Agent-based framework to simulate metabolic processes

M. Beurton-Aimar[1] , N. Parisey[2]

[1]LaBRI - UMR 5800 Univ. Bordeaux 1.
[2]Laboratoire Physiopathologie Mitochondriale - U688 - Univ. Bordeaux 2.

Abstract

Since the last ten years, simulations of living systems have had new perspectives due to the inflow of biological data available on the one hand and the increase of computer amount in term of calculus and memory storing on the other hand. So simulation of biological processes can now be considered for large sets of molecules not only with differential equations but also by modelling molecules independently as with agent-based systems. In a large project about the studying of the mitochondrial metabolism, MitoScoP[1], we have designed a generic agent-based framework to test molecules behaviours where considering mean behaviour (like with differential equations) is not well adapted. The implementation of this framework is done in Lisp and uses the McClim interface for the graphical simulations.

Introduction

For several years, agent-based systems have been used for studying population biology/ecology like predator/prey problems or epidemiological researches like viral diffusions. Theses models are able to represent population level of organisation but often deal with a small number of types even if there are many instances of each. In another context, artificial chemistry tries to reproduce in silico molecule behaviours at the atomic level. Similar agent-based frameworks can be used to simulate ecology or chemistry and there are several examples of such frameworks which are used for biological applications.
Metabolic processes are characterised by complex arrangements of heterogeneous entities. So modelling these processes at the atom level is out of our reach and often not informative. A more appropriate abstraction level allows to represent a complex system with the right level of description[7]. Agent-based modelling driven by object oriented approach is a powerful tool for such design. We have studied several enzymatic reaction chains and built a generic framework using reactive agents to model molecules. We present a specific one

to design oxidation/reduction (redox) reaction chain which occurs inside the mitochondrial respiratory chain[9]. After a quick presentation of the biological context of our project, we will describe the agent model that we have designed and how it is used to simulate the redox process which is one of the main processes involved in metabolism. Finally we will discuss the advantages and the limitations of this approach.

1 Modelling Metabolic Processes

Cell functions, like self-replication or energy generation, are the result of the work of metabolic pathways. Metabolic pathways are sets of enzymatic reactions involving reacting molecules (also called metabolites) and catalysing macromolecules (called enzymes). As these pathways are strongly connected, the whole organism metabolism is a complex network often compared to "small world network". Organelles, like mitochondria, have their own metabolism made of ten to twenty main connected functions (pathways). Modelling such set of biological processes means to take into account more than fifty different types of entities: metabolites or enzymes, each of them being present in several hundred copies. But the working hypothesis that there is an homogeneous repartition of the molecules inside the mitochondrial volume is not always verified and many examples of different results between mean behaviours and local calculation are reported [3][8].
Agent-based modelling allows to take into account localisation of the molecules individually and to compute peer-to-peer interaction easily. If we examine the case of a simple enzymatic reaction, the basis process is:

- if two molecules, an enzyme and a metabolite, are spatially close enough, they can interact with each other.
- This interaction can be resumed by a chemical transformation of the metabolite - the substrate - into another one - the product.
- At the end, the product is released and the enzyme is free to react again with a new substrate.
- Of course the new product can react another time with another enzyme and so on.

Agent-based design is a bottom-up approach very similar to the object oriented approach and from now we will assume that agents will be considered as objects in OO modelling. The different steps of modelling a biochemical process can be described as:

- definition of the agent classes - reification step.
- description of the interaction rules between agents.
- description of the life cycle for all agents.

The life cycle in agent-based models is controlled by an algorithm which defines the ordered list of operations that must be applied at each simulation step.

