants that are involved in the ExternalNetwork are DecoratedReactants (DR). DRs are "decorated" with variables that serve as indices for their corresponding Cellzilla compartment representation. The SpatialModel is a complete Sigmoid representation of the CellzillaNotebook. The ModelEmitter class in the population library of Sigmoid produces console output of the model details. The model currently requires Sigmoid to Cellzilla translation code from the middleware in order to be sent to Cellzilla for simulation. Also, SME will display the model just as for any other Sigmoid model, but the multi-cellular aspects of the model will require additional development in the front end for them to appear and be properly represented.

### 4.1.8 *Zhang Optimizer Models*

Four models were constructed for the Simulated Annealing Optimizer (SAO) that was integrated into Sigmoid from xCellerator notebooks. The SAO is discussed in Section 5.1. The four models that were created by Li Zhang [Zhang2008] consist of:

1) The Zhang_2007_SEP model is a simple enzymatic process model. The model reactions present in the Zhang_2007_SEP model parallel the UniUni kMech reaction for simple catalysis of one substrate (S) by an enzyme (E) into product (P) (SEP). The kMech reaction notation [Yang 2005] is as follows:

$$\mathrm{Enz}[\{\mathrm{S\_}\} \overset{\mathrm{En\_}}{\rightleftarrows} \{\mathrm{P\_}\}, \mathrm{UniUni}[\mathrm{kf\_}, \mathrm{kr\_}, \mathrm{kcat\_}]] := \{\{S + \mathrm{En} \rightleftarrows \mathrm{NC}[S, \mathrm{En}], \mathrm{kf}, \mathrm{kr}\}, \{\mathrm{NC}[S, \mathrm{En}] \rightarrow \mathrm{En} + P, kcat\}\};$$

2) The Zhang_2007_ATM model is a mathematical model for the DNA damage and response signaling transduction pathway focusing on a critical sensor protein called phosphoprotein kinase (Ataxia Telangiectasia-Mutated (ATM)) activation by infrared radiation.

3) The Zhang_2007_ATM_PP2A_MRN is an expanded model of Zhang_2007_ATM that includes an upstream input and positive regulator of ATM, the MRE11-RAD50-NBS1 complex (MRN), and protein phosphatase 2A which is a negative regulator of ATM.

4) Phosphatidylinositol 3 kinase (PI3k) is believed to contribute to cellular transformation and the development of cancer. The Zhang_2007_PI3k model is composed of three Compound enzymatic MathematicalReactions involved in PI3k the membrane-bond network.

### 4.1.9 *Demonstration Models*

Several demonstration and test models were constructed to provide simple examples of basic Sigmoid functions. These models are the MAP-K Demo Reactions, the Algebraic Demo, and the AlgebraicEnz demo. The MAP-K Demo is a simple one-reaction model designed for short demonstrations of Sigmoid. The Algebraic Demo and AlgebraicEnz Demo were designed to demonstrate and test the AlgebraicPassthrough Reactions in Sigmoid and to pass algebraic rate law expressions to Mathematica for processing. AlgebraicPassthrough reactions consist of algebraic rate expressions that do not have corresponding specific xCellerator functions.

## *4.2 Automated Population of the Sigmoid Database*

Since the flexible but comprehensive schema of the Sigmoid database allows us to easily leverage other databases, we have developed “populator” programs which capture community input from external databases and markup languages for entry into Sigmoid.

### 4.2.1 *Kyoto Encyclopedia of Genes and Genomes*

The Kyoto Encyclopedia of Genese and Genomes (KEGG) consists of a growing set of databases. The KEGG database is located at http://www.kegg.jp/kegg/ and has been in development since 1995 by Kanehisa Laboratories. The KEGG database serves as a knowledge base for integration and interpretation of large-scale molecular data sets generated by genome sequencing and other high-throughput experimental technologies. One of their main systems biology databases, in particular the KEGG PATHWAY database, is a collection of pathway maps focusing on molecular interaction and reaction networks [Ogata 1999] [Kanehisa 2006]. These pathway maps are graphical diagrams of biochemical and regulatory pathways, cellular processes, pathway systems on the scale of organisms, genetic information processing, and signal transduction. The pathway maps contain reaction equations that indicate participating molecules, cofactors, and enzymes. Molecular information consisting of naming and synonyms, reaction references, participating pathways, associated enzymes, molecular formulas, molecular structure, and links to external molecular databases are referenced by the KEGG pathway maps as well.

The KEGG PATHWAY maps are stored in the exchange format KGML files and are publicly available. KGML is an extension to the XML format. SBML translations of KGML have been provided at the KEGG website in the past and can also be produced with one of the KEGG to SBML converters available. One of the more recent converters is KEGGConverter [Moutselos *et al.*2009].

The KEGG pathway maps are particularly suitable for Sigmoid integration because the mapping of reactant and reaction data between the KGML format and Sigmoid schema is fairly straightforward.

Early on, we were able to develop an XML parser that translated KEGG biological entities such as genes, proteins, small molecules and reactions into Sigmoid Reactants and Reactions. It became apparent that SBML was becoming a prominent systems biology interchange format and our efforts shifted to developing an SBML translator for Sigmoid. Some effort was expended on developing our own SBML/XML parser and then libSBML became available. LibSBML is an Application Programming Interface (API) for reading, writing and manipulating SBML files and data streams [Bornstein *et al.* 2008]. To avoid duplication of effort, we used libSBML to build an SBML-to-Sigmoid translation program. This translation program provided a mechanism that could be used to automate the population of the Sigmoid database with the KEGG pathway models. We were able to successfully populate the Sigmoid database with hundreds of KEGG pathway models. These models were composed of networks of Sigmoid Reactants and BiologicalReactions. Since KEGG pathway models lack kinetic parameter and rate laws they are excellent candidates to serve as “template pathways” for users in Sigmoid. Users could, with a properly functioning pathway editor in the GUI, add MathematicalReactions and corresponding rate parameters to the BiologicalReaction network to produce simulateable Models. Sabio-RK, discussed in Section 4.2.5, may be a good source for rate expressions and parameters.

### 4.2.2 ***SBML***

In Section 4.1.2 we discussed constructing user coded models from preexisting SBML models. Since the number of available models in the SBML format continues to grow and SBML is supported by a variety of systems biology applications, developing a fully capable automated reader for SBML would be advantageous.

We discussed in the last section that we have an SBML reader that can read the Biologcal reaction network from an SBML file. An important challenge that we've identified, discussed in Section 4.1.2, is that SBML stores reaction kinetics as algebraic rate law expressions that differ from the xCellerator based parameterized reactions that Sigmoid implements. Automated conversion of these algebraic expressions into a Sigmoid / xCellerator form will require several things. A parser must be constructed to identify the symbolic representations of reactants in the rate expressions and to extract the mathematical operators of the expressions. An algebraic expression can have many equivalent forms. Identifying a particular function (a Michaelis-Menten reaction equation for example) in an algebraic expression, when the form of the expression is a variable poses an interesting challenge in pattern recognition and symbolic manipulation.

