Bioinformatics, 32, 2016, i60–i69
doi: 10.1093/bioinformatics/btw250
ISMB 2016

OXFORD

BioASF: a framework for automatically generating executable pathway models specified in BioPAX

Reza Haydarlou[1], Annika Jacobsen[1], Nicola Bonzanni[1,2], K. Anton Feenstra[1], Sanne Abeln[1] and Jaap Heringa[1,*]

[1]Centre for Integrative Bioinformatics (IBIVU) & Amsterdam Institute for Molecules Medicines and Systems (AIMMS), VU University Amsterdam, De Boelelaan 1081, Amsterdam, The Netherlands and [2]NKI-AVL, The Netherlands Cancer Institute, Plesmanlaan 121, Amsterdam, The Netherlands

*To whom correspondence should be addressed.

Abstract

Motivation: Biological pathways play a key role in most cellular functions. To better understand these functions, diverse computational and cell biology researchers use biological pathway data for various analysis and modeling purposes. For specifying these biological pathways, a community of researchers has defined BioPAX and provided various tools for creating, validating and visualizing BioPAX models. However, a generic software framework for simulating BioPAX models is missing. Here, we attempt to fill this gap by introducing a generic simulation framework for BioPAX. The framework explicitly separates the execution model from the model structure as provided by BioPAX, with the advantage that the modelling process becomes more reproducible and intrinsically more modular; this ensures natural biological constraints are satisfied upon execution. The framework is based on the principles of discrete event systems and multi-agent systems, and is capable of automatically generating a hierarchical multi-agent system for a given BioPAX model.
Results: To demonstrate the applicability of the framework, we simulated two types of biological network models: a gene regulatory network modeling the haematopoietic stem cell regulators and a signal transduction network modeling the Wnt/β-catenin signaling pathway. We observed that the results of the simulations performed using our framework were entirely consistent with the simulation results reported by the researchers who developed the original models in a proprietary language.
Availability and Implementation: The framework, implemented in Java, is open source and its source code, documentation and tutorial are available at http://www.ibi.vu.nl/programs/BioASF.
Contact: j.heringa@vu.nl

1 Introduction

The National Human Genome Research Institute (NHGRI) has defined a biological pathway as 'a series of actions among molecules in a cell that leads to a certain product or a change in a cell' (http://www.genome.gov/27530687). Biological pathways play a key role in most cellular processes including metabolism, gene expression regulation and signal transduction. According to Pathguide (http://www.pathguide.org) (Bader *et al.*, 2006), there are more than 500 resources, including databases, in which biological pathway data are stored in various data formats. To increase the uniformity of pathway data from different sources, make biological pathway data exchangeable, and increase the efficiency of computational pathway research, a community of researchers have defined BioPAX (*Biological Pathway Exchange*): a semantic-web based standard language to specify biological pathway models at the molecular and cellular level (Demir *et al.*, 2010). A biological pathway specified in the BioPAX language is called a *BioPAX model*. Currently, the most comprehensive biological pathway databases, including Pathway Commons (Cerami *et al.*, 2011), KEGG (Kanehisa and Goto, 2000), NCI/Nature PID (Schaefer *et al.*, 2009), Reactome (Croft *et al.*, 2014), WikiPathways (Kelder *et al.*, 2012), and NetPath (Kandasamy *et al.*, 2010), provide pathway descriptions in BioPAX format.

Since the introduction of BioPAX, the BioPAX community has provided various tools: *Paxtools* (Demir *et al.*, 2013) for reading

and writing BioPAX models, *Validator* (Rodchenkov *et al.*, 2013) for validating a BioPAX model to see whether or not it complies with the BioPAX specification, *Pattern Search* for enabling the search of specific topological structures in a BioPAX model (Babur *et al.*, 2014) and *ChiBE* (Babur *et al.*, 2010) for visualizing BioPAX models in the standard *Systems Biology Graphical Notation* (SBGN) (Le Novère *et al.*, 2009) format. Although simulation of biological pathway models makes it possible to achieve an adequate level of understanding of pathway models, a generic software framework for simulating biological pathway models specified in BioPAX has not been available to date. We address this omission by introducing BioASF (*BioPAX-based Agent-oriented Simulation Framework*).

BioASF is different in a number of ways from the existing simulation frameworks for biological pathways handling qualitative regulatory and signaling pathway models. These frameworks are mostly based on either *Petri-nets* or *process calculi*. In a Petri-net, a *place* can represent molecules such as genes, proteins or complexes, and a *transition* can represent their interactions. Firing of a transition leads to consuming substrates and creating products. The execution is constrained by the weight of an *arc*, which connects a place with a transition. An example of a Petri-net based simulation framework is *Cell Illustrator* (Nagasaki *et al.*, 2009). In process calculi, a *communicating process* represents molecules, a *communication event* represents an interaction and a *state change* represents a modification of the molecule. An example of a process calculi based simulation framework is *Bio-PEPA* (Ciocchetta and Hillston, 2009). Note that because the focus of the simulation frameworks that are based on the *Systems Biology Markup Language* (SBML) (Hucka *et al.*, 2003) is mainly the dynamic simulation of quantitative models, we do not mention these frameworks in this paper.

One important difference between Petri-net and process calculi-based frameworks and BioASF is the language used to specify pathway models. In a typical Petri-net based simulation framework, a pathway model is specified in various proprietary languages in which molecules are assigned to places and molecular interactions are assigned to transitions. In a typical process calculi-based simulation framework, a pathway model is specified in a language containing terms such as *actions* and *synchronizations*. This language is primarily meant for specifying performance models of computer and communication systems. On the contrary, BioASF uses a standard language (BioPAX) for specifying pathway models.

Another important difference between the mentioned frameworks and BioASF is the type of network they build during simulation. Both Petri-net and process calculi-based frameworks use intrinsically flat networks and the nature of the network nodes is very generic (places and transition nodes in a Petri-net and processes and reaction nodes in process calculi). In BioASF the network nodes have a natural semantic meaning, are organized hierarchically, and the network exhibits a high degree of modularity. BioASF gets its modularity from BioPAX in which biological entities are defined in a hierarchical fashion, providing natural modularity for model developers.

Moreover, in BioASF, there is a clear distinction between the *biological network model* and *simulation execution model*. The biological network model in BioASF is represented as a network in which nodes are biological entities (such as proteins, genes, biochemical reactions, regulatory interactions), interconnected via well-defined properties and according to the constraints defined in the BioPAX language. Note that BioPAX is a specification language and it does not provide any execution semantics. In order to simulate BioPAX models, BioASF provides the simulation execution model (see Appendix). The simulation execution model is represented as a network in which nodes are agents controlling the execution of their corresponding biological interactions in the biological network model. The simulation execution model enforces three types of constraints during a model execution: (i) constraints defined in BioPAX, (ii) constraints defined by simulation rules and (iii) constraints defined by analysis rules. Hence, BioASF explicitly requires the corresponding inputs. The explicit specification of these separate inputs enhances the reproducibility and consistency of the modeling procedure. For example, from a biological viewpoint, a protein cannot be translated from another protein. RNA is translated into proteins. BioPAX respects this constraint. The constraints provided by BioPAX disallows a *template reaction* interaction (an interaction that polymerizes its *product* based on a *template*, and is used for specifying transcriptions, translations and replications) to accept a protein or small molecule as its template. BioPAX requires the template to be either an RNA (for specifying translations) or DNA (for specifying transcriptions and DNA replications). However, due to intrinsically very generic nature of nodes in Petri-net (and also process calculi) frameworks, it is possible to define a node representing a protein and accidentally use this as a template for a transcription, translation, or a replication reaction.

