Validation of an Agent Based System Using Petri Nets

Thomas Moncion, Guillaume Hutzler
Lab. de Méthodes Informatiques
Université d'Evry Val d'Essonne
Evry France
{tmoncion,hutzler}@lami.univ-evry.fr

Patrick Amar
Lab. de Recherche en Informatique
Université Paris Sud
Orsay France
pa@lri.fr

Abstract

Hsim is an agent-based simulator that allow the modelling of any kind of macromolecules interactions in a 3D virtual cell using a description language. Theses models which describe reactions between the various molecules are written by the user of the simulator. Our aim is to verify the validity of the user's model and to exhibit properties of the model without needing to run a simulation. To this end, we construct a Petri Net to determine all the possible macromolecular assemblies.

1 Introduction

Agent-Based Simulation (ABS) enables the modelling and simulation of complex systems, in which lots of entities evolve in a common environment and interact with each other. In this framework, each individual entity of the real system is modelled as an agent whose behaviour captures the evolution of the real entity. When placed in a simulated environment, the agents reproduce together the behaviour of the whole system. Lots of complex systems, either natural (physical, chemical, biological, ethological, etc.) or human (economical, sociological, linguistic, etc.) can be modelled and simulated using this technique. As an example, the *Hsim* simulator [Amar *et al.*, 2004] has been designed for the simulation of a virtual cell in which molecules diffuse and interact with each other according to specified chemical reactions.

The validation of the simulator is difficult because choices made with respect to the representation of space (continuous or discrete, rectangular or hexagonal geometry, etc.) or time (continuous or discrete, time step, scheduling of agents, etc.) can greatly influence the corresponding results when applied to a given model. In addition, one has to make sure that the semantics of the modelling language is correctly implemented by the simulator. The validation of the model is also difficult because it generally grasps lots of different entities (chemical species in the case of *Hsim*) and lots of rules describing the behaviour and the interactions of these entities (chemical reactions in the case of *Hsim*). To validate such models in the *Hsim* simulator, it is first important to make sure that the reaction rules included in the model exactly correspond to the prior knowledge about the real system. No two reactions should be incompatible (two given species can react with each other in only one way), and no reaction should have been forgotten or incorrectly added.

The aim of the work that we describe in this article is to provide a precise methodology and the corresponding tools to validate beforehand (without running the simulation) some aspects of an agent-based model and simulator. Our approach consists in transposing the model in an abstract framework where spatial and temporal constraints no longer apply and to fire the reactions virtually to see which complex molecules may be generated by the simulation. As soon as reactions apply, we build a bipartite graph in which places represent the complex molecules that are created and transitions represent the corresponding reactions. The Petri Net semantics and algebra can thus be used to analyse such graphs, which allows to check given properties about the system and its dynamics. Producing or consuming places can be identified and invariants can be checked on sets of places.

We applied our approach to the biological example of an enzymatic cascade, in which a substrate successively participates in several reactions catalysed by different enzymes. In addition, the enzymes can form complexes by reacting together in predefined conditions. Such cascades can be found for example in the glycolysis, which leads to the production of ATP from glucose. By generating the graph of all possible molecules, we identified some of them that should not have been created. After having checked the rules that allowed the production of such "chimera", it appeared that it was due to an incoherence in the simulator, in the application of the rules. Once the simulator patched to eliminate these incoherence, we were able to generate the correct bipartite graph, and analyse which of the molecules were consumed, produced, or preserved by the cascade, thus verifying the biological and chemical knowledge about the global reaction.

In section 2, we present the functioning of the *Hsim* simulator and the corresponding multi-agent model. We detail in section 3 the construction of the bipartite graph of molecules and reactions. In section 4, we

explain how the Petri nets semantics allows to analyse such a graph. We finally discuss the potential of the method and conclude in section 5.

2 Multi-agent model

To describe the movement and the association and dissociation phenomena within a cell, a simulation program has been developed. This simulator, written in C++ and OpenGL, uses a modelling language allowing to describe the molecules involved, the biochemicals reactions and the initial conditions. The program simulates a virtual cell as a three dimensional space surrounded by a membrane. When the simulation begins, the molecules diffuse and interact according to the reaction rules defined with the dedicated modelling language.