1.1 Agent model

The choice of the abstraction level for the agent model drives the choice of object classes. In molecular biology the natural level of description is the molecule level and as it is described above enzymatic reactions involves two types of molecules: enzymes and metabolites. These two kind of entities share some characteristics: molecular weight, spatial position, several details of description like databank ID, bibliography references .... These characteristics are put into a super class called `BioAgent`. Enzyme and metabolite can be considered as catalysers and reacting units. So this defines two subclasses of `BioAgent` which are `Catalyser` and `Reacting-Unit`. The difference between these two classes is that only the `Catalyser` needs to know the reaction rules and that `reacting-unit` instances are simply under the influence of a "general" brownian movement.
These two subclasses are sufficient to express generic enzymatic reaction. To simulate specific enzymatic processes it is needed to dynamically load descriptions of different types of catalysers or reacting-units. In this case, the Lisp property to easily generate new classes from data files allows us to obtain all the needed classes *"without additional cost in the code"*.
This is an example of the simple macro which permits to generate all `catalyser` classes by reading their names inside data file:

```
;;;Generic method to define biological type for enzyme
(define-bioagent-type catalyser (reacting-unit)
  ((active-sites :initarg :active-sites :accessor active-sites)))
```

```
;;; Creates an agent class corresponding to a biological type
(defmacro define-bioagent-grid (bioagent-type slots)
  `(progn
    (defclass ,(intern (find-agent-name bioagent-type))
        (,bioagent-type bioagent-grid) ,slots)
    (execute-when-defined (make-instance
                           (find-agent-symbol ',bioagent-type)))))
```

The `biological-grid-agent` is a BioAgent placed in a 3D space.
The same kind of macro is done for the `reacting-unit`.

Interacting rules are organised as a list of accepted molecules for the reaction, so for each catalyser a list of reacting-units which are accepted to react is done.

Random encounters with non reacting molecules are ignored. The list is part of each catalyser class description and is loaded from the data file too.

1.1.1 Generic molecule behaviours

In biochemical simulation, all molecules are subject to brownian movement. Brownian movement is a generic and "general" law that happen to all molecule. Brownian movement is one of many molecules behaviour which depends upon forces that occurs at the molecular level. These forces are usually summarised in term of generic and "general" functions. These generic functions are attached to the BioAgent and parametrised by attributes like molecular weights of the different subclasses.

1.1.2 Life Cycle description

Multi-Agent system modelling supposes to give the list of actions to perform for each agent for one step of simulation. At the higher level one must decide if the system is synchronous or asynchronous. Many algorithms are based on the guaranty that all the agents are visited before switching to the next step of simulation. This kind of constraints is easily implemented with `before` or `after` methods in Common Lisp functions. In general, first order functions can be used to describe some characteristics concerning the whole system behaviour like the well-known sequence of actions in agent system : perception (find neighbouring), decision (try to find something to do), action (move, fire the reaction ...). This sequence is the generic algorithm applied to each agent at each simulation step. An agent visited during the life cycle must check its neighbouring to know if some reactions must be fired. This task is devoted to the catalyser agents and takes into account reacting-unit list defined by the catalyser agent type.
One of the interest of living system simulations is to observe for a long time the system behaviour and to interact dynamically with it, so good level of interaction between users and the program is crucial. We want to be able to disturb locally the simulation, to reproduce pathological event which can happen, or in the contrary to correct some undesirable phenomena as troubles induced by drugs reactions. This is the second reason why we have chosen the Lisp environment to develop our framework. The lisp interacting loop allows the user to suspend simulation, to modify dynamically the program sequence, to analyse the system state if it breaks down and to resume the simulation after some corrections.

1.2 Graphic interface for simulations

One of the main goal of multi-agent system is the observation of emergent phenomena so graphic interface is required in order to observe the whole behaviour of the system. The graphic toolkit McClim[11] (as Clim) supplies a collection of `presentation` functions to program the interfaces which limits massively code size to write down. Our framework has a simulation engine module without any reference to the graphic interface and a module to manage graphics. This last

one only contains less than 600 lines of code to describe the whole interface: menu, buttons, drawing window to show molecule behaviours and statistic window to observe concentration evolution. There is no doubt that this is a very *compact* code. For example, a similar interface for showing interactions between phospholipides (with only one type of agents) built in Java language by one of our collaborators (with Java3d library), has required more than 1200 lines of code.