Figure 1 depicts the different inputs and the output of BioASF. A BioASF user provides biological network data specified in BioPAX, a list of initial concentration values of the physical entities, simulation rules indicating how biological interactions should be performed, and analysis rules indicating how the simulation results over time are interpreted and analyzed. Based on these inputs, BioASF automatically generates a hierarchical multi-agent system where each agent in the system is a goal-directed and autonomous entity responsible for executing a BioPAX interaction and pathway. Throughout the simulation, the result of the execution of a BioPAX interaction influences the behavior of the agents and makes changes in the concentration of physical entities in the environment. During the simulation, it is possible to query BioASF about the current simulation data such as number and type of executed interactions and concentration level of physical entities. The following sections elaborate in more detail on the internal components, organization and various features of BioASF.

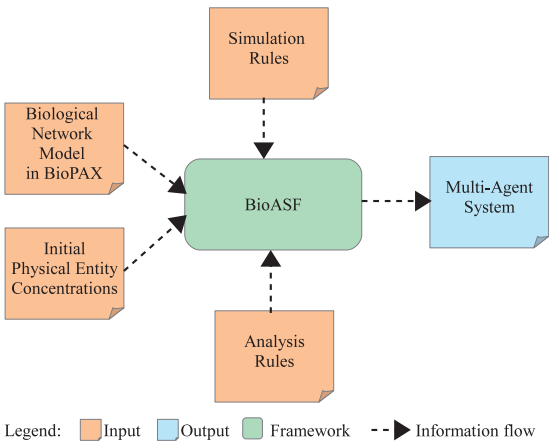

Fig. 1. Inputs and output of BioASF. Inputs are a pathway description in BioPAX, initial concentration values of physical entities such as proteins, and rules governing the execution of interactions and the analysis of simulation results. The output is a multi-agent system which is automatically generated by BioASF and is responsible for executing BioPAX interactions

2 Methods

2.1 The BioPAX language

The BioPAX language is based on the *Ontology Web Language* (OWL) (Bechhofer *et al.*, 2004), which is a description logic language intended to augment the current web with formalized knowledge in order to make information on the web machine-processable. Figure 2 depicts the BioPAX core concepts and their hierarchical relationships. Two main core concepts of the BioPAX language are *pathway* and *interaction*. A pathway is defined as a series of interactions demarcated by biologists in order to group interactions for certain biological reasons. A pathway can contain sub-pathways and interactions. An interaction represents a biological relationship among *biological entities*. A biological entity can be an interaction, pathway, *gene*, or *physical entity* (such as *protein* and *DNA region*). Interactions can transform one or more physical entities to others (*conversion interactions*) or control and regulate other interactions (*control interactions*). Interactions can be shared between different pathways (Demir *et al.*, 2010).

There are four types of conversion interactions: (i) *biochemical reaction* representing a conversion in which molecules of physical entities undergo covalent modifications, (ii) *complex assembly* representing a conversion in which either a set of physical entities aggregate to from a complex or a complex is disassembled into its constituents, (iii) *degradation* representing a conversion in which a macro-molecule is degraded into its elementary units and (iv) *transport* representing a conversion in which a physical entity changes its sub-cellular location (Anwar *et al.*, 2010).

BioPAX defines three types of control interactions: (i) *catalysis* representing an interaction in which a physical entity increases the rate of a conversion interaction, (ii) *modulation* representing an interaction in which a physical entity modulates a catalysis interaction and (iii) *template reaction regulation* representing an interaction in which a physical entity (such as a transcription factor) regulates the expression of a macro-molecule from a template macro-molecule (such as a DNA region) (Anwar *et al.*, 2010). These BioPAX core concepts together with the BioPAX utility concepts and language properties form the *BioPAX meta-model*.

The basic idea behind BioASF is to associate an agent to each BioPAX pathway and BioPAX interaction in order to manage the execution of them. Note that execution of pathways and interactions change the concentration of physical entities in the environment. Additional agents are needed for managing the biological environment and analyzing changes in it. By representing execution managers as agents, we can base our framework on multi-agent technology. The challenge is to find a suitable communication pattern between different types of agents, as pathways are related to other pathways and interactions, and interactions are also related to other interactions and biological entities. We have formally specified all different types of agents, their inputs and outputs and their communications in mathematical set notation (see Appendix).

2.2 Architecture of the framework

The framework is based on the principles of the *discrete event systems* (DES) theory (Cassandras and Lafortune, 2009) and the *multi-agent systems* (MAS) (Wooldridge, 2009) paradigm. Similar to a discrete event system, the various components of the framework cooperate with each other to perform a function by utilizing discrete events. Each component manages a number of discrete states, and the state transition depends on the occurrence of discrete events. Corresponding to a multi-agent system, each component of the framework is represented as an agent which is autonomous, exhibits goal-directed behavior, and operates according to its own rules. Figure 3 shows the event-based cooperation of different agents of the framework that realize a biological simulation.

A simulation is started by activation of the `Analysis Agent` that in turn activates the `Environment Agent`. This agent loads the initial concentrations of all relevant physical entities (such as proteins and small molecules) from a storage. After environment initialization, the `Analysis Agent` activates the *master pathway agent* (MPA). MPA is responsible for bootstrapping the simulation and controlling all top-level pathway agents. Pathway agents in the framework are organized in a hierarchical fashion, with the MPA at the top of the hierarchy. There is also a hierarchical relationship between a pathway agent and interaction agents. Hence, both sub-pathway agents and interaction agents belonging to a pathway are considered the pathway's children. The MPA activates all top level pathway agents each of which in turn activates its immediate sub-

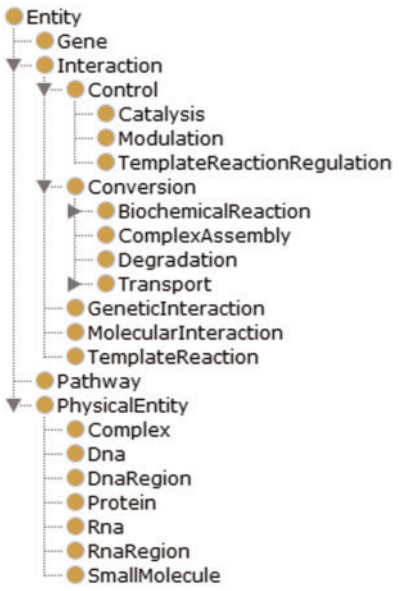

Fig. 2. A snapshot from the Protégé OWL editor (http://protege.stanford.edu/overview/protege-owl.html) showing the core concepts of the BioPAX language and their hierarchical relationships