An alternative approach would be to construct a parser that makes extensive use of the AlgebraicPassthrough reactions and avoid implementing kinetics as xCellerator functions. There are other SBML features that will require implementation and translation to Sigmoid model format such as SBML events, functions, and the SBML representation of compartments in order to fully support SBML reading into Sigmoid.

### 4.2.3 ***Gene Networks***

Original software for the formalized modeling of various levels of genetic systems was developed by the Institute for Cytology and Genetics (ICG) of the Siberian branch of the Russian Academy of Sciences in Novosibirsk. A language, named SiBML, for specification of these models was developed. SiBML is oriented to the construction of mathematical models of molecular genetic systems. Another ICG technology, GeneNet (Ananko *et al.* 2002, 2005), enables the accumulation of information on gene and metabolic networks. A software system was developed by an ICG / UC Irvine team to translate the GeneNet data to the Sigmoid schema. GeneNet data is stored in format based on XML. Each XML diagram file consists of a Header and list of Nodes. The Header possesses GeneNet identifier information, dates of creation and modification, and a description of the biological functions of the gene network. The Nodes consist of genes, proteins, substance, RNAs, reactions and regulatory events.

A collaborative effort facilitated loading the GeneNet data into the Sigmoid database [Podkolodny *et al.* 2006]. The Sigmoid schema representation was sufficient to allow a direct mapping of the GeneNet genes, proteins, supplementary information, comments, and substances (molecules). GeneNet and Sigmoid possess different perspectives on how gene regulation should be modeled. In GeneNet the regulatory events could be organized in cascades, meaning a regulatory event could regulate another regulatory event. In Sigmoid, regulation is modeled so that a regulatory Reactant participates as modifier of a reaction and is not consumed. The GeneNet reaction and regulatory events were mapped to Sigmoid as activators, inhibitors, enzymes and regulators of reactions.

### 4.2.4 ***Gene Ontologies***

Without much effort we were able to populate Sigmoid with the yeast GOnet database [Irwin *et al.*2005], which contains information about yeast ORFs and their annotations, gene ontology (GO), and protein-protein interactions. The population was accomplished by a Java code translation program. The GOnet database itself is periodically updated and integrates information from three different sources: (1) ORFs (description, mutant phenotype, gene product, etc.) from the Saccharomyces Genome DataBase (SGD); (2) GO term annotation from the Gene Ontology Consortium arranged in the three categories of Molecular Function, Biological Process, and Cellular Component; and (3) genetic and physical interactions information from the General Repository for Interaction Datasets (GRID) [Compani, Su *et al.* 2010].

Also, another team member (Trent Su) was able to establish a web service connection to this database so that GO data could be browsed and then instantiated as Sigmoid GO Reactants in SME. Establishing web service interfaces to other databases provides a valuable mechanism to incorporate data into the Sigmoid framework. One advantage of using the web service interface is that the connection provides users with access to current versions of data from specialized external databases. Data acquisition can be done on-the-fly. Thus, if the data set is rapidly changing, users will always be able to access the most recent changes. Unfortunately, relying on the external database for data can present a drawback if the external database or database web service format is unstable or discontinued. This was the case with the GOnet database, as the database has disappeared intermittently because of personnel or administrative changes.

### 4.2.5 ***SabioR-K***

The Sabio Reaction Kinetics database (Sabio-RK) is a database for storing kinetic properties for biochemical reactions. The database is located at http://sabiork.h-its.org. Kinetic parameters and rate equations for biochemical reactions are stored in their relational database along with the experimental conditions with which the parameters were measured. The Sabio-RK system provides SBML exports of their data and a set of web services for data acquisition. One of their stated missions is to provide support for the development of biochemical network models [Wittig 2006, 2011]. Sabio-RK data could be a key component for expanded Sigmoid models derived from KEGG pathway models, as the majority of their reaction pathway networks are extracted from KGML datasets. Their kinetics data is compiled and curated manually from literature. Since KEGG data is easily converted into Sigmoid BiologicalReaction networks (see Section 4.2.1), a corresponding MathematicalReaction network could be created with curated kinetic data and rate expressions from Sabio-RK.

Web services were established by another team member (Trent Su) in spring 2009. It was possible to browse their rate expressions and kinetic parameters from within SME. The Sabio-RK schema and web services were rapidly evolving at the time, so permanent web service connections will require more development and maintenance. Since the Sabio-RK interface now generates SBML, a SBML translator for Sigmoid may provide more stable access to Sabio kinetic data. An alternate strategy (if Sabio-RK web services have not stabilized) would be to restore the web service interface and to extract all Sabio-RK data at one time or periodically, for storage within Sigmoid for investigation.

# Chapter 5. Support for Exterior Simulators and Utilities

## 5.1 Parameter Optimization

A Simulated Annealing Optimizer [Zhang2008] has been integrated into Sigmoid through the web services interface. It uses a global optimization technique and Lam-Delosme schedule to make the optimization process faster and more efficient when compared with other general schedules available [Lam and Delosme1988]. It aims to reverse engineer model parameters (for example: kinetic rate constants) given both the model structure (represented as ordinary differential equations) and empirical system dynamics as expressed by time series experimental data [Compani, Su *et al.* 2010].

## 5.2 Alternate Simulators

The modularity of Sigmoid along with its separation of biological and mathematical representations enables us to build interfaces to additional computer algebra systems outside of the Mathematica/Cellerator superstructure. SAGE [Stein 2012)] for instance, an open source mathematics software program largely constructed upon the Python framework provides a gateway to a broad array of open source math programs such as Axiom, GAP, GP/PARI, Macaulay2, Maxima, Octave, and Singular. In addition, the SAGE language includes interfaces to commercial mathematics programs like Magma, Maple, Mathematica, MATLAB, and MuPAD as well. Constructing a language interface to SAGE or some similar project would enable Sigmoid to harness the additional functional functionality provided by these packages

## *5.3 Graph Crunch*

Sigmoid has already incorporated into its schema a representation for Library of Efficient Data types and Algorithms (LEDA) formatted graphs (see Section 2.4.2). LEDA supplies implementations for common algorithms used in graph theory and computational geometry [Mehlhorn 1999]. GraphCrunch is an open source software utility that uses LEDA to perform graph analysis on biological networks and is a likely candidate for integration with Sigmoid. GraphCrunch would enable comparison of Sigmoid BiologicalReaction networks with sets of user specified random graph models. GraphCrunch supports five different types of random graphs. Furthermore, GraphCrunch generates statistics of network property similarities between data and model networks [Milenković 2008].

# Chapter 6. Conclusion

We have successfully extended the capabilities of Sigmoid and improved the object oriented representation for the modeling, storage and simulation of biological components and processes. Parameter sets and parameters are now structured in an improved class structure where reaction parameters are stored by reference within Sigmoid, MathReactions and Sigmoid Models can use multiple sets of parameters. Previously parameters were stored as symbol-value pairs that were more closely tied to the Cellerator notebook structure.