2.1 Simulator description

A step of simulation (called a generation) is done by applying the following process:

- Choose the source molecule S (randomly, in order to avoid artefacts).
- Check if close enough to S, in a location randomly chosen L, there is another molecule T, the target.
- If so, and if a reaction rule is given between a molecule of the type of S and a molecule of the type of T, this rule is applied, according to a probability representing the reaction kinetics.
- Else, molecule S may move to the empty location L, according to a probability representing the diffusion speed.

When all the molecules involved in the simulation are processed, the current generation is finished and a new one can begin. The generation simulated time slice is set to 100 microseconds, which corresponds to the average time for a protein to move a distance of 10 nanometres (approximately its diameter).

2.2 Rules and configuration

The simulator is designed to be independent of a particular model. A language has been developed to describe the interactions rules which can exist between the various agents involved in the simulation.

This language describes four possible types of interaction between two molecules S and T.

- Reaction: the molecule S reacts with the molecule T to produce two new molecules S' and T'.
- Association: the molecule S binds to the molecule T to form the complex S-T.
- Dissociation: a complex S-T can dissociate and release the molecules S and T
- Catalysis: a complex S-T can be transformed into another complex S'-T'

The association and dissociation rules can also change the molecule type of the products. Each rule is assigned an execution probability which corresponds, in the long-range, to the kinetic of the reaction. The maximum number of bonds between each type of molecules can also be set.

The first section of the configuration file describes the molecules involved in the simulation:

```
molecule A, B, C, D;
```

The second section specifies the diffusion speed for each molecule.

```
speed(A) = 0.0;
speed(B) = 1.0;
speed(C) = 0.0;
speed(D) = 0.2;
```

In this example, only molecules B and D can diffuse. Molecules A and C do not diffuse.

The third section specifies the reaction rules between molecules. We can have:

- Association rules

  ```
  A + B -> A(1)* B(1) [0.6];
  ```

  This rule means that when a molecule of type A is close enough to a molecule of type B, they form the complex A-B with probability 0.6. The numbers between brackets in the right part of the rule mean that molecule A (respectively B) cannot bind to more than one molecule B (respectively A).

  It is also possible to restrict the scope of a reaction to already bound (or unbound) molecules. We can take again the reaction of the previous example and trigger the reaction only if the molecule of type A is already bound to a molecule of type C, and if the molecule of type B is not bound to a molecule of type D.

  ```
  {C}A + {~D}B -> A(1)* B(1) [0.6]
  ```

- Dissociation rules

  ```
  {~C}A * B -> A + B [0.01]
  ```

  This rule means that the binding of molecule B to molecule A can be cleaved with probability 0.01 (only if the molecule of type A is not also bound to a molecule of type C).

- Reaction rules

  ```
  A + B -> C + D [0.5]
  ```

  This rule shows that two molecules of type A and type B can react, the molecule of type A being transformed to a molecule of type C (resp. the B being transformed to a D).

- Catalysis rules

  ```
  A * B -> C * B [0.9]
  ```

  This rule shows that a complex of type A-B can be transformed to a complex of type C-B with proba-bility 0.9.

With this set of rules, it is thus possible to simulate the formation of molecular assemblies in a cell.

The last part of the configuration file specifies the initial population for each species (the number of

copies of each type of molecules and their localisation in the cell).

```
surface (A, 6);
cube (1,4,3,4,B);
cube (6,5,0,2,C);
cube (2,11,4,5,D);
```

The first line means that 36 copies of molecules of type A are located in the membrane. In the other lines, the first three numbers specify the coordinates x, y, z of a cube containing the molecules. The fourth number gives the size of the edge of the cube. In the second line, the number 4 indicates that we have $4^3 = 64$ copies of molecules of type B.

3 Determination of the whole structure

Since there is a language describing the biochemical reactions between the molecules, we can compute the complete set of all the assemblies that can possibly be made. For this purpose, we have designed an algorithm to build a *Petri net* in which the places represent the different molecular species (single molecules and assemblies) and the transitions represent the biochemical reactions.