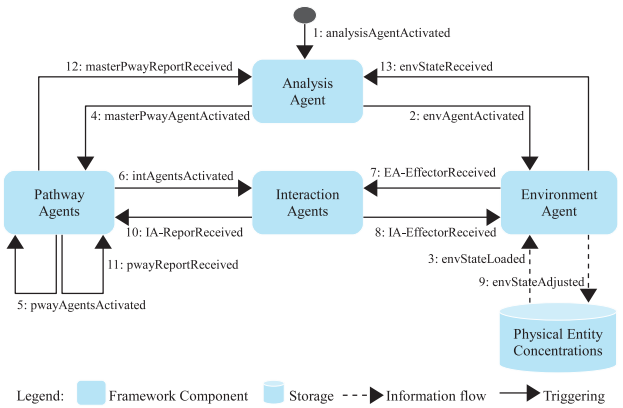

Fig. 3. High level architecture of BioASF showing how different agents communicate with each other by exchanging events. Events are numbered chronologically

pathway agents and its interaction agents. An activated pathway agent is now ready to listen to events coming from its children.

After the activation of an `Interaction Agent` is completed, the activated `Interaction Agent` subscribes itself to any changes in the concentration of particular physical entities published by the `Environment Agent`. An interaction agent receiving such an event checks its *simulation rules* to see whether the required condition for performing its associated BioPAX interaction can be met. For example, a biochemical reaction with two participants, each requiring a certain stoichiometry can only be performed if the concentrations of both participants are equal or higher than their stoichiometric value. Besides, if this biochemical reaction is controlled by a catalysis interaction then it must be checked to see whether the catalysis interaction has already occurred and whether its effector (i.e. its result) is available. As soon as the condition is met, the `Interaction Agent` performs its associated BioPAX interaction and sends its effector to the `Environment Agent`. At the same time, the `Interaction Agent` sends an event to its parent `Pathway Agent`. An example payload of such an event is a report indicating whether or not an interaction has occurred. The `Pathway Agent` integrates such reports received from its children and sends one integrated report to its parent. Finally, the MPA integrates all reports from the top level pathway agents and informs the `Analysis Agent` about the situation regarding the execution of all interactions. Consequently, `Analysis Agent` utilizes its *analysis rules* to store simulation results, examine environmental changes (e.g. changes in the concentration of physical entities), combine the current simulation results with the previous ones and initiate a new cycle of simulation with new initial concentrations.

2.3 Artifacts of the framework

One of the important features of the framework is that it is able to generate pathway and interaction agents from a BioPAX model, and there is no need for users to write code for these agents. This feature is realized by utilizing various artifacts. Note that in software engineering, an *artifact* is a physical piece of information such as a data file, source code file and library file that is used or produced during a software development process. We use this terminology to explain different software elements of the BioASF framework depicted in Figure 4.

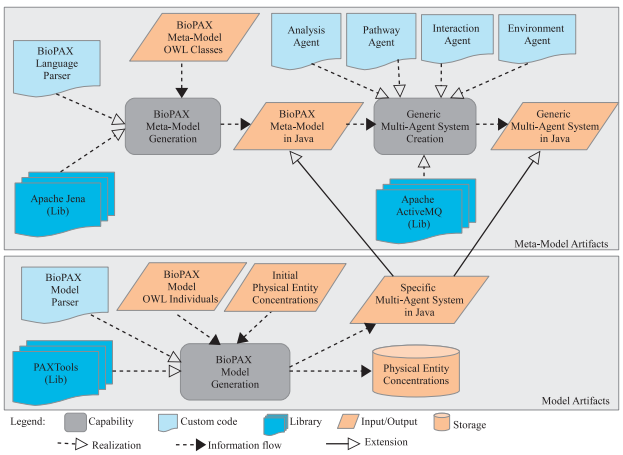

Fig. 4. Meta-model layer and model layer artifacts of BioASF showing how the multi-agent system code in Java is generated for simulating the biological pathway specified in a given BioPAX model

We make a distinction between meta-model artifacts and model artifacts. Meta-model artifacts are related to the BioPAX language constructs (OWL classes and properties) but model artifacts are related to particular pathways and interactions expressed in the BioPAX language (OWL individuals), which is also known as a *BioPAX model*.

Meta-model artifacts are obtained by two capabilities: `BioPAX Meta-Model Generation` and `Generic Multi-Agent System Creation`. The first capability takes as its input the BioPAX OWL classes and properties, parses the input by utilizing the `Apache Jena` library (Apache, 2009), and produces Java classes which are related to each other in the same way as the BioPAX OWL classes. The second capability uses the meta-level Java classes (which are the Java representation of the BioPAX meta-model) and the manually developed Java classes (which contain code for all different types of agents) to produce the generic multi-agent system. This multi-agent system utilizes the messaging framework `Apache ActiveMQ` (Apache, 2005) for sending and receiving message events among agents. Also the Java classes of the multi-agent system are related to each other in the same fashion as the BioPAX OWL classes. The meta-model artifacts are created once in the life-cycle of the framework. They may require re-generation or adjustment if a new version of BioPAX is released.

Model artifacts are obtained by the capability `BioPAX Model Generation` which takes as its input the BioPAX OWL individuals, parses the input by utilizing the `PAXTools` library (Demir *et al.*, 2013), and produces a specific multi-agent system which is an extension of the meta-level generic multi-agent system. Moreover, BioASF also generates a storage containing the concentration of all physical entities (such as proteins, complexes, small molecules) that are part of a BioPAX model.

2.4 Flexibility of the framework

The framework offers flexibility on the following four aspects: simulation rules, simulation modes, analysis mechanisms and scalability. BioASF allows users to define their own simulation rules, choose their proper simulation mode, incorporate their own analysis mechanisms and run simulation in a single or multiple *Java Virtual Machine* (JVM). The following four paragraphs briefly explain these aspects.

Obviously, it is impossible to have a one-size-fits-all set of rules governing all types of simulations. Each simulation requires its specific rules. For example, the authors in (Albert and Othmer, 2003) propose a Boolean model of regulatory interactions to get insights into the functioning of the segment polarity gene network in Drosophila. In their model, they represent mRNA or protein by a node and the interactions between them by a directed edge. The state of each node is determined by a Boolean function which is highly specific for each node and it does not follow a generic pattern. Another example is the network model of the cell cycle regulatory network of fission yeast (Davidich and Bornholdt, 2008). In the yeast cell cycle network, similar to the previous example, the nodes of the network are genes, and each node is assigned a binary value. Differently to the previous example, this value is determined by a generic rule for all nodes, except for a few nodes. However, this generic rule is highly specific for the regulatory processes that control the cell cycle in fission yeast. Simulation rules are thus determined by the type of a biological network under study, the goal of an experiment and the experimental data. Our framework provides well-defined programming interfaces that can be used to integrate specific simulation rules with the remaining parts of the framework.