The biological Reactant hierarchy has been expanded to incorporate additional relevant classes of biological entities. These classes span several scales of size and complexity. Sigmoid reactant class groups listed in increasing complexity are Particles, SmallMolecules, MacroMolecules, Complexes, and Structured Reactants of even greater scales such as, organelles, cells, colonies, and organisms.

The Reaction hierarchy has been reorganized and expanded to separate biological classification from mathematical implementation. Groupings of simple, catalytic, and CompoundBio (CompoundBio reactions are a subnetwork of biological reactions) reactions provide the flexibility that biologists can use to model biological reaction networks and regulatory processes. The representation of kinetics is now separated from BioReactions and exists within the MathematicalReaction subhierarchy of Reactions. Currently Sigmoid supports a great portion of xCellerator Mathematica functions and a large library of kMech enzyme mechanisms.

Support for GeneOntoloties has been successfully integrated into the Sigmoid schema allowing us to populate the database with the yeast GOnet database [Irwin *et al.*2005], which contains information about yeast ORFs and their annotations, gene ontology (GO), and protein-protein interactions. In addition another team member was able to utilize the schema support to add a web service interface to the GOnet database thereby allowing for user construction of models incorporating GO components into Sigmoid Models.

We have designed a new utility SigMech and its corresponding KMechReaction that more efficiently represents enzymatic processes. The KMechReaction and associated classes can represent all the previously existing kMech enzyme reactions in a parameterized form. This parameterized representation can subsequently be processed by SigMech to generate a subreaction network of bidirectional mass action reactions that constitute the enzyme mechanism. Additionally, SigMech processes enzyme mechanism descriptions, entered by a user into a GUI, which can be accessed from SME. SigMech processes the descriptions into sets of corresponding subreactions that constitute the enzyme mechanism. SigMech combines the reaction motifs of previous kMech reaction mechanisms with a flexible conversion phase that was implemented in the three-stage xCellerator reaction (ThreeStageCatalytic in the Sigmoid schema) yielding a greater parameter space of possible enzymatic reaction mechanisms that can be generated on the fly inside SME, and that can consequently be simulated by Mathematica/ xCellerator.

We've included classes for Library of Efficient Data types and Algorithms (LEDA) formatted graphs. The LEDA graph representation can provide a framework to assist with graph analysis of biological reaction networks stored within the Sigmoid database.

A likely graph analysis candidate for integration with Sigmoid is GraphCrunch. GraphCrunch would enable comparison of Sigmoid BiologicalReaction networks with sets of user specified random graph models.

We now have a spatial and compartmental modeling representation in Sigmoid. Although we intend our representation to be more comprehensive than simply supporting Cellzilla, Cellzilla support is a valuable first step as the simulation platform is already part of xCellerator. Our SpatialModel classes were flexible enough to construct a multicompartmental developmental Sigmoid SpatialModel that implements the Wuschel pattern formation model of [Jönsson 2005], as part of an "iPlant" exploratory project.

A Simulated Annealing Optimizer (SAO) [Zhang2008] has been integrated into Sigmoid that uses a global optimization technique and Lam-Delosme schedule [Lam and Delosme1988]. Integration consists of web services and a set of SAO parameter sets in the schema. Four SAO integrated Models have been constructed: one simple enzymatic reaction model and three other models focusing on DNA damage and response of a critical sensor protein called phosphoprotein kinase (Ataxia Telangiectasia-Mutated (ATM)) activation by infrared radiation.

The classes that constitute the schema were sufficient to allow us to populate the Sigmoid database with over twenty published models, the majority of which focus on virtual representation of intracellular pathways that include examples in signaling, metabolism, the cell cycle, and gene regulation. Some of the models, such as the Bardwell_2007_MAPK_VariableFeedback model, exhibit varying behaviors when switching between different parameter sets.

We’ve developed automated database population programs that can convert pathway models derived from the Kyoto Enzyclopedia of Genes and Genomes (KEGG) into Sigmoid models. These programs initially translated data from the KEGG format KGML to Sigmoid. Further integration of KEGG pathways and SABIO-RK derived kinetics may constitute an abundant source of pathway models for Sigmoid.

The process of developing an SBML-to-Sigmoid translator revealed an interesting challenge. SBML reaction kinetics are expressed in the form of algebraic expressions, whereas xCellerator generates sets of ordinary differential equations from an arrow input notation. Translating common reaction kinetics written as algebraic expressions, that may be present in variable but equivalent forms, into specific xCellerator functions is a problem in symbolic pattern recognition and manipulation that will require some investigation.

## Bibliography:

[Ananko 2002]Ananko E.A. *et al*. (2002) GeneNet: a database on structure and functional organisation of genenetworks. *Nucl. Acids Res*., **30**, 398–401.

[Ananko 2005]Ananko E.A. *et al*. (2005) GeneNet in 2005. *Nucl. Acids Res*., **33**, Database issue D425–D427.

[Ashburner *et al.* 2000] Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G The Gene Ontology Consortium "Gene Ontology: tool for the unification of biology". Nature Genetics 25 (1): 25–29. (2000).

[Borghans *et al.*1997] J. M. Borghans, G. Dupont, and A. Goldbeter. Complex intracellular calcium oscillations. a theoretical exploration of possible mechanisms. *Biophys Chem.*, 66(1):25–41, 1997.

[Bornstein *et al.* 2008] Bornstein, B. J., Keating, S. M., Jouraku, A., and Hucka M. (2008) LibSBML: An API Library for SBML. *Bioinformatics*, 24(6):880–881, doi:10.1093/bioinformatics/btn051.

[Brands and van Boekel2002] C. M. Brands and M. A. van Boekel. Kinetic modeling of reactions in heated monosaccharide-casein systems. *J Agric Food Chem.*, 50(23):6725–39, 2002.

[Bullock and Fersht2001] A. N. Bullock and A. R. Fersht. Rescuing the function of mutant p53. *Nat Rev Cancer*, 1(1):68–76, 2001.

[Chang 2011] PhD. Dissertation, Integrative Modeling of Mitochondrial Bioenergetics and Application to Clinical Diagnostics of Mitochondrial Disorders Chang, Ivan; Baldi, Pierre F.; Mjolsness, Eric; Lee, Abraham P.. University of California, Irvine, 2011. 2011. 3491269.

[Cheng *et al.*2005] J. Cheng, L. Scharenbroich, P. Baldi, and E. Mjolsness. Sigmoid: Towards a generative, scalable software infrastructure for pathway bioinformatics and systems biology. *IEEE Intelligent Systems*, 20(3):68–75, 2005.