3.1 Construction algorithm

Let $G(P, T, A, M_0)$ a Petri net where P is a set of places, each one uniquely labelled by an assembly S_i: $P = \{p_1^{S_1}, \ldots, p_i^{S_i}, \ldots, p_t^{S_t}\}$. The elements of the set P will be ordered, it will be seen as an ordered vector of places. T is the set of the transitions of the net, $T = \{t_1^r, \ldots t_j^r, \ldots t_u^r\}$ where each transition is labelled by a reaction $r \in R$, R being the set of all the reactions. Finally, A, the set of arcs, is splitted in two subsets, the arcs from the places to the transitions: A^- and the arcs from the transitions to the places: A^+. So we have $A = A^- \cup A^+$ with $A^- = \{(p_i, t_j)\}$ and $A^+ = \{(t_j, p_i)\}$. In this section we do not consider the initial marking M_0; its use will be seen in section 4.1.

The assemblies S_i are represented as a graph (X, L) with $X = \{x_1, \ldots, x_j, \ldots, x_n\}$ where the x_j represent the single molecules and $L = \{(x_l, x_m)\}$ represent the bindings between the molecules. The set $R = \{r_1, \ldots, r_k, \ldots, r_p\}$ is the set of all the reactions, each reaction being represented by *r=[e1]mol1 <op> [e2]mol2 → res1 <op> res2.*

To understand how this algorithm works we will explain it using a simple example: a cascade of two enzymes, $e1$ which catalyses the transformation of the initial substract $s1$ to $s2$, and $e2$ which catalyses the substract $s2$ giving the final product $p3$. We will suppose that the whole reaction is *channelised* by the self assemblied complex $e1 - e2$. Here are the rules used to model this example:

```
r1: s1 + e1 -> s1(1) * e1(1) [0.6];
r2: {s1}e1 + e2 -> e1(1) * e2(1) [0.9];
r3: {~s1}e1 * e2 -> e1 + e2 [0.0001];
r4: {e1}s1 + {e1}e2 -> s1(1) * e2(1) [1.0];
r5: {e2}e1 * {e2}s1 -> e1 + s2 [1.0];
r6: s2 * e2 -> p3 + e2 [0.9];
```

The places p_1, p_2 and p_3 which represent the initially provided single molecules $e1$, $s1$ and $e2$ are inserted into the vector P. The initial Petri net is shown in figure 1 while the final vector P showing the mapping between the places and the chemical species is in figure 5 (at this point of the explanation we consider only the three first entries). The algorithm will build step by step both the vector P (the places) and the set A (the transitions), giving the Petri net.

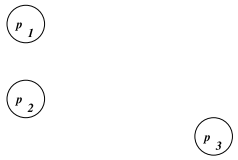

Figure 1: Initial Petri net.

P={initial places} (provided molecules);
forall *i from 1 to* $|P|$ **do**
  forall *j from 1 to* $|P|$ **do**
    forall *k from 1 to* $|R|$ **do**
      let r_k **such as**
      ($(r_k = \{e1\} x_{im} < op > \{e2\} x_{jn} \rightarrow \ldots \in R)$ *and* $(t^{r_k} \notin T)$ *and* $((p_i, t^{r_k}) \notin A)$ *and* $((p_j, t^{r_k}) \notin A)$)

      Add new places in P**:**
      $\begin{cases} if\ 1\ product: \\ p_{|P|+1}\ if\ p_{|P|+1} \notin P \\ if\ 2\ products: \\ \begin{cases} p_{|P|+1}\ if\ p_{|P|+1} \notin P \\ p_{|P|+2}\ if\ p_{|P|+2} \notin P \end{cases} \end{cases}$
      Add transition $t^{r_k}_{|T|+1}$
      Add new edges in A^-**:**
      $\begin{cases} if\ p_i = p_j: \\ (p_i, t^{r_k}_{|T|+1}) \\ if\ p_i \neq p_j: \\ \begin{cases} (p_i, t^{r_k}_{|T|+1}) \\ (p_j, t^{r_k}_{|T|+1}) \end{cases} \end{cases}$
      Add new edges in A^+**:**
      $\begin{cases} if\ 1\ product: \\ (t^{r_k}_{|T|+1}, p_{|P|+1}) \\ if\ 2\ products: \\ \begin{cases} (t^{r_k}_{|T|+1}, p_{|P|+1}) \\ (t^{r_k}_{|T|+1}, p_{|P|+2}) \end{cases} \end{cases}$
    end
  end
end

The main process is done by sweeping the vector P from the first to the last entry. Starting from the initial set of places, the vector is built by appending new entries at the end. So let's consider the first entry p_1 labelled by $e1$; The reaction r_1 can be applied using the molecule $s1$ labelling the place p_2. This reaction builds a new complex $e1s1$, as this is a new molecular species a new entry p_4, labelled by $e1s1$, is appended at the end of the vector. The new node $t_1^{r_1}$ is put into the set of transitions and the corresponding arcs are

added to update the Petri net (cf. figure 2).