2 The example of the redox reactions

We have used our agent framework to model redox reactions involved into a specific mitochondrial pathway: the respiratory chain. These redox reactions take place inside the Complex III enzyme. A redox reaction is characterised by an electron exchange between two partners according to qualitative and quantitative rules. A redox reaction can be divided into two parts: reduction and oxidation. Reduction is the gain of electrons whereas oxidation is the lose of electrons. To be precise, redox reactions correspond to changes in the oxidation number that may not involve electrons (involving covalent bonds for instance).

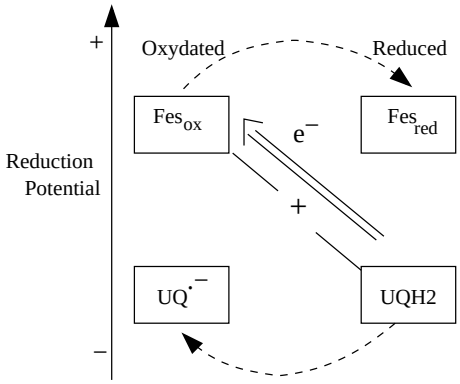

Figure 1: electron exchange between two molecules

Each molecule involved in a redox reaction is considered either in a reduced state or in an oxidised state depending on the energy level separating it from its stable state. This stable state is reached by giving or getting electrons. In the case described by the figure 1, a molecule of iron, noted `FeS`, in oxidised state meets an `Ubiquinone` molecule in reduced state - in this case this molecule is called `UQH2`. The electron release/capture causes the state change of the two partners: `FeS` becomes reduced and `Ubiquinone` oxidised i.e. $\texttt{UQ}^{\bullet -}$. The rules applied to fire this reaction are:

- **qualitative rule:** one molecule must be in oxidised state and the other in reduced state,

- **quantitative rule:** the two molecules must exhibit a positive difference of their reduction potential. The reduction potential is a value that gives the energy needed to reach the stable state.

There exists reduction scales which give for each type of molecule the thresholds (reduction potentials) at which this molecule switches its state from a reduced state to an oxidised one. One can note that some type of molecules can have several reduction potentials. This is the case of Ubiquinone as we will explain in the next paragraph. A redox reaction occurs if the oxidised molecule has a higher reduction potential than the reduced one.

An agent-based system based on *reactive agents* is a simple way to simulate metabolic processes like redox reactions. We will now present the retained model to design such process and the rules used for simulation.

The agent-based model which has been designed uses a 3D continuous space where the agents are located. The space and the distance between agents are used in interactions evaluation. All the agents have a local perception of their neighbourhood. A grid divides the space into neighbouring units which simplify interactions computations.
The model is called *open* because of the variable number of agents during simulation. The continuous degradation and synthesis of molecules is a common biological process. Since molecules are driven by chemical laws, the agents of our model are reactive agents and not cognitive ones. As we need to take into account several molecules the agent system is heterogeneous.

3 Results and discussion

This work reports an experience of using Lisp in a domain where it is not really frequent to encounter it: graphic simulation of dynamic processes. At the prototyping step, we have chosen this language because it seems to be well suited to our needs: code easy to modify dynamically, easy to maintain (shorter code) and set of tools like graphic libraries easy to use. Moreover in Lisp we can rely on an efficient object oriented model (with CLOS [6][10]) which is a very important element for research projects where many modifications and additions are continually made. In that way, using interactive language allows us to quickly write (and to quickly test) new modules.
The whole project MitoScoP contains also a databank to store knowledge and a web site to give access to the databank. We have begun to develop a framework with pathway tools [5] based on the Metacyc model[2] all written in Lisp.
An another MitoScoP module must contain mathematical equations to simulate metabolism processes, mainly with differential equations, As this set of equations requires only classical equation solver, there is no more problem to use Lisp unless the misgivings of mathematicians more accustomed to program in Fortran language. To summarise, we can tell that this experience is conclusive and there is no doubt that we will continue in that way.