[Compani, Su *et al.* 2010] B. Compani, T. Su, I. Chang, J. Cheng, K. H. Shah, T. Whisenant, Y. Dou, A. Bergmann, R. Cheong, B. Wold, L. Bardwell, A. Levchenko, P. Baldi, and E. Mjolsness, A Scalable and Integrative System for Pathway Bioinformatics and Systems Biology. Adv Exp Med Biol. 2010; 680: 523–534.

[Ferrell and Machleder.1998] J. E. Ferrell and E. M. Machleder. The biochemical basis of an all-or-none cell fate switch in xenopus oocytes. *Science*, 280:895–898, 1998.

[Finney and Hucka 2003] Finney, A., and Hucka, M. (2003). Systems Biology Markup Language: Level 2 and Beyond. *Biochemical Society Transactions*, vol. 31, part 6.

[Hilioti *et al.*2004] Z. Hilioti, D. A. Gallagher, S. T. Low-Nam, P. Ramaswamy, P. Gajer, T. J. Kingsbury, C. J. Birchwood, A. Levchenko, and K. W. Cunningham. Gsk-3 kinases enhance calcineurin signaling by phosphorylation of rcns. *Genes Dev.*, 18(1):35–47, 2004.

[Hoffmann *et al.*2002] A. Hoffmann, A. Levchenko, M. L. Scott, and D. Baltimore. The ikappab-nf-kappab signaling module: temporal control and selective gene activation. *Science*, 298(5596):1241–5, 2002.

[Hucka *et al.* 2003] Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A. , Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofmeyr, J.-H., Hunter, P. J., Juty, N. S., Kasberger, J. L., Kremling, A., Kummer, U., Le Novère, N., Loew, L. M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E. D., Nakayama, Y., Nelson, M. R., Nielsen, P. F., Sakurada, T., Schaff, J. C., Shapiro, B. E., Shimizu, T. S., Spence, H. D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., Wang, J. (2003). The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, vol. 19, no. 4, pp. 524–531.

[Irwin *et al.*2005] B. Irwin, M. Aye, P. Baldi, N. Beliakova-Bethell, H. Cheng, Y. Dou, W. Liou, and S. Sandmeyer. Retroviruses and yeast retrotransposons use overlapping sets of host genes. *Genome Research*, 15:641–654, 2005.

[Jönsson 2005] Jönsson H, Heisler M, Reddy GV, Agrawal V, Gor V, Shapiro BE, Mjolsness E, Meyerowitz EM (2005) "Modeling the organization of the Wuschel domain in the shoot apical meristem," Bioinformatics 21(S1): i232-i240

[Kanehisa 2006] Kanehisa M, Goto S, Hattori M, Aoki-Kinoshita KF, Itoh M, Kawashima S *et al.* (2006). "From genomics to chemical genomics: new developments in KEGG.". Nucleic Acids Res 34 (Database issue): D354-7.

[Kholodenko *et al.*1999] B. N. Kholodenko, O. V. Demin, G. Moehren, and J. B. Hoek. Quantification of short term signaling by the epidermal growth factor receptor. *J Biol Chem.*, 274(42):30169–81, 1999.

[Kholodenko2000] B. N. Kholodenko. Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. *Eur J Biochem*, 267:1583–1588, 2000.

[Kofahl and Klipp2004] B. Kofahl and E. Klipp. Modelling the dynamics of the yeast pheromone pathway. *Yeast.*, 21(10):831–50, 2004.

[Lam and Delosme1988] J. Lam and J. Delosme. Performance of a new annealing schedule. Pages 306–311. 1988.

[Markevich *et al.*2004] N. I. Markevich, J. B. Hoek, and B. N. Kholodenko BN. Signaling switches and bistability arising from multisite phosphorylation in protein kinase cascades. *J Cell Biol*, 164(3):353–9, 2004.

[Marwan2003] W. Marwan. Theory of time-resolved somatic complementation and its use to explore the sporulation control network in physarum polycephalum. *Genetics*, 164(1):105–15, 2003.

[Mehlhorn 1999] Mehlhorn, K., Näher S. LEDA: A Platform for Combinatorial and Geometric Computing, Cambridge University Press 1999 ISBN 0-521-56329-1.

[Middleton 2010] Middleton AM, King JR, Bennett MJ, Owen MR. Mathematical modelling of the Aux/IAA negative feedback loop. Bull Math Biol. 2010 Aug;72(6):1383-407.

[Milenković 2008] Tijana Milenković, Jason Lai, and Nataša Pržulj, GraphCrunch: a tool for large network analyses, BMC Bioinformatics 2008, 9:70.

[Mjolsness 2007] Towards a Calculus of Biomolecular Complexes at Equilibrium. Eric Mjolsness, Briefings in Bioinformatics, 8(4):226-33 July 2007.

[Moutselos *et al.* 2009] K. Moutselos, I. Kanaris, A. Chatziioannou, I. Maglogiannis,and F. Kolisis KEGGconverter: a tool for the in-silico modelling of metabolic networks of the KEGG Pathways database BMC Bioinformatics. 2009; 10: 324.

[Najdi *et al.*2005] T. S. Najdi, C. R. Yang, B. E. Shapiro, G.Wesley Hatfield, and E. D. Mjolsness. The generalized Monod, Wyman, Changeux model for mathematical modeling of metabolic enzymes with allosteric regulation. In *Proc. IEEE Computational Systems Bioinformatics Conference*, Stanford University, CA, 2005.

[Najdi *et al.*2006] T. S. Najdi, C. R. Yang, B. E. Shapiro, G. W. Hatfield, and E. D. Mjolsness. Application of a generalized MWC model for the mathematical simulation of metabolic pathways regulated by allosteric enzymes. *J Bioinform Comput Biol.*, 4(2):335–55, 2006.

[Najdi 2010]Najdi TS, Hatfield GW, Mjolsness ED. A 'random steady-state' model for the pyruvate dehydrogenase and alpha-ketoglutarate dehydrogenase enzyme complexes. Phys Biol. 2010; 7:16016.

[Najdi 2010]  TerTer Random Addition kMech Reaction, via private communication Tarik Najdi. June, 2010.

[Nielsen *et al.*1998] K. Nielsen, P. G. Sarensen, F. Hynne, and H. G. Busse. Sustained oscillations in glycolysis: an experimental and theoretical study of chaotic and complex periodic behavior and of quenching of simple oscillations. *Biophys Chem.*, 72(1-2):49–62, 1998.

[Ogata 1999] H Ogata, S Goto, K Sato, W Fujibuchi, H Bono, and M Kanehisa KEGG: Kyoto Encyclopedia of Genes and Genomes. Nucleic Acids Res. 1999 January 1; 27(1): 29–34

[Oracle 2012] The Java database connectivity API.
URL: http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136101.html. Last accessed on March 8, 2012

[Podkolodny *et al.* 2006] Podkolodny NL, Podkolodnaya NN, Miginsky DS, Poplavsky AS, Likhoshvai VA, Compani B, Mjolsness E. An integration of the descriptions of gene networks and their models presented in Sigmoid (Cellerator) and GeneNet, 5th International Conference on the Bioinformatics of Genome Regulation and Function (BGRS-2006), Volume 3, pp. 86-90.