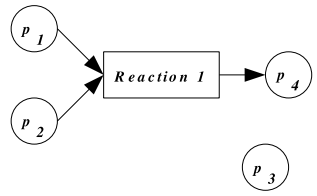

Figure 2: First transition of the Petri net

As no other reaction can be applied to the molecular species labelling the place p_1, the algorithm goes to the next entry of the vector P i.e. p_2 labelled by $s1$. The only possible reaction with $s1$ has already been used, r_1 giving the complex $e1s1$; As both this molecular species labels an entry in the vector, and this entry has been created using this same reaction, nothing more is done.

The process continues with the third entry of the vector, p_3 labelled by $e2$. The reaction r_2 can be applied to $e2$ and $e1e2$ building a new complex $e1s1e2$, this complex labelling a new entry p_5 in the vector. As previously, a new node is put into the set of transitions and the corresponding arcs are added to update the Petri net (cf. figure 3).

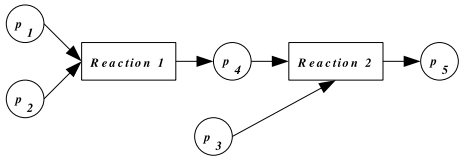

Figure 3: Second transition of the Petri net

The process continues with the next entry, the place p_4 labelled by $e1s1$, which will lead to nothing more as the only possible reaction, r_2 has already been used with this assembly. Then we go to the next entry, the place p_5 labelled by $e1s1e2$. At this point, the only possible reaction is r_4 which applies *inside* the complex $e1s1e2$ leading to a third binding, between $s1$ and $e2$, and thus to a new place p_6 and a new transition $t_3^{r_4}$.

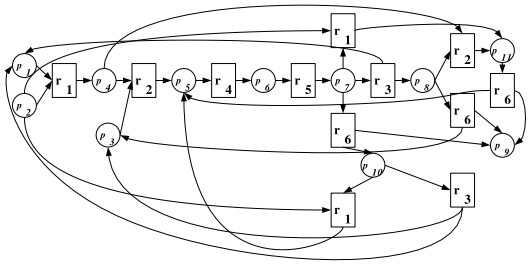

Figure 4: Final Petri net. The chemical species are shown in figure 5.

This algorithm continues until the last entry in the vector P has been processed, the Petri net being incrementally built (cf. figure 4).

One can notice that at each step of the process, all the possible reactions between every molecule of the *current* assembly and every molecule belonging to all the assemblies already exhibited are considered, so we are certain to avoid missing any product at all. Of course this version of the algorithm finishes only if the chemical species made are in a finite number. This assumption is clearly true when no recursive rule is present in the model.

By simply observing the final Petri net, we can notice that the place p_2, labelled by the molecule $s1$, has only outgoing arcs while the place p_9, labelled by the molecule $p3$, has only incoming arcs. This shows that the molecules of type $s1$ are consumed (and can only be consumed) while molecules of type $p3$ are produced (and only produced) by the whole system.

P1	P2	P3	P4	P5
e1	*s1*	*e2*	*e1* | *s1*	*e1—e2* | *s1*
P6	P7	P8	P9	P10
e1—e2 | *s1*	*e1—e2* | *s2*	*e2* | *s2*	*p3*	*e1—e2*
P11				
e1—e2 | | *s1* *s2*				

Figure 5: Vector of the places P_i of the Petri net and the corresponding chemical species.

3.2 Detection of errors in the simulator

The construction of such Petri nets from the reactions provided by the modeller has shown the possible production of "chimeras" generated by the first version of *Hsim* program.

This was due to a bug in the simulation program: the maximum number of bonds between each type of molecules was not correctly checked. Obviously the current version of the simulator has no longer this bug!

4 Properties of the network of biochemical reactions

4.1 Petri Nets

Definition

A Petri Net (PN) is a bipartite directed graph represented by $G(P, T, A, M_0)$ where:

- $P = \{p_1, p_2, p_3, \ldots, p_l\}$ is a finite set of places, which are represented by circles. In our case, these places correspond to the resources of the system. A place can thus contain several tokens.
- $T = \{t_1, t_2, t_3, \ldots, t_m\}$ is a finite set of transitions, which are represented by rectangles. A transition corresponds to an event of the system.