Now however we need to try to develop more functionalities in our framework and we start to test the limits of the different tools we have used. Graphic displayed must work with object in 3D space, we want to use OpenGL library in order to do that. Of course it exists some bindings between Lisp and OpenGL but not McClim. We participate to a project of writing a graphic engine in CommonLisp and using OpenGL but it remains many things to do. In the next months, we want firstly to test the OpenGL binding libraries and secondly to work on the bindings between McClim and OpenGL.
Another question is the interactions between the simulation program and the user. During a simulation, the Lisp loop has always something to do, i.e. to apply the life cycle to all agents. So it is difficult to listen if the user asks to do something, for example to interrupt the program, to change a parameter .... Using threads to implement that is possible but management of threads is not completely standardised through the different architectures (Linux or Mac OS) and implementation of Lisp machine. But one can note that this problem is also encountered with another programming languages.
If we summarise this experience we can note that the greatest barrier to the use of Lisp in our project is not the lack of Lisp tools but the non familiarity of the different participants to this language. Into interdisciplinary projects, the non-informaticians can have some knowledge in programming but only for the most popular ones (in general Fortran or C) and so there are many resistances to learn new one. In another way, there also exist several frameworks to develop simulation examples, a very popular one in Multi-Agents community is NetLogo [4]. It is able to build 3D simulations but as agent descriptions are done in a specific and very simple language, complex descriptions stay out of reach in NetLogo.
In conclusion despite the difficulties that we have noted, we will continue to develop our Lisp framework because we can dispose of all the essential functionalities and the quality of the produced code keeps the maintenance charge at low level.

4 Acknowledgement

This work was supported by the ACI MitoScoP. We extended our thanks to Alexandre Bonnet, Florent Collot, Guillaume Labat, Fabrice Laporte for their assistance in the development of Basil, the first prototype of our multi-agents framework.

References

[1] Marie Beurton-Aimar, Bernard Korzeniewski, Thierry Letellier, Stephane Ludinard, Jean-Pierre Mazat, and Christine Nazaret. Virtual mitochondria: metabolic modelling and control. *Mol Biol Rep*, 29(1-2):227–232, 2002.

[2] R. Caspi, H. Foerster, C. Fulcher, R. Hopkinson, J. Ingraham, P. Kaipa, M. Krummenacker, S. Paley, J. Pick, S. Rhee, C. Tissier, P. Zhang, and P. Karp. Metacyc: A multiorganism database of metabolic pathways and enzymes. *Nucleic Acids Res.*, 34:D511–D516, 2006.

[3] R. Durett and S. Levin. The importance of being discrete (and spatial). *Theorical population biology*, 46:363–394, 1994.

[4] N. Gilbert and Troitzsch Klaus G. *Simulation for the Social Scientist*. Open University Press, 2005.

[5] P. Karp, S. Paley, and Romero P. "the pathway tools software". *Bioinformatics*, pages S225–32, 2002.

[6] Sonya Keene. Object-oriented programming in common lisp: A programmer's guide to clos.

[7] C. Lales, N. Parisey, J.P. Mazat, and M. Beurton-Aimar. Simulation of mitochondrial metabolism using multi-agents system. In *Fourth International Joint Conference on Autonomous Agents and Multiagent Syst ems (MAS*BIOMED'05 Workshop)*, july 2005.

[8] C. LePage and P. Cury. How spatial heterogeneity influences population dynamics: simulations in seallab. *Adaptive Behavior*, 4(3/4):255–281, 1996.

[9] N. Parisey, M. Beurton-Aimar, C. Lales, R. Strandh, and J.P. Maz at. Towards modelling the q cycle by multi agent systems. In *Systems Biology Workshop at ECAL 2005 VIIIth European Conference on Artifi cial Life*, Canterbury, Kent (UK), 5th-7th sept 2005.

[10] Guy L. Steele. Common lisp : The language, 1991.

[11] Robert Strandh and Timothy Moore. A free implementation of clim. In *International Lisp Conference*, oct 2002.