[Poolman *et al.*2004] M. G. Poolman, H. E. Assmus, and D. A. Fell. Applications of metabolic modelling to plant metabolism. *J Exp Bot.*, 55(400):1177–86, 2004.

[Savageau  1969] Savageau M.A. Biochemical systems analysis. II. The steady-state solutions for an n-pool system using a power-law approximation. J Theor Biol 1969 25:370-379.

[Savageau  1970] Savageau M.A. Biochemical systems analysis. 3. Dynamic solutions using a power-law approximation. J Theor Biol 1970 26:215-226.

[Segel 1992] I. H. Segel. *Enzyme Kinetics. Behavior and Analysis of Rapid Equilibrium and Steady State Enzyme Systems*. Wiley, New York, NY, 1992.

[Shapiro *et al.*2003] B. E. Shapiro, A. Levchenko, E. M. Meyerowitz, B. J.Wold, and E. D. Mjolsness. Cellerator: Extending a computer algebra system to include biochemical arrows for signal transduction simulations. *Bioinformatics*, 19(5):677–678, 2003.

[Shapiro 2007] M. Hucka E. Mjolsness B. Shapiro, J. Lu. Mathematica platforms for modeling in systems biology: Recent developments in mathsbml and cellerator. 2007.

[Stein 2012] W. Stein. *SAGE: Software for Algebra and Geometry Experimentation*. http://www.sagemath.org/ and http://sage.scipy.org/. Last accessed March 2012.

[Su 2004] T. Su, *Web-based visualization and manipulation tool for systems biology models* Dissertation Thesis (M.S., Information and Computer Science)--University of California, Irvine, LD 791.8 .I5 2004 S8, 2004.

[Tyson *et al.*1999] J. J. Tyson, C. I. Hong, C. D. Thron, and B. Novak. A simple model of circadian rhythms based on dimerization and proteolysis of per and tim. *Biophys J.*, 77(5):2411–7, 1999.

[Wittig *et al.* 2006] Wittig U.,Golebiewski, M., Kania, R., Krebs, O., Mir, S., Weidemann, A., Anstein, S., Saric, J. and Rojas, I., SABIO-RK: Integration and Curation of Reaction Kinetics Data Wittig U. In proceedings of the 3rd International workshop on Data Integration in the Life Sciences 2006 (DILS'06). Hinxton, UK. Lecture Notes in Bioinformatics, 4075: 94-103(2006).

[Wittig *et al.* 2011] Wittig, U.; Kania, R.; Golebiewski, M.; Rey, M.; Shi, L.; Jong, L.; Algaa, E.; Weidemann, A. et al (2011). "SABIO-RK--database for biochemical reaction kinetics". Nucleic Acids Research 40 (Database issue): D790–6.

[Yang *et al.*2005a] C. R. Yang, B. E. Shapiro, S. P. Hung, E. D. Mjolsness, and G. W. Hatfield. A mathematical model for the branched chain amino acid biosynthetic pathways of Escherichia coli k12. *J Biol Chem.*, 280(12):11224–32, 2005.

[Yang *et al.*2005b] C. R. Yang, B. E. Shapiro, E. D. Mjolsness, and G. W. Hatfield. An enzyme mechanism language for the mathematical modeling of metabolic pathways. *Bioinformatics*, 21:774–780, 2005.

[Zhang2008] L. Zhang. *Dynamic Biological Signaling Pathway Modeling and Parameter Estimation Through Optimization*. PhD thesis, Information and Computer Science: University of California, Irvine, 2008. LD 791.9 I5 2008 Z43, OCLC:276454918.

# Appendix:

## A. Sigmoid Class Authorship

**Key:**

CA: B. Compani is the author.

CAE: B. Compani has edited the class attributes.

CR: B. Compani has revised this class.

PA: Previous authors include Lucas Scharenbroich and Jainlin Cheng.

USE: A U indicates the class has experienced some use, either in a model, or by Sigmoid related code such as SME, a populator program or SigMech. Entries in this column are a rough estimate from my recollection of their use in Sigmoid.

X: Indicates authorship. (or use in the Use column)

m: Indicates a change in the class due to a change in class inheritance.

| Sigmoid Class | CSA | CAE | CR | PA | USE |
|---|---|---|---|---|---|
| AcidBase | X | | | | |
| AcidsAmines | | | | | |
| ActivationPattern | | X | | X | |
| Acylation | X | | | | |
| AdjacencyMatrix | X | | | | |
| AffinityColumnMassSpect | | | | X | |
| AffinityDerivedComplex | | | X | X | |
| AlgebraicPassthrough | X | | | | U |
| Allosteric | | X | X | X | |
| AllostericInteraction | X | | | | |
| Alpha | X | | | | |
| Amines | X | | | | |
| AminoAcid | X | | | | |
| AminoAcidSequence | | | X | X | |
| Ampere | X | | | | |
| Antibody | X | | | | |
| Article | | m | m | X | U |
| AssemblyDisassembly | X | | | | |
| Author | X | | | | U |
| AutoCatalysis | X | | | | U |
| BiBi | | X | X | X | U |
| BiBiCompetitive | X | | | | U |
| BidirectionalCatalyticAlgebraicPassthrough | X | | | | U |
| BidirectionalMassAction | X | | | | U |