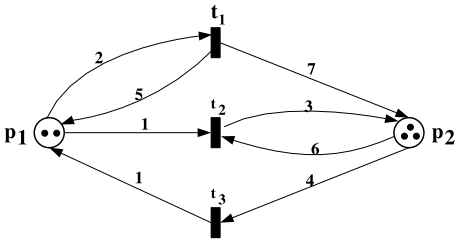

Figure 6: Petri Net with 2 places p_1, p_2 and 3 transitions t_1, t_2, t_3. Initial marking M_0 is [2, 3]

- $A = \{a_1, a_2, a_3, \ldots, a_n\}$ is a finite set of directed edges where $A = \{A^- = P \times T \mapsto \mathbb{N}\} \cup \{A^+ = T \times P \mapsto \mathbb{N}\}$ corresponding to the set of directed edges from places to transitions or from transitions to places.
- $M_0 = \{m(p_1), m(p_2), \ldots, m(p_l)\}$ is an initial marking. This marking specifies the number of tokens in each place for the initial state of the system.

In a Petri net, if an edge is directed from place p to transition t, we say that p is an input place for transition t. An output place is defined similarly. If every input place for a transition t has at least one token, we say that t is enabled. A firing of an enabled transition removes one token from each input place and adds one token to each output place.

Matrix representation

It is possible to build a matrix representation of a Petri net. We consider two applications Pre and $Post$ such as:

- $Pre = P \times T \mapsto \mathbb{N}$ is the pre-incidence application, where $Pre(p_i, t_j)$ is the weight of edge (p_i, t_j). $Pre(p_i, t_j) > 0$ if the edge exists, $Pre(p_i, t_j) = 0$ otherwise.
- $Post = T \times P \mapsto \mathbb{N}$ is the post-incidence application, where $Post(p_i, t_j)$ is the weight of edge (t_j, p_i). $Post(p_i, t_j) > 0$ if the edge exists , $Post(p_i, t_j) = 0$ otherwise.

If we take the example of the network of figure 6, we have the two following matrixes Pre et $Post$:

$$Pre = \begin{array}{c} \\ p_1 \\ p_2 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{vmatrix} 2 & 1 & 0 \\ 0 & 6 & 4 \end{vmatrix} \end{array} \quad Post = \begin{array}{c} \\ p_1 \\ p_2 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{vmatrix} 5 & 0 & 1 \\ 7 & 3 & 0 \end{vmatrix} \end{array}$$

We then have a matrix C which we call the incidence matrix, which is defined as follows:

$$C = Post - Pre$$

If s is a firing sequence associated with a firing vector $\bar{s}$, where each element of $\bar{s}$ is the number of times the corresponding transition fires in the sequence s, the marking M' reached from the initial marking M after the firing of s is given by:

$$M' = M + C.\bar{s}$$

where C is the incidence matrix

4.2 Properties of the Petri Net

In this section we focus on the properties of the Petri net obtained in section 3.1. As we have already noticed in section 3.1 the observation of this network immediately shows that some structures can only be produced (no outgoing edges) or consumed (no incoming edges). This simple observation however is not sufficient, which justifies the formal study of the properties of the PN so as to:

- confirm the validity of a model;
- show the principal properties of the reaction network;
- study the dynamics of the system.

Several works [Voss *et al.*, 2003] [Zevedel-Oancea and Shuster, 2003] show the interest of this type of study within the framework of metabolic networks.

P-invariants

P-invariants are vectors of places, which we note y. The multiplication of the transpose of y with any marking is identical to the multiplication with the initial marking:

$$y^T.M = y^T.M_0$$

Vector y thus describes a conservation relation of markings. By taking into account this conservation relation, it comes:

$$y^T.C = 0$$

where C is the incidence matrix. These relations imply that, for any two place invariants that we note I_1 and I_2, we have:

$$I_1 + I_2 = const \implies c_1 I_1 + c_2 I_2 = const$$

where c_1 and c_2 are natural integer.

The essential property of P-invariants is that the weighted sum of the tokens associated to the vector is constant whatever the evolution of the PN.

