Validation of an Agent Based System Using Petri Nets

Thomas Moncion
Laboratoire de Méthodes Informatiques
Tour Evry 2 523 Place des terrasses de l'agora
91000 Evry Cedex France
tmoncion@lami.univ-evry.fr

Guillaume Hutzler
Laboratoire de Méthodes Informatiques
Tour Evry 2 523 Place des terrasses de l'agora
91000 Evry Cedex France
hutzler@lami.univ-evry.fr

Patrick Amar
Laboratoire de Recherche en Informatique
Bât 490 Université Paris-Sud
91405 Orsay Cedex France
pa@lri.fr

ABSTRACT

Hsim is an agent-based simulator used to model the interactions of macromolecules in a 3D virtual cell surrounded by a membrane. This simulator is not dedicated to a specific model of macromolecular interactions but allows the modelling of any kind of interactions, described using a modelling language, written by the user of the simulator and representing reactions between the various molecules, including the formation of complexes.

Our aim was 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 and use the Petri Net algebra to verify chosen properties.

Keywords

agent-based simulator, Petri Net

1. MULTI-AGENT MODEL

To describe the movement and the association and dissociation phenomena within a 3D virtual cell surrounded by a membrane, a simulation program has been developed in C++ and OpenGL[1].

The simulator is designed to be independent of a particular model. A language has been developed to describe the four possibles types of interaction which can exist between the various agents involved in the simulation. We have:

- Association: the molecule A binds to the molecule B to form the complex A-B.
  Example: `A + B -> A * B [0.6]`
- Reaction: the molecule A reacts with the molecule B to produce two new molecules C and D.
  Example: `A + B -> C + D [0.5]`
- Dissociation: a complex A-B can dissociate and release the molecules A and B
  Example: `A * B -> A + B [0.01]`
- Catalysis: a complex A-B can be transformed into another complex C-B
  Example: `A * B -> C * B [0.9]`

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

2. PETRI NET CONSTRUCTION AND PROPERTIES

2.1 Petri Net construction

Since there is a language describing the biochemical reactions between the molecules, we can compute the complete set of all the assemblies which can possibly be made. For this purpose, we have designed an algorithm to build a Petri Net in which places correspond to macromolecular assemblies and transitions correspond to reactions.

Let $G(P, T, A, M_0)$ a Petri Net (PN) where:

- $P = \{p_1, \ldots, p_i, \ldots, p_t\}$ is a set of places
- $T = \{t_1, \ldots t_j, \ldots t_u\}$ is a finite set of transition
- $A = \{a_1, a_2, a_3, \ldots, a_n\}$ is a finite set of directed edges.
- $M_0 = \{m(p_1), m(p_2), \ldots, m(p_l)\}$ is an initial marking.

To understand how the 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 assembled complex $e1 - e2$. Here are the rules used to model this example:

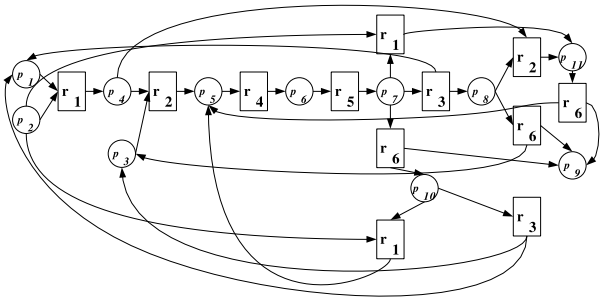
```
r1: s1 + e1 -> s1(1) * e1(1) [0.6];
r2: {s1}e1 + e2 -> e1(1) * e2(1) [0.9];
r3: {~s1}e1 * e2 -> e1 + e2 [0.0001];
r4: {e1}s1 + {e1}e2 -> s1(1) * e2(1) [1.0];
r5: {e2}e1 * {e2}s1 -> e1 + s2 [1.0];
r6: s2 * e2 -> p3 + e2 [0.9];
```

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 algorithm will build step by step both the vector P (the places) and the set A (the transitions), giving the PN.

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 is put into the set of transitions and the corresponding arcs are added to update the PN.

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 $s1$. This algorithm continues until the last entry in the vector P has been processed, the Petri net being incrementally built (cf. figure 1).



p_1	p_2	p_3	p_4	p_5	p_6
e1	*s1*	*e2*	*e1* | *s1*	*e1—e2* | *s1*	*e1—e2* | *s1*
p_7	p_8	p_9	p_{10}	p_{11}	
e1—e2 | *s2*	*e2* | *s2*	*p3*	*e1—e2*	*e1—e2* | | *s1* *s2*	

Figure 1: Final Petri net.

2.2 Properties of the Petri Net

By simply observing the final PN, we can notice that the place p_2 (molecule $s1$) has only outgoing arcs while the place p_9 (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.

This simple observation however is not sufficient, which justifies the formal study of the properties of the PN. This study allows us to confirm the validity of a model, to show the principal properties of the reaction network and to observe the dynamics of the system.

2.2.1 P-invariants

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

- $Pre = P \times T \mapsto \mathbb{N}$ 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.

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

$$C = Post - Pre$$

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. It then comes:

$$y^T.C = 0$$

These relations imply that, for any two place invariants that we note I_1 and I_2, we have:

$$I_1 + I_2 = const \implies c_1 I_1 + c_2 I_2 = const$$
(c_1, c_2: 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.

2.2.2 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 and to observe the consequences in the evolution of the system when some resources are suddenly missing.

3. DISCUSSION

The *Hsim* simulator is able to model and simulate any kind of macromolecular assemblies 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. It is necessary to check the validity of a model defined by the modeller. We proposed a method based on the construction of a Petri Net of macromolecular assemblies and reactions. This enabled to verify that well known properties of a given system may be verified (e.g. conservation of enzymes). It also enabled to identify unknown bugs with the simulator leading to abnormal molecular assemblies.

4. REFERENCES

[1] P. Amar, G. Bernot, and V. Norris. Hsim: a simulation programme to study large assemblies of proteins. *Journal of Biological Physics and Chemistry*, 4:79–84, 2004.