| | | | | | |
|---|---|---|---|---|---|
| BindingPair | X | | | | U |
| BioAnnihilation | X | | | | U |
| BioCatalyticAnnihilation | X | | | | U |
| BioCatalyticCreation | X | | | | U |
| BioComplex | | X | X | X | U |
| BioCreation | X | | | | |
| BiologicalProcess | | | | X | |
| BiologicalReaction | | X | X | X | U |
| BioRegulatoryRelationship | X | | | | U |
| Biotinylation | X | | | | |
| BireplicatedReaction | X | | | | U |
| BirthProcess | | | | X | |
| BiTer | X | | | | |
| BIUni | X | | | | U |
| BiUniCompetitive | X | | | | U |
| BiUniCompetitiveNonCompetitive | X | | | | |
| Book | | m | m | X | |
| Booklet | | m | m | X | |
| BooleanOverReactantState | | | | X | |
| BooleanParameter | X | | | | U |
| Candela | X | | | | |
| Carbohydrate | X | | | | U |
| Carbohydrates | X | | | | |
| Carboxylation | X | | | | |
| Cartesian2D | X | | | | U |
| Cartesian3D | X | | | | |
| Catalytic | | X | X | X | U |
| CatalyticAlgebraicPassthrough | X | | | | U |
| CatalyticEnzymatic | X | | | | |
| CatalyticViaHill | X | | | | U |
| CatalyticViaMichaelisMenten | X | | | | U |
| CatalyticWithAllostericRegulation | X | | | | U |
| CatalyzedWithInhibitorsAndOrActivators | X | | | | U |
| Cell | X | | | | |
| CelleratorModel | | | X | X | |
| CelleratorParameterSet | X | | | | U |
| CellMembrane | X | | | | |
| Centriole | X | | | | |
| CGIKnowledgeSource | | X | X | X | |
| Channel | X | | | | |
| Chloroplast | X | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Citation | | m | X | X | U |
| Class_3 | | | | | |
| CoEnzyme | X | | | | U |
| Colony | X | | | | |
| Combination | X | | | | |
| Compartment | X | X | | X | |
| CompartmentRelationshipPair | X | | | | U |
| Composition | X | | | | |
| CompoundBio | X | | | | |
| CompoundReaction | | X | X | X | U |
| Conference | | m | m | X | |
| ConformationalIsomerization | X | | | | |
| ConstantNode | | | | X | |
| CoordinateBin | | | | | |
| Coordinates | X | | | | U |
| CRUserDefinedModule | X | | | | |
| CTerminalAmidation | X | | | | |
| CuboidMatrix | X | | | | |
| Culture | X | | | | |
| Cysteinylation | X | | | | |
| Cytosol | X | | | | |
| Database | | | X | X | |
| DataElement | X | | | | |
| DataElementIDReference | X | | | | |
| Dataset | | | | X | |
| Deamidation | X | | | | |
| DeathProcess | | | | X | |
| DecomposedTranscription | | | X | X | |
| Decomposition | X | | | | |
| DecoratedReactant | X | | | | U |
| DecorationTypePair | X | | | | |
| DecorativeActivation | | | | X | |
| DecorativeActivationModification | X | | | | |
| DecorativeActivationModifications | | | | X | |
| Deoxyhexoses | X | | | | |
| DerivedUnit | X | | | | |
| DimensionalCompartment | X | | | | U |
| DisulphideBond | X | | | | |
| DNA | | | X | X | U |
| DNADNA | X | | | | |
| DNAProtein | X | | | | |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| DNARNAProtein | X |  |  |  |  |
| DocumentedProtein |  |  | X | X |  |
| DoubleElement |  |  |  | X | U |
| DoubleParameter | X |  |  |  | U |
| DoubleReplacement | X |  |  |  | U |
| DoubleVector | X |  |  |  | U |
| Edge | X |  |  |  | U |
| EdgeList | X |  |  |  | U |
| Electron | X |  |  |  |  |
| EndoplasmicReticulum | X |  |  |  |  |
| EnzymaticReaction |  | X | X | X | U |
| Enzyme | X |  |  |  | U |
| EnzymeSubstrate | X |  |  |  | U |
| Events | X |  |  |  |  |
| ExclusiveGroup | X |  |  |  | U |
| ExperimentalDataset | X |  |  |  |  |
| ExternalNetwork | X |  |  |  | U |
| Farnesylation | X |  |  |  |  |
| FattyAcid | X |  |  |  |  |
| Formylation | X |  |  |  |  |
| Function |  | X |  | X |  |
| FunctionDefinition | X |  |  |  |  |
| FunctionNode |  |  |  | X |  |
| Gate | X |  |  |  |  |
| Gene |  |  | X | X |  |
| GeneOntologyComponent | X |  |  |  | U |
| GeneOntologyComponents | X |  |  |  | U |
| GeneOntologyFunction | X |  |  |  | U |
| GeneOntologyProcess | X |  |  |  | U |
| GeneOntologyProcesses | X |  |  |  | U |
| GeneOntologySource | X |  |  |  | U |
| GeneralCatalyzedReaction | X |  |  |  | U |
| GeneralConversions | X |  |  |  | U |
| GeneralizedMWC |  | X | X | X | U |
| GeneralizedMWCCompetitive | X |  |  |  | U |
| GenericComplex | X |  |  |  |  |
| Geranylgeranylation | X |  |  |  |  |
| GlobalParameters | X |  |  |  | U |
| Glutathionylation | X |  |  |  |  |
| Glycosylation | X |  |  |  |  |
| GOAnnotation | X |  |  |  | U |