BioASF supports three modes of simulations: *synchronous* (Garg *et al.*, 2008), *asynchronous* (Garg *et al.*, 2008) and *maximally parallel* (Bonzanni *et al.*, 2009b; Burhard, 1983). The key difference between these modes is whether or not interaction agents within one execution step execute independently from each other. In synchronous mode, all interaction agents consume/produce physical entities simultaneously, and consumption/production of physical entities by one interaction agent has no influence on the other interaction agents. In asynchronous mode, interaction agents consume/produce physical entities in different times such that a physical entity produced by an interaction agent is immediately available for other interaction agents. Hence, they compete for consumption of the shared physical entities. In maximally parallel mode, interaction agents compete for consumption of physical entities but the production of one interaction agent is not immediately available for other interaction agents within the same execution step.

Each experiment requires its own approach for analyzing simulation results to get insight into the behavior of the model under study. For example, in an *in silico* experiment for cell fate determination of *C. elegans* during vulval induction, the simulation results are used as variables for specific scoring functions to assign to a cell the fate corresponding to the highest scoring function (Bonzanni *et al.*, 2009b). In another *in silico* experiment for studying the behavior of the regulatory network of 11 haematopoietic stem/progenitor cell genes, the simulation results are used to find the steady state of the model by computing the terminal strongly connected components of the state space of the model (Bonzanni *et al.*, 2013). These examples show that different analysis mechanisms may be used in each experiment. Our framework provides well-defined programming interfaces for integrating specific analysis mechanisms with the remaining parts of the framework.

Since a BioPAX model may contain multiple pathways each consisting of a large number of interactions, running simulation of such large models on one JVM (i.e. on one machine) can cause considerable problems. Therefore, BioASF provides two types of communication among agents: local communication using the Java *observer* and *observable* pattern (where a Java object (observable) maintains a list of its dependents (observers) and notifies them automatically of any state changes) and remote communication using the messaging system *Apache ActiveMQ* (where agents are distributed across a network and communicate with each other through *queuing* and *publish/subscribe* mechanisms).

3 Results

In order to validate BioASF, we have simulated two different biological network models in BioASF: a gene regulatory network modeling the *haematopoietic stem cell* (HSC) regulators (Bonzanni *et al.*, 2013) and a signal transduction network modeling the *Wnt/β-catenin* signaling pathway (Jacobsen *et al.*, 2016). Both original models have been developed in Petri-net. We first made BioPAX models for both networks, used the BioASF's plugin to express the interaction rules required by the models, plugged the corresponding analysis module in BioASF, prepared the initial concentration values for the physical entities occurring in the models, generated Java code for all agents and finally ran the simulations. We observed that the results of the simulations performed using BioASF were similar to the results reported by the researchers in Bonzanni *et al.* (2013) and Jacobsen *et al.* (2016). In the following sections, we give a brief explanation of the simulations and show the results.

3.1 Gene regulatory simulation

3.1.1 Model description

Haematopoiesis has long served as a model process for studying stem cells and represents the best characterized adult stem cell system. Transcriptional regulation is a key factor controlling haematopoiesis. In (Bonzanni *et al.*, 2013), the authors build the first comprehensive regulatory network model based on the systematic curation of 11 fully validated regulatory elements linking together 11 transcription factors, all of which are active in early haematopoietic stem/progenitor cells. All 11 regulatory elements included in the model have been previously published and studied extensively using DNA/protein-binding assays, as well as reporter gene assays of wild-type and mutant elements. The direction and value of each of the regulatory interactions are also known with certainty. Moreover, protein–protein interactions curated from the literature were included. The resulting network was modeled as logical interactions encoding the activating and/or inhibitory links, including the specific combinations in which particular interactions occur.

3.1.2 BioPAX representation

We considered the gene regulatory network controlling haematopoiesis as a BioPAX pathway and defined a BioPAX model. Figure 5 shows the graphical representation of the BioPAX model in SBGN format. In the model, we represented the gene product definitions for all genes used in the network as the BioPAX `Protein`, the transcription of the genes as the BioPAX `TemplateReaction` and the regulatory role of the transcription factors in the transcription as the BioPAX `TemplateReactionRegulation`. For example, as shown in Figure 6, `TmpReg-PU1-GATA1-GATA1` regulates (inhibits) the transcription of `GATA1 (TmpReac-GATA1)` through its controllers `PU1` and `GATA1`.

3.1.3 Simulation rules

For the simulation of the haematopoiesis model, we extracted a number of simple rules from the description of the original *in silico* experiment (Bonzanni *et al.*, 2009a, 2013), and plugged them in BioASF. We recognized three categories of rules: template reaction regulation rules, template reaction rules and degradation rules.

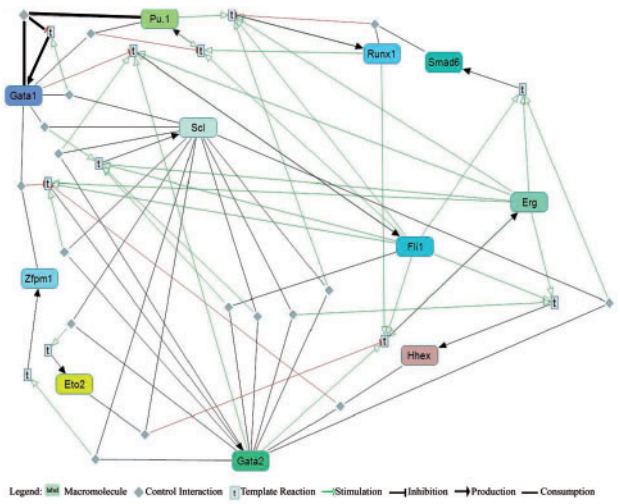

Fig. 5. SBGN representation of the haematopoiesis gene regulatory network generated by ChiBE (Babur *et al.*, 2010) based on our BioPAX model. The highlighted part of the graph corresponds to the part of the BioPAX model depicted in Figure 6

```
<bp:Protein rdf:ID="GATA1">
 <bp:displayName rdf:datatype = "&xsd;string">
     GATA1</bp:displayName>
</bp:Protein>
  ...
<bp:TemplateReaction rdf:ID="TmpReac-GATA1">
 <bp:product rdf:resource="#GATA1"/>
</bp:TemplateReaction>
  ...
<bp:TemplateReactionRegulation
           rdf:ID="TmpReg-PU1-GATA1-GATA1">
 <bp:controlType rdf:datatype="&xsd;string">
           INHIBITION</bp:controlType>
 <bp:controller rdf:resource="#PU1"/>
 <bp:controller rdf:resource="#GATA1"/>
 <bp:controlled rdf:resource="#TmpReac-GATA1"/>
</bp:TemplateReactionRegulation>
  ...
<bp:Pathway rdf:about="Haematopisis">
  <bp:pathwayComponent rdf:resource="#TmpReac-GATA1" />
  <bp:pathwayComponent rdf:resource=
           "#TmpReg-PU1-GATA1-GATA1" />
  ...
</bp:Pathway>
```