If we come back to the example of the PN shown in section 3.1, it is then possible to show that all the places containing either $e1$ or $e2$ are invariants. Translated into biological vocabulary, this means that whatever the evolution of the system, the enzymes will never be consumed. If we hadn't detected these P-invariants for the enzymes $e1$ and $e2$, this would have been the sign of a problem in the model.

Dynamics of the Petri net

The study of the dynamics of the PN corresponds to the observation of the evolution of the tokens in this PN. This allows to:

- determine the molecules that are produced and consumed;
- observe the consequences in the evolution of the system when some resources are suddenly missing.

Let's take the example of the PN of section 3.1 with the initial marking $M_0 = \{m(p_1) = 1, m(p_2) = 4, m(p_3) = 1\}$. The study of the evolution of the PN

shows us that at the end (i.e. when no more transitions are enabled), all the tokens initially present in p_2 can be found in p_9, with one token left in each of places p_1 and p_3.

By considering the simple case where $M_0 = \{m(p_1) = 1, m(p_2) = 4, m(p_3) = 0\}$, the evolution of the system will stop at the level of p_4. Indeed, the lack of the enzyme $e2$ prevents the production of p_3.

5 Discussion

The *Hsim* simulator is not dedicated to specific types of macromolecular assemblies. On the contrary, it's a multi-purpose simulation kernel that enables the modelling and simulation of any type of macromolecular assemblies, provided that they can be described as the result of elementary biochemical reactions between molecules.

We demonstrated in this work that, even if the main interest of agent-based modelling lies in the simulation of emerging spatio-temporal structures, it still may be useful to develop validation tools that allow to explore some properties of models without even running the corresponding simulations. On the one hand, it is necessary to verify that the semantics of the modelling language is correctly implemented by the simulator. On the other hand, it is necessary to check the validity of a model defined by the modeller. We proposed a method based on the construction of bipartite graphs of macromolecular assemblies and reactions, and on the use of Petri nets algebra to make a number of verifications. This allowed to exhibit a problem in the firing of the reactions by the simulator, which has since been corrected. This also enabled to verify that well known properties of a given system may be verified (conservation of enzymes for example).

In the example shown in the article, the biological properties that we exhibited were simple and could have been noticed by the simple observation of the reaction network. As the number of reactions increases however, the number of possible structures can become very important. It then rapidly becomes impossible to study visually the properties of the model, which justifies the automated techniques that we proposed.

However, a restriction of our method is that the construction algorithm only terminates if assemblies produced by the model are of limited size. When we take potentially infinite structures like polymers, this method is not valid any more. Further investigations are then necessary to identify potential recursivities in the reaction networks, either by a preliminary study of the reactions or by the identification of similar subparts in the graph.

In addition, our method has proven to be useful for the validation of models but it may also be useful to characterise the dynamics of agent-based models and automatically identify emergent phenomena such as spatial and temporal structures. The first step would be to automatically identify, in the Petri net, such interesting properties as P-invariants, loops, consumed or produced assemblies etc. In our example, we defined the P-invariant that we wanted to verify (corresponding to the knowledge that enzymes are not transformed by a reaction). It would be best if such P-invariants could be found automatically by the validation tool, which would enable the modeller to see directly all the invariants in his or her model. The second step would then be to identify, in the reaction network, subnetworks that may be considered as higher-level structures in the simulation. To be able to take into account such higher-level structures, it is necessary to extend the modelling language so as to allow the definition of reactions between structures at different levels. It will also be necessary to study the behaviour of these higher-level structures in the dynamic context of the simulation, so as to reintegrate space and time factors in the characterization of their behaviour. By integrating our validation tool in the simulator, the long-term objective is thus to enable the dynamic characterization of emergent structures, which may lead to the dynamic design of multi-level agent-based models and simulations.

References

[Amar *et al.*, 2004] Patrick Amar, Gilles Bernot, and Victor Norris. Hsim: a simulation programme to study large assemblies of proteins. *Journal of Biological Physics and Chemistry*, 4:79–84, 2004.

[Voss *et al.*, 2003] K. Voss, M. Heiner, and I. Koch. Steady state analysis of metabolic pathways using petri nets. *In silico Biology*, 2003.

[Zevedel-Oancea and Shuster, 2003] I. Zevedel-Oancea and S. Shuster. Topological analysis of metabolic networks based on petri net theory. *In silico Biology*, 2003.