| | | | | | |
|---|---|---|---|---|---|
| GolgiApparatus | X | | | | |
| Graph | X | | | | |
| GRN | X | | | | |
| Hexosamines | X | | | | |
| Hexoses | X | | | | |
| Hill | | X | X | X | U |
| Hormone | X | | | | |
| Hydrolase | X | | | | |
| Hydroxylation | X | | | | |
| HydroxylGroups | X | | | | |
| HypotheticalProtein | | | X | X | |
| InBook | | m | m | X | |
| InCollection | | m | m | X | |
| Index | X | | | | U |
| IndexNode | X | | | | U |
| InitialConditions | X | | | | U |
| InlineFunction | X | | | | |
| InProceedings | | m | m | X | |
| IntegerElement | | | | X | U |
| InternalNetwork | X | | | | U |
| InternalNetworks | | | | | |
| IntParameter | X | | | | U |
| IntVector | X | | | | U |
| Ion | X | | | | U |
| Isomerase | X | | | | |
| KEGGInfo | X | | | | U |
| Kelvin | X | | | | |
| Kilogram | X | | | | |
| KMechIndexedReactant | X | | | | U |
| kMechInhibitor | X | | | | U |
| KMechProduct | X | | | | U |
| KMechReaction | X | | | | U |
| KMechReactionGroup | X | | | | U |
| KMechResidualEnzActivity | X | | | | U |
| KMechSubstrate | X | | | | U |
| KnowledgeSource | | X | | X | U |
| Layout | X | | | | U |
| LayoutNode | X | | | | U |
| LedaEdge | X | | | | |
| LedaGraph | X | | | | |
| License | X | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Lipase | X | | | | |
| Lipid | | | X | X | |
| LipoicAcid | X | | | | |
| ListHead | X | | | | |
| Locus | | | | X | |
| Lyase | X | | | | |
| Lysosome | X | | | | |
| MacroMolecule | X | | | | U |
| Manual | | m | m | X | |
| Marker | X | | | | |
| MastersThesis | | m | m | X | |
| MathematicalReaction | | X | X | X | U |
| MathematicaNotebook | X | | | | U |
| Messenger | X | | | | |
| Metaconstraint | | | | X | |
| Meter | X | | | | |
| Methylation | X | | | | |
| MichaelisMenten | X | | | | U |
| MiscCitation | | m | m | X | |
| MiscFile | X | | | | |
| Mitochondrion | X | | | | |
| Model | | X | X | X | U |
| Modification | | X | | X | |
| Mole | X | | | | |
| Molecule | | | X | X | U |
| MultiMulti | X | | | | U |
| MWC | X | | | | U |
| Myristoylation | X | | | | |
| NAcetylhexosamines | X | | | | |
| NameSymbolPair | | | | X | |
| NameValuePair | | X | | X | U |
| NeuroTransmitter | X | | | | |
| NHCA | | X | X | X | |
| NHCACompetitive | | X | X | X | |
| Node | X | | | | U |
| NodeList | X | | | | U |
| NonCatalyzed | X | | | | U |
| NTerminalFormylation | X | | | | |
| NTerminalMyristylation | X | | | | |
| Nucleotide | X | | | | |
| NucleotideSequence | X | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Nucleus | X | | | | |
| NVPbyCompartment | X | | | | |
| ODEReaction | X | | | | U |
| ODEUserDefinedModule | X | | | | |
| OMethylesterification | X | | | | |
| Optimization | X | | | | U |
| OrderedBiBi | X | | | | U |
| ORF | | | X | X | |
| Organ | X | | | | |
| Organelle | X | | | | |
| Organism | X | | | | |
| Organisms | | | X | X | |
| OrganizationalGroup | X | | | | U |
| Oxidation | X | | | | |
| OxidationReduction | X | | | | |
| Oxidoreductase | X | | | | |
| Palmitoylation | X | | | | |
| Parameter | X | | | | U |
| ParameterizedCell | X | | | | |
| ParameterSet | | X | X | X | U |
| Particle | X | | | | |
| Pathway | | | | X | |
| Pentoses | X | | | | |
| Peptide | | | X | X | |
| Peroxisome | X | | | | |
| PhDThesis | | m | m | X | |
| PhenomenologicalRegtranscription | | | X | X | |
| Phosphorylation | X | | | | |
| Photon | X | | | | |
| Phylum | | | | X | |
| PingPongBiBi | X | | | | U |
| PingPongCompetitiveNonCompetitive | X | | | | U |
| PingPongDifferentialNonCompetitiveResidual | X | | | | U |
| PingPongNonCompetitive | X | | | | U |
| PingPongNonCompetitive2ndsub | X | | | | |
| PingPongNonCompetitive3Inhibitor | X | | | | |
| PingPongNonCompetitiveResidual | X | | | | U |
| PingPongTerTerOrderedAdditionRandomRelease | X | | | | |
| PingPongTerTerRandomAdditionOrderedRelease | X | | | | |
| PosttranslationalModification | X | | | | |
| PreDefinedCell | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Prenylation | X | | | | |
| PrimalComplex | X | | | | |
| Proceedings | | m | m | X | |
| Protein | | | X | X | |
| ProteinMultimer | | X | X | X | |
| ProteinProtein | X | | | | |
| Protocol | | m | m | X | |
| Proton | X | | | | |
| PubMethod | | | | X | |
| Pump | X | | | | |
| Pyroglutamicacid | X | | | | |
| Quantity | | | | | |
| RateConstants | X | | | | U |
| RateFunction | X | | | | U |
| Reactant | | X | | X | U |
| ReactantCoefficientPair | X | | | | U |
| ReactantData | | | | | |
| ReactantFunction | | X | | X | |
| ReactantKineticPair | X | | | | U |
| ReactantLocationPair | X | | | | U |
| Reactantmod | | | | | |
| Reactants | | | | | |
| Reaction | | X | X | X | |
| ReactionConstraint | | | | X | |
| ReactionGroup | X | | | | U |
| Receptor | X | | | | |
| Region | X | | X | | |
| RegulatoryFunction | X | | | | |
| ReplicatedReaction | X | | | | U |
| Replication | X | | | | |
| Ribosome | X | | | | |
| RNA | | | X | X | U |
| RNADNA | X | | | | |
| RNAProtein | X | | | | |
| RNARNA | X | | | | |
| Rules | X | | | | |
| SBMLInstructions | X | | | | |
| SBMLRepresentation | X | | | | |
| SDE | X | | | | |
| Second | X | | | | |
| SialicAcid | X | | | | |

| | | | | | |
|---|---|---|---|---|---|
| SIBaseUnit | X | | | | |
| SIDerivedUnit | X | | | | |
| Sign | | | | | |
| Simple | X | | | | |
| SimpleValue | | | | X | |
| SimplifiedNonSaturatedCatalytic | | X | X | X | U |
| SingleReplacement | X | | | | U |
| Site | | X | | X | |
| SmallMolecule | | | X | X | U |
| SMolecular | | | | | |
| SparceArray | | | | | |
| SparseArray | X | | | | |
| SparseMatrixElement | X | | | | |
| SpatialModel | X | | | | U |
| Species | | | | X | |
| SSystem | X | | | | |
| StateVector | X | | | | |
| Stearoylation | X | | | | |
| StochasticReaction | X | | | | |
| StringElement | X | | | | U |
| Stringl | | | | | |
| StringParameter | X | | | | U |
| Structural | | | | | |
| Structure | X | | | | |
| StructuredComplex | X | | | | |
| SubstrateC | | | | | |
| SubstrateCompetitiveInhibitorSet | X | | | | |
| Sugar | X | | | | U |
| Sulfation | X | | | | |
| Sulphydryls | X | | | | |
| SymbolicReactant | | | | X | |
| SymbolicReaction | | | | X | |
| SymbolicValue | | | | X | |
| SyntaxTree | | | | X | |
| SyntaxTreeNode | | | | X | |
| Taxon | | | | X | |
| TechReport | | m | m | X | |
| TerBi | X | | | | |
| TerTer | X | | | | |
| TerTerCompetitive | X | | | | |
| ThreeStageCatalytic | X | | | | U |

| | | | | | |
|---|---|---|---|---|---|
| TimeValuePair | X | | | | |
| Transcription | X | | | | |
| TranscriptionalRegulation | X | | | | |
| TranscriptionalTranslationalControl | X | | | | |
| TranscriptionFactor | X | | | | |
| Transferase | X | | | | |
| Translation | X | | | | |
| Translocation | X | | | | |
| Transport | X | | | | |
| TwoStageCatalytic | X | | | | U |
| UncertainValue | | | | X | |
| UniBi | X | | | | U |
| UniBiCompetitive | X | | | | U |
| UniBiCompetitiveNonCompetitive | X | | | | |
| UnidirectionalMassAction | X | | | | U |
| Units | | X | X | X | |
| UniUni | X | | | | U |
| UniUniCompetitive | X | | | | U |
| UniUniCompetitiveResidual | X | | | | |
| UniUniNonCompetitive | X | | | | |
| UniUniNonCompetitiveResidual | X | | | | |
| UniUniUncompetitive | X | | | | |
| UniUniUncompetitiveResidual | X | | | | |
| Unpublished | | m | m | X | |
| UserDefinedPC | | | | | |
| UserDefinedRegulation | X | | | | |
| UserOrganelle | | | | | |
| UserSpecifiedPTM | X | | | | |
| UserUnit | X | | | | |
| Vacuole | X | | | | |
| Value | | | | X | |
| VariableNode | X | | | | |
| Vesicle | X | | | | |
| Virus | X | | | | |
| WebKnowledgeSource | | m | m | X | U |
| YeastProteomeDatabase | | m | m | X | |
| YeastTwoHybridSystem | | m | m | X | |