Fig. 6. Part of the BioPAX model for the haematopoiesis gene regulatory network

These rules determine whether or not a BioPAX interaction should be performed during the simulation. A template reaction regulation interaction can be performed if the concentration values of its inputs (i.e. all of its controllers) are higher than zero. After performing this interaction, the concentration value of its output is increased by one by the environment agent. There are two conditions which must be fulfilled before a template reaction interaction can be performed: (i) the template reaction regulation interaction regulating the template reaction interaction must already have occurred, and (ii) the concentration value of the output (i.e. product) of the template reaction interaction in the environment must either be zero (in case, the control type is `ACTIVATION`) or one (in case, the control type is `INHIBITION`). After performing this interaction, the concentration value of its output is either increased or decreased by one by the environment agent, depending on the control type of its regulator. According to the degradation rules, the output of a template reaction interaction is degraded if the concentration value of one of its inputs in the environment is either zero (in case, the control type is `ACTIVATION`) or one (in case, the control type is `INHIBITION`).

As the firing of an interaction in the haematopoiesis Petri-net model has no impact on the other interactions, within one execution step, this model was simulated in BioASF in the synchronous mode.

3.1.4 Model analysis

For the analysis of the simulation results of the haematopoiesis model, we created an analysis module in Java and plugged it in BioASF. This module initially constructs the state space graph of 2048 nodes each containing a bit vector of size 11, initiates the concentration value of all gene products (0 or 1) at the beginning of each simulation step for each state, utilizes BioASF to execute one simulation step, and updates the graph based on the simulation results. In line with the original *in silico* experiment (Bonzanni *et al.*, 2009a, 2013), at the end of the simulation, after the graph was completely built, the *terminal strongly connected components* (TSCC) of the state space are computed in order to identify the attractors which represent the biological steady states.

As reported by the original *in silico* experiment, we found three TSCCs: the TSCC-1 containing one state that corresponds to a non-haematopoietic cell, the TSCC-2 containing one state that closely resembles a mature erythrocyte, and the TSCC-3 containing 32 states that match the expected pattern for haematopoietic stem cells, as shown in Table 1.

3.2 Signal transduction simulation

3.2.1 Model description

Wnt/β-catenin signaling is highly conserved in all animals and is important for embryonic development and stem cell maintenance in adults. WNT ligand stimulation leads to inhibition of an important destruction complex that degrades CTNNB1 (β-catenin). This leads to CTNNB1 accumulation and translocation to the nucleus, where it binds TCF/LEF and activates transcription of WNT target genes. In Jacobsen *et al.* (2016), the authors build a Wnt/β-catenin signaling model based on interactions, well established in literature. Signaling is initiated by the binding of external WNT ligand to the receptor, FZD, and the co-receptor, LRP. Dishevelled (DVL) and AXIN bind the intracellular tail of FZD. AXIN, APC, CK1 and GSK3 form the destruction complex. CTNNB1 is constitutively produced and either binds the destruction complex for degradation or translocates to the nucleus, where it activates transcription of WNT target genes.

3.2.2 BioPAX representation

Figure 7 shows the graphical representation of the BioPAX model for the Wnt/β-catenin signaling pathway in SBGN format. In the BioPAX model, we used a number of BioPAX classes including `DnaRegion` (for specifying the genes such as ctnnb1), `ComplexAssembly` (for the formation of a complex from a number of proteins such as the formation of the destruction complex from AXIN, APC, CK1 and GSK3) and `Degradation` (for specifying the degradation of CTNNB1).

Figure 8 shows a part of the BioPAX model in which the complex assembly interaction `CmpAsm-TCF-LEF-CTNNB1` uses `CTNNB1` and `TCF-LEF` to create the `TCF-LEF-CTNNB1` complex. This complex plays the role of transcription factor in the template regulation interaction `TmpReg-TCF-LEF-CTNNB1-TARGET`, which regulates (activates) the transcription of the WNT target genes. In order to slow down the rate of the complex assembly interaction and the transport of `CTNNB1` from the nuclear to the cytoplasm, we used the BioPAX `Stoichiometry` class.

3.2.3 Simulation rules

Similar to the haematopoiesis simulation, for the simulation of the WNT/β-catenin model, we extracted a number of rules from the description of the original *in silico* experiment (Jacobsen *et al.*, 2016), and plugged them in BioASF. We recognized three categories of rules: template reaction regulation rules, template reaction rules and conversion rules. The conversion rules are applicable for all complex assembly and degradation interactions. We also defined a generic rule which determines whether or not an interaction is allowed to be executed. According to this rule, an interaction might be performed if the stoichiometries of all participants of an interaction are equal or higher than the available concentration values of the participants. In keeping with the Petri-net formalism, the stoichiometry of a participant is assumed to be one if it has not been specified.

Table 1. The result of the analysis of the haematopoiesis simulation. Each row shows a TSCC (i.e. stable state) in which genes (the columns 2–11) have been switched on (1) or off (0)

Comp	Erg	Eto2	Fli1	GATA1	Gata2	Hhex	Pu.1	Runx1	Scl	Smad6	Zfpm1
TSCC-1	0	0	0	0	0	0	0	0	0	0	0
TSCC-2	0	0	0	1	0	0	0	0	1	0	0
TSCC-3	0	0	1	0	0	1	1	0	1	1	0
TSCC-3	1	0	1	0	0	1	1	0	1	1	0
TSCC-3	0	1	1	0	0	1	1	0	1	1	0
TSCC-3	1	1	1	0	0	1	1	0	1	1	0
TSCC-3	0	0	1	0	1	1	1	0	1	1	0
TSCC-3	1	0	1	0	1	1	1	0	1	1	0
TSCC-3	0	1	1	0	1	1	1	0	1	1	0
TSCC-3	1	1	1	0	1	1	1	0	1	1	0
TSCC-3	0	0	1	0	0	1	1	1	1	1	0
TSCC-3	1	0	1	0	0	1	1	1	1	1	0
TSCC-3	0	1	1	0	0	1	1	1	1	1	0
TSCC-3	1	1	1	0	0	1	1	1	1	1	0
TSCC-3	0	0	1	0	1	1	1	1	1	1	0
TSCC-3	1	0	1	0	1	1	1	1	1	1	0
TSCC-3	0	1	1	0	1	1	1	1	1	1	0
TSCC-3	1	1	1	0	1	1	1	1	1	1	0
TSCC-3	0	0	1	0	0	1	1	0	1	1	1
TSCC-3	1	0	1	0	0	1	1	0	1	1	1
TSCC-3	0	1	1	0	0	1	1	0	1	1	1
TSCC-3	1	1	1	0	0	1	1	0	1	1	1
TSCC-3	0	0	1	0	1	1	1	0	1	1	1
TSCC-3	1	0	1	0	1	1	1	0	1	1	1
TSCC-3	0	1	1	0	1	1	1	0	1	1	1
TSCC-3	1	1	1	0	1	1	1	0	1	1	1
TSCC-3	0	0	1	0	0	1	1	1	1	1	1
TSCC-3	1	0	1	0	0	1	1	1	1	1	1
TSCC-3	0	1	1	0	0	1	1	1	1	1	1
TSCC-3	1	1	1	0	0	1	1	1	1	1	1
TSCC-3	0	0	1	0	1	1	1	1	1	1	1
TSCC-3	1	0	1	0	1	1	1	1	1	1	1
TSCC-3	0	1	1	0	1	1	1	1	1	1	1
TSCC-3	1	1	1	0	1	1	1	1	1	1	1