## B. Model Code Authorship

Author Key: Compani is Behnam Compani. Tom is Thomas Whisenant. Kandarp is Kandarp Shah. Najdi is Tarek Najdi. Zhang is Li Zhang. Vadim is Vadim Bichiutsky.

| Model | Author | Year | Run in Math SBML | Cellerator.org | Cellerator | Cellerator author | Notebook Comments | Sigmoidable | Java code author | In Database | Layout author | Standard Icons | Verified SME Output | Complete | SME Status | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Asp Thr | Yang | | | N | Y | Yang | OK | Y | Compani | Y | Yang | Y | Y | Y | Deleted | |
| Ile Val Leu | Yang | 2005 | | N | Y | Yang | OK | Y | Compani | Y | Tarek | Y | Y | Y | Curated | |
| Asp Thr | Najdi | 2006 | | N | Y | Tarek | OK | Y | Tarek | Y | Tarek | Y | Y | Y | Curated | |
| MAPK | Bardwell | 2007 | | N | Y | Bardwell/Lucas | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | |
| MAPK Demo | Bardwell/Compa | - | | N | Y | | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | |
| NFKappaB | Hoffmann | 2002 | | N | Y | Hopkins | OK | Y | Compani | Y | Tom | Y | Y | Y | Curated | Levchenk |
| NFKappaB | Johns Hopkins | 2007 | | | | Hopkins | | | Compani | N | | | | | | |
| p53 | Bullock & Fersht | 2001 | | N | Y | Vadim/Charity | Not verified | Y | Vadim | Y | Kandarp | Y | Y | Y | Curated | |
| Calcineurin | Hilioti | 2004 | | N | Y | Tom | OK | Y | Compani | Y | Tom | Y | Y | Y | Curated | |
| Clavata | | | | N | Y | ? | OK | Y | Compani | - | Kandarp | Y | Y | N | - | |
| MAPK Scaffold | Levchenko, Shapiro, Mjolsness | | | N | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | - | |
| Wuschel | Cellzilla | | | Y | Y | Shapiro | ? | Y | Compani | N | | Y | N | Y | - | |
| Stem Cell | Chickarmane | 2006 | | N | Y | Tom | OK | Y | Tom | Y | Tom | Y | Y | Y | Curated | |
| p53 | Chickarmane | 2007 | | N | | | | | | | | | | | | |
| Algebraic | | 2008 | | N | | | | Y | Compani | Y | | Y | Y | Y | | |
| Algebraic | | 2008 | | N | | | | Y | Compani | Y | | Y | Y | Y | | |
| Xyl_Ara_Eth_Bi | | 2009 | | N | Y | Najdi | | Y | Compani | Y | Najdi | Y | Y | ? | | |
| FA_Biosynthesi | | 2010 | | N | Y | Najdi | | Y | Compani | Y | Najdi | Y | Y | ? | | |
| ATM | | 2007 | | N | Y | Zhang | | Y | Compani | Y | | Y | Y | Y | | |
| ATM_MRN_PP | | 2007 | | N | Y | Zhang | | Y | Compani | Y | | Y | Y | Y | | |
| Pi3K | | 2007 | | N | Y | Zhang | | Y | Compani | Y | | Y | Y | Y | | |
| SEP | | 2007 | | N | Y | Zhang | | Y | Compani | Y | | Y | Y | Y | | |
| Auxin | Middleton | 2008 | | | Y | | | Y | Compani | Y | Compan | Y | Y | Y | | |
| Biomodels DB | | | | | | | | | | | | | | | | |
| 9 | Huang | 1996 | Y | N | Y | Bardwell | OK | Y | Kandarp | Y | Kandarp | Y | Y | Y | Curated | |
| 13 | Poolman | 2004 | N | N | Y | Tom | OK | Y | Tom | Y | | | | | Curated | |
| 26 | Markevich | 2004 | Y | N | Y | Tom | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | OrderedElementary |
| 27 | Markevich | 2004 | Y | N | Y | Tom | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | OrderedMM |
| 28 | Markevich | 2004 | Y | N | Y | Tom | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | MAPK_phosphoRndmElemntry |
| 29 | Markevich | 2004 | Y | N | Y | Tom | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | MAPK_phosphoRandomMM |
| 30 | Markevich | 2005 | Y | N | Y | Kandarp | OK | Y | Kandarp | Y | Kandarp | Y | Y | Y | Curated | MAPK_AllRandomElementary |
| 31 | Markevich | 2004 | Y | N | Y | Kandarp | OK | Y | Kandarp | Y | Kandarp | Y | Y | Y | Curated | MAPK_AllRandomMM2Kinases |
| 32 | Kofahl | 2004 | Y | N | Y | Kandarp | OK | Y | Kandarp | Y | Kandarp | Y | Y | Y | Under Review | |
| 36 | Tyson | 1999 | Y | N | Y | Kandarp | OK | Y | Kandarp | Y | Kandarp | Y | Y | Y | Curated | |
| 37 | Marwan | 2003 | Y | N | Y | Kandarp | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | |
| 42 | Nielsen | 1998 | Y | N | Y | Kandarp | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | |
| 44 | Borghans | 1997 | Y | N | Y | Kandarp | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | |
| 45 | Borghans | 1997 | Y | N | Y | Kandarp | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | |
| 46 | Olsen | 2003 | Y | N | Y | Compani | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | |
| 48 | Kholodenko | 1999 | Y | N | Y | Compani | OK | Y | Compani | Y | Kandarp | Y | Y | Y | Curated | |
| 50 | Martins | 2003 | Y | N | Y | Tom | OK | Y | Tom | Y | Tom | Y | Y | Y | Curated | |
| 52 | Brands | 2002 | Y | N | Y | Tom | OK | Y | Tom | Y | Tom | Y | Y | Y | Curated | |

Prepared by Shah, Whisenant, and Compani 3/10/2012

## C. Posttranslational Modification Classes.

A list of posttranslational modification classes available in the Sigmoid schema. Subclasses of a class are in parenthesis. Class attributes follow the colon.

AcidsAmines: EDQN- String (Deamidation QorN: String, Pyroglutamicacid Q:String, Carboxylation EorD:String)

Amines: KorNTerminus:String (Formylation, Myristoylation, Stearoylation, Biotinylation, Farnesylation, Acylation, Palmitoylation, Geranylgeranylation, LipoicAcid, Methylation)

Carbohydrates: STN-String (Hexosamines, Deoxyhexoses, Hexoses, NAcetylhexosamines, Pentoses, SialicAcid)

CTerminalAmidation

Glycosylation LinkOorN- String

Hydroxylation

HydroxylGroups: STY -String (Phosphorylation, Sulfation)

NTerminalFormylation

NTerminalMyristylation

OMethylesterification

Prenylation

Sulphydryls: (Cysteinylation, DisulphideBond, Oxidation, Glutathionylation)

UserSpecifiedPTM kind-String