Another generic rule is that the concentration value of the output of an interaction is increased by one by the environment agent, after the interaction was performed. The changes in the concentration value of the input of an interaction depend on its type. The concentration values of the inputs of the template reaction regulation and template reaction interactions are decreased by one, and the ones of the conversion interactions are decreased by the stoichiometry of their participants. As the firing of an interaction in the WNT/β-catenin model leads to the consumption of its input physical entities and they are not any more available for other interactions, within the same execution step, this model was simulated in BioASF in the maximally parallel mode.

3.2.4 Model analysis

Similar to the gene regulatory simulation, for the analysis of the simulation results of the WNT/β-catenin model, we created an analysis module in Java and plugged it in BioASF. This module initially populates the pre-defined concentration values of all proteins, complexes and genes, creates a file for writing the results of the simulations, utilizes BioASF to execute one simulation step, and writes the CTNNB1 levels in the file. In line with the original *in silico* experiment (Jacobsen *et al.*, 2016), the model is simulated with initial WNT levels ranging from 0 to 5 over 100 steps. The simulation for each initial WNT level is repeated 50 times and the CTNNB1 levels for each step is stored in a file. Finally, an R module uses this file to calculate, for each initial WNT level, the mean CTNNB1 level for each step over the 50 simulations and plots these.

The BioASF simulations closely resemble the increase in the CTNNB1 levels from the original Petri-net simulations, as shown in Figure 9. Similar to the original simulations, we observe four different CTNNB1 levels in response to the initial WNT levels (0–5): a horizontal line in response to the WNT levels of 0–2, and three oblique lines with three slopes in response to the initial WNT levels of 3–5.

4 Discussion

We introduced here a simulation software framework, called BioASF, for simulating biological pathway models specified in BioPAX. We demonstrated the applicability of BioASF by simulating models of haematopoiesis and Wnt/β-catenin signaling, which reproduced the results reported by the researchers who developed the original models. These two models had already been developed in our group and therefore the exact data of the original *in silico* experiments (initial concentration, simulation and analysis rules) were

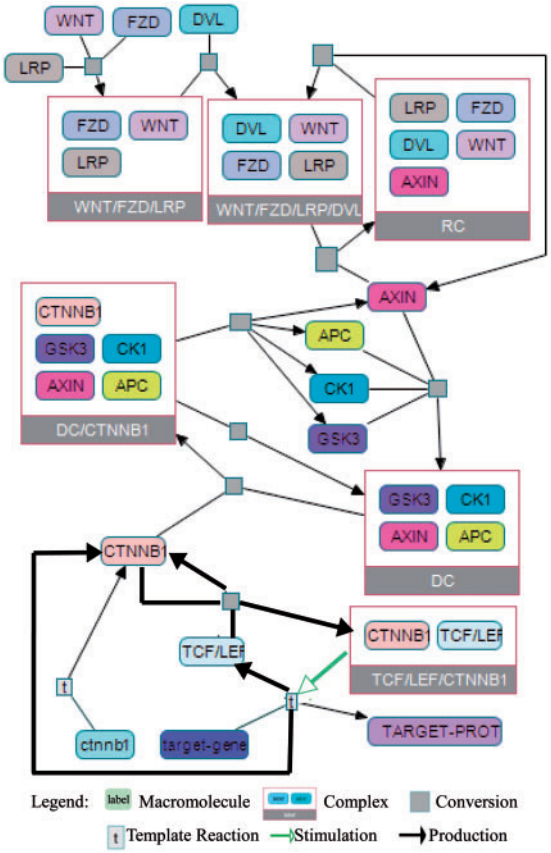

Fig. 7. SBGN representation of the WNT/β-catenin signaling pathway generated by ChiBE (Babur *et al.*, 2010) based on our BioPAX model. The highlighted part of the graph corresponds to the part of the BioPAX model depicted in Figure 8

```
<bp:ComplexAssembly rdf:ID="CmpAsm-TCF-LEF-CTNNB1">
 <bp:left rdf:resource="#CTNNB1" />
 <bp:left rdf:resource="#TCF-LEF" />
 <bp:participantStoichiometry rdf:resource=
      "#Stoi-CmpAsm-Left-CTNNB1" />
 <bp:right rdf:resource="#TCF-LEF-CTNNB1" />
 <bp:right rdf:resource="#CTNNB1" />
 <bp:participantStoichiometry rdf:resource=
      "#Stoi-CmpAsm-Right-CTNNB1" />
</bp:ComplexAssembly>
<bp:Stoichiometry rdf:ID="Stoi-CmpAsm-Left-CTNNB1">
 <bp:physicalEntity rdf:resource="#CTNNB1" />
 <bp:stoichiometricCoefficient rdf:datatype =
      "&xsd;float">3.0</bp:stoichiometricCoefficient>
</bp:Stoichiometry>
<bp:Stoichiometry rdf:ID="Stoi-CmpAsm-Right-CTNNB1">
 <bp:physicalEntity rdf:resource="#CTNNB1" />
 <bp:stoichiometricCoefficient rdf:datatype =
      "&xsd;float">2.0</bp:stoichiometricCoefficient>
</bp:Stoichiometry>
<bp:TemplateReactionRegulation rdf:ID=
      "TmpReg-TCF-LEF-CTNNB1-TARGET">
 <bp:controlType rdf:datatype="&xsd;string">
      ACTIVATION</bp:controlType>
 <bp:controller rdf:resource="#TCF-LEF-CTNNB1"/>
 <bp:controlled rdf:resource="#TmpReac-TARGET"/>
</bp:TemplateReactionRegulation>
  ...
<bp:Pathway rdf:about="Wnt-Experiment">
  ...
  <bp:pathwayComponent rdf:resource=
      "#CmpAsm-TCF-LEF-CTNNB1" />
  <bp:pathwayComponent rdf:resource=
      "#TmpReg-TCF-LEF-CTNNB1-TARGET" />
  ...
</bp:Pathway>
```

Fig. 8. Part of the BioPAX model for the WNT/β-catenin signaling pathway

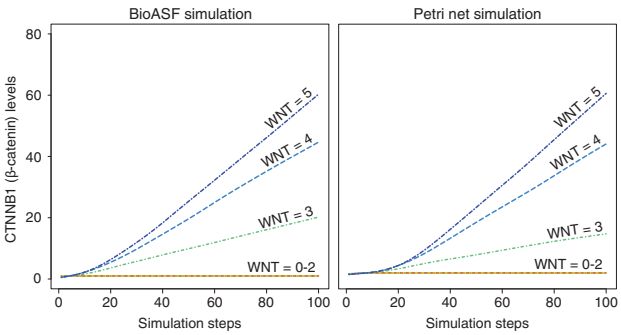

Fig. 9. The CTNNB1 (β-catenin) levels for initial WNT levels from 0 to 5 simulated 50 times over 100 steps using BioASF and Petri-net (Jacobsen *et al.*, 2016), respectively

available. The availability of this data is required in order to compare the simulation results of BioASF with the results of the original ones. Although obtaining such input data to compare modelling frameworks may be difficult, particularly when originating in other groups, we feel that the clarity of BioASF and its explicit input requirements safeguard sound simulation of pathway models developed by other groups.

The main advantages of BioASF are: (i) ability to directly simulate pathway models specified in the standard language for pathway descriptions (i.e. BioPAX), (ii) support for hierarchical (nested) pathway simulations, (iii) inherent scalability because of using agent technology, (iv) extensibility by providing pluggable architecture, (v) provision of flexibility to the biological model developers by enforcing the *separation of concerns* principles (i.e. a clear distinction is made among biological model, execution model, simulation rules and simulation result analysis) and (vi) formal specification of the execution model in mathematical set notation. The current BioASF limitations are: (i) disability to simulate differential equations, (ii) required rules can only be specified in Java and (iii) no user interface for starting/stopping/debugging simulations. This makes the framework well suited for developers, but not yet easily accessible to biologists. Nevertheless, the web-site includes an extended tutorial to help non-expert users getting started.

As mentioned in Section 2.4, each simulation requires its specific rules on pathway and interaction levels, governing execution of a pathway or interaction. In BioASF, these rules currently are specified in Java, which negatively influences the framework's usability and deployability. The preferred way would be to express simulation rules in a rule language such as *Semantic Web Rule Language* (SWRL) (Horrocks *et al.*, 2004) and relate these rules to the biological interactions in BioPAX models. This is similar to the approach taken in the *Semantic Markup for Web Services* (OWL-S) (Martin *et al.*, 2004) where the pre-conditions of an OWL-S service can be expressed in a rule language. In this way, both biological models and simulation rules can be provided by the same users (e.g. biomedical researchers).

The BioASF framework can be used in different scenarios. For example, the framework can be used for model calibration. In this

scenario, a BioPAX model can be visualized in SBGN and shown to users. Users can define break-points and watch-points on pathways and interactions, similar to the debugging of application codes. When a break-point is hit, users can then inspect the concentration value of the related physical entities and they can change these values during the simulation in order to influence the simulation results and consequently adjust their model. Another possibility would be to use BioASF as a tool for measuring the fitness of individuals in the population in a genetic programming experiment. The purpose of such an experiment would be to automatically generate biological pathways based on a combination and integration of gene expression data, protein-to-protein interaction data, existing pathway data, gene ontology data and triplets representing other biological data. Our future work will be focused on these items.

Acknowledgement

We thank Özgün Babur for providing helpful information about the BioPAX tools *ChiBE*, *Paxtools* and *Pattern Search*.

Funding

This work is part of the research programme 'Engineering Executable Models of Biological Networks' (612.001.203), which is financed by the Netherlands Organisation for Scientific Research (NWO).

Conflict of Interest: none declared.

References

Albert,R. and Othmer,H.G. (2003) The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster*. *J. Theor. Biol*., **223**, 1–18.

Anwar,N. *et al*. (2010) BioPAX – Biological Pathways Exchange Language Level 3, Release Version 1 Documentation. *BioPAX Workgroup*.

Apache Software Foundation (2005) Apache ActiveMQ: a powerful open source messaging and Integration Patterns server.

Apache Software Foundation (2009) A free and open source Java framework for building Semantic Web and Linked Data applications.

Babur,Ö. *et al*. (2010) ChiBE: interactive visualization and manipulation of BioPAX pathway models. *Bioinformatics*, **26**, 429–431.

Babur,Ö. *et al*. (2014) Pattern search in BioPAX models. *Bioinformatics*., **30**, 139–140.

Bader,G. *et al*. (2006) Pathguide: a pathway resource list. *Nucleic Acids Res*., **34**, D504–D506.

Bechhofer,S. *et al*. (2004) OWL – Web Ontology Language Reference. *World Wide Web Consortium*.

Bonzanni,N. *et al*. (2009a) *What Can Formal Methods Bring to Systems Biology? in FM 2009: Formal Methods, Volume 5850 of Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg, pp. 16–22.

Bonzanni,N. *et al*. (2009b) Executing multicellular differentiation: quantitative predictive modelling of C.elegans vulval development. *Bioinformatics*, **25**, 2049–2056.

Bonzanni,N. *et al*. (2013) Hard-wired heterogeneity in blood stem cells revealed using a dynamic regulatory network model. *Bioinformatics*, **29**, i80–i88.

Burhard,H.D. (1983) On priorities of parallelism: Petri nets under the maximum firing strategy. In: *Logics of Programs and Their Applications*, vol. 148. Springer, Berlin/Heidelberg, pp. 15, 18, 32.

Cassandras,C. and Lafortune,S. (2009) Introduction to Discrete Event Systems. *Springer*, **2**, 2–35.

Cerami,E. *et al*. (2011) Pathway Commons: a web resource for biological pathway data. *Nucleic Acids Res*., **39**, D685–D690.

Ciocchetta,F. and Hillston,J. (2009) Bio-PEPA: a framework for the modelling and analysis of biological systems. *Theor. Comput. Sci*., **410**, 3065–3084.

Croft,D. *et al*. (2014) The Reactome pathway knowledgebase. *Nucleic Acids Res*., **42**, D472–D477.

Davidich,M. and Bornholdt,S. (2008) Boolean network model predicts cell cycle sequence of fission yeast. *PLoS ONE*., **3**, e1672.

Demir,E. *et al*. (2010) The BioPAX community standard for pathway data sharing. *Nat. Biotech*., **28**, 935–942.

Demir,E. *et al*. (2013) Using biological pathway data with paxtools. *PLoS Comput. Biol*., **9**, e1003194.

Garg,A. *et al*. (2008) Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinformatics*, **24**, 1917–1925.

Horrocks,I. *et al*. (2004) *SWRL: A Semantic Web Rule Language Combining OWL And RuleML*. World Wide Web Consortium.

Hucka,M. *et al*. (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, **19**, 524–531.

Jacobsen,A. *et al*. (2016). Construction and Experimental Validation of a Petri net Model of Wnt/β-catenin Signaling. *Available in bioRxiv*. doi:10.1101/044966.

Kandasamy,K. *et al*. (2010) NetPath: a public resource of curated signal transduction pathways. *Genome Biol*., **11**, R3.

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

Kelder,T. *et al*. (2012) WikiPathways: building research communities on biological pathways. *Nucleic Acids Res*., **40**, D1301–D1307.

Le Novère,N. *et al*. (2009) The systems biology graphical notation. *Nat. Biotech*., **27**, 735–741.

Martin,D. *et al*. (2004) *OWL-S: Semantic Markup for Web Services*. World Wide Web Consortium.

Nagasaki,M. *et al*. (2009) Foundations of systems biology. *Using Cell Illustrator and Pathway Databases*. Springer, London, p. 16.

Rodchenkov,I. *et al*. (2013) The BioPAX Validator. *Bioinformatics*, **29**, 2659–2660.

Schaefer,C. *et al*. (2009) PID: the pathway interaction database. *Nucleic Acids Res*., **37**, D674–D679.

Wooldridge,M. (2009) *An Introduction to MultiAgent Systems*, vol. **2**. Wiley, Glasgow, pp. 21–38.

Appendix: Formal Specification of BioASF

Inspired by the description logic of BioPAX, we define our multi-agent system model (MASM) as a tuple. Note that here, we follow the mathematical convention to use capital letters to denote a set (e.g. I), and lower-case letters to denote one single element (e.g. i). Besides, we denote all agents with the superscript a (e.g. I^a or i^a). The tuple is defined as follows:

$$MASM = < mp^a, P^a, I^a, e^a, a^a, B, S, E, T, R >$$

where mp^a is the *master pathway agent* which is responsible for bootstrapping the simulation and controlling all top-level *pathway agents*. P^a is a set of pathway agents, each executing its associated pathway and controlling its sub-pathway agents and *interaction agents*. I^a is a set of interaction agents executing interactions. e^a is the *environment agent* maintaining and monitoring the status of physical entities, and a^a is the *analysis agent* responsible for analyzing the new situation that arises as a result of execution of pathways and interactions. A set of biological entities is represented by B. S is a set of sensors which trigger an agent to start its execution, and E is a set of effectors which is the result of the execution of an agent and which can influence the execution of other agents. T is a set of transition rules dictating the execution pattern of an agent. How and whether or not an agent should be executed are determined by these rules. This element of the model allows for model flexibility and extensibility, as model users can provide their own rules. Finally, R is a set of *relations* among pathways, interactions and physical entities.

In line with the BioPAX description logic, the B set can be partitioned as follows:

$$B = G \cup I \cup P \cup H$$

where G is a set of *genes*, I is a set of interactions, P is a set of pathways, H is a set of physical entities. Accordingly, the I set can be partitioned as follows:

$$I = CI \cup VI \cup GI \cup MI \cup TI$$

where CI is a set of control interactions, VI is a set of conversion interactions, GI is a set of *genetic interactions*, MI is a set of *molecular interactions*, and TI is a set of *template reactions*. Furthermore, CI is partitioned as follows:

$$CI = CCI \cup TCI \cup MCI$$

where CCI is a set of *catalysis interactions*, TCI is a set of *template reaction regulation interactions*, and MCI is a set of *modulation interactions*. The set of conversion interactions is also partitioned:

$$VI = AVI \cup BVI \cup DVI \cup TVI$$

where AVI is a set of *complex assembly interactions*, BVI is a set of *biochemical reaction interactions*, DVI is a set of *degradation interactions*, and TVI is a set of *transport interactions*.

Corresponding to each BioPAX interaction type, the model of each agent is formally specified. Each agent is considered as a simple entity having knowledge only about its own managed entity (a BioPAX interaction or pathway) and about its limited relationships with other related agents. Note that there are no agents associated with genes and physical entities in our model. We have specified the formal models of all agent types. Here, for the sake of illustration, we describe the model of pathway agent and catalysis agent.

Each member (p^a) of the P^a element of our multi-agent system model ($MASM$) is also defined as a tuple in which it manages its associated BioPAX pathway (p_p), is triggered by its sensors (S_p), influences other agents by its execution results (E_p), and is governed by its rules (T_p):

$$p^a = < p_p, I_p^a, P_p^a, S_p, E_p, T_p >$$

where $p_p \in P$, $I_p^a \subset I^a$, $P_p^a \subset P^a$, $S_p \subset S$, $E_p \subset E$, and $T_p \subset T$. From the viewpoint of a pathway agent, interactions belonging to its managed pathway are called *internal interactions* and the ones belonging to some other pathway are called *external interactions*. A pathway agent can be triggered by agents managing its internal interactions, sub-pathways, or external control interactions, all together forming the sensor set of the pathway agent (with $i \in I$ and $ci \in CI$):

$$S_p = \left\{ i^a \in I_p^a \,|\, managed(i^a, i) \wedge component(p_p, i) \right\} \cup$$
$$\left\{ p_s^a \in P_p^a \,|\, managed(p_s^a, p_s) \wedge component(p_p, p_s) \right\} \cup$$
$$\left\{ ci^a \in CI^a \,|\, managed(ci^a, ci) \wedge controlled(ci, p_p) \right\}$$

Similarly, a pathway agent can influence agents managing external interactions or external control interactions by playing the role of *participant* or *controller* respectively:

$$E_p = \left\{ i^a \in I_p^a \,|\, managed(i^a, i) \wedge participant(i, p_p) \right\} \cup$$
$$\left\{ ci^a \in CI^a S \,|\, managed(ci^a, ci) \wedge controller(ci, p_p) \right\}$$

Each catalysis agent (cci^a) is defined as a tuple in which it manages its associated BioPAX catalysis interaction (cci_c), is triggered by its sensors (S_c), influences other agents by its execution results (E_c), and is governed by its rules (T_c):

$$cci^a = < cci_c, S_c, E_c, T_c >$$

where $cci_c \in CCI$, $S_c \subset S$, $E_c \subset E$ and $T_c \subset T$. A catalysis agent can be triggered by the environment agent maintaining concentration of its controllers, and by modulation agents managing its modulation interactions (with $h \in H$ and $m \in MCI$):

$$S_c = \{ e^a \,|\, managed(e^a, h) \wedge controller(cci_c, h) \} \cup$$
$$\{ e^a \,|\, managed(e^a, h) \wedge cofactor(cci_c, h) \} \cup$$
$$\{ mci^a \in MCI^a \,|\, managed(mci^a, mci) \wedge controlled(mci, cci_c) \}$$

A catalysis agent can influence agents managing its conversion interactions (with $vi \in VI$):

$$E_c = \{ vi^a \in VI^a \,|\, managed(vi^a, vi) \wedge controlled(cci_c, vi) \}$$