Constraint Based Languages for Biological Reactions

Stefano Bistarelli[1,2,3] and Marco Bottalico[1]

[1] Università G. d'Annunzio, Pescara, Italy
[bottalic,bista]@sci.unich.it
[2] Istituto di Informatica e Telematica (CNR), Pisa, Italy
[stefano.bistarelli]@iit.cnr.it
[3] Dipartimento di Matematica Informatica, Università di Perugia, Italy
bista@dipmat.unipg.it

Abstract. In this paper, we study the modelization of biochemical reaction by using concurrent constraint programming idioms. In particular we will consider the stochastic concurrent constraint programming (sCCP), the Hybrid concurrent constraint programming languages (Hcc) and the Biochemical Abstract Machines (BIOCHAM).

Keywords: Biochemical Reactions, (Stochastic - Hybrid) Concurrent Constraint Programming, Biocham.

1 Introduction

System biology is a science integrating experimental activity and mathematical modeling. They study the dynamical behaviors of biological systems. While current genome project provide a huge amount of data on genes or proteins, lots of research is still necessary to understand how the different parts of a biological system interact in order to perform complex biological functions. Mathematical and computational techniques are central in this approach to biology, as they provide the capability of formally describing living systems and studying their proprieties.

A variety of formalisms for modeling biological systems has been proposed in the literature. In [2], the author distinguishes three basic approaches: discrete, stochastic, continuous, additionally we have various combinations between them. Discrete models are based on discrete variables and discrete state changes; continuous models are based on differential equations that typically model biochemical reactions; finally in the stochastic the probabilities may appear explicitly in random variables and random numbers, or implicitly like in kinetics laws. In the latest approach we have a simplified representation of processes, an integration of stochastic noise in order to get more realistic models. The need to capture both discrete and continuous phenomena, motivates the study of dynamical systems [6].

The goal of this paper is to show different kinds of languages to model biochemical reactions in order to compare and to use their appropriate features in different ways.

2 Background on Biochemical reactions and Blood Coagulation

In this paper, we want to examine the Biochemical Reactions; they are chemical reactions involving mainly proteins. In a cell, there are many different proteins, hence, the number of reactions that can take place, can be very high. All the interactions that take place in a cell, can be used to create a diagram, obtaining a biochemical reaction network.

We will examine in the following, one of the thirteen enzymatic reaction of the blood coagulation [13], in the generic form, through the Michaelis-Menten kinetics:

$$XI + XIIa \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} XI : XIIa \rightharpoonup^{k_2} XI + XIa$$

where XI is the enzyme (E) that binds substrate (S) = $XIIa$, to form an enzyme-substrate complex (ES) = $XI : XIIa$. After we have the formation of product (P) = XIa and the release of the unchanged enzyme (E) = XI, ready for a new reaction.

We are interesting at the Blood Coagulation [13]. It is part of an important host defense mechanism termed hemostasis (the cessation of blood loss from a damaged vessel). Blood clotting is a very delicately balanced system; when hemostatic functions fail, hemorrhage or thromboembolic phenomena results. The chemical reactions that constitute all the process, can be see as a decomposition of many kinds of enzymatic reactions, involved reactants, products, enzymes, substrates, stoichiometric coefficients, proteins, inhibitors and chemical accelerators. Upon vessel injury, platelets adhere to macromolecules in the subendothelial tissues and then aggregate to form the primary hemostatic plug. The platelets stimulate local activation of plasma coagulation factors, leading to generation of a fibrin clot that reinforces the platelet aggregate. Later, as wound healing occurs, the platelet aggregate and fibrin clot are broken down. Mechanisms that restrict formation of platelet aggregates and fibrin clots to sites of injury are necessary to maintain the fluidity of the blood.
The classical model of blood coagulation involves a series (or "cascade") of zymogen activation reactions as shown in fig. 1. At each stage a precursor protein (zymogen) is converted to an active protease by cleavage of one or more peptide bonds in the precursor molecule. The types of components that can be involved at each stage include the following:

- a protease (from the preceding stage)
- a zymogen
- a non-enzymatic protein cofactor
- calcium ions
- an organizing surface (provided by a phospholipid emulsion in vitro or by platelets in vivo)

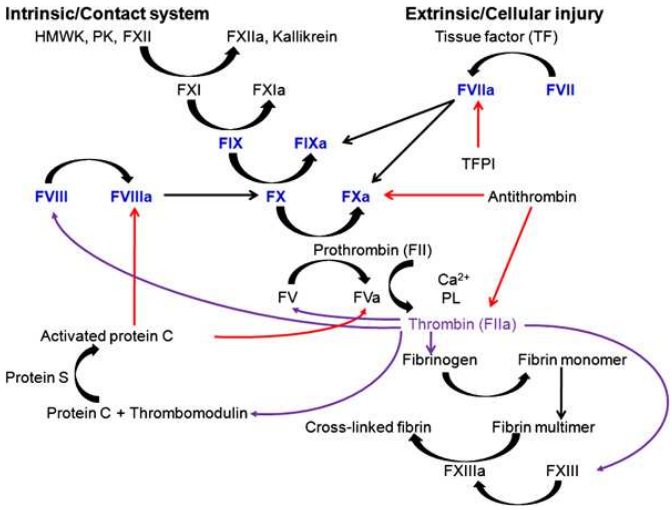

Fig. 1. Coagulation Cascade: (http:dels.nas.edu/ilar_n/ilarjournal/50_2/Graphics/50_2_144f1.jpg)

Under the **Michaelis-Menten** hypotheses [5], the most important equations are:
$K_M = \frac{k_{-1}+k_2}{k_1}$ Michaelis constant. It measures the affinity of the enzyme for the substrate: if K_M is small there is a high affinity, and viceversa.
$V_{MAX} = V_0 = k_2[E_0]$. This is the maximum rate, would be achieved when all of the enzyme molecules have substrate bound (Hp1). $[E_0]$ is the starting quantity of enzyme E. k_2 is also called k_{cat}.
$\frac{d\,[P]}{dt} = \frac{V_{MAX}[S]}{K_M + [S]}$. This final equation, is usually called the "Michaelis-Menten equation". It shows the speed of the formation of the product.

When the amount of product P is small, this will be a good approximation and the equations can now be integrated:

E: $\frac{d\,[E]}{dt} = (k_2 + k_{-1})[ES] - (k_1)[E][S]$.
S: $\frac{d\,[S]}{dt} = k_{-1}[ES] - (k_1)[E][S]$.
ES: $\frac{d\,[ES]}{dt} = k_1[E][S] - (k_{-1} + k_2)[ES]$.
P: $\frac{d\,[P]}{dt} = k_2[ES]$.

3 Concurrent Constraint Programming

The Concurrent Constraint (cc) programming paradigm [11] concerns the behaviour of a set of concurrent agents with a shared store, which is a conjunction of constraints. Each computation step possibly adds new constraints to the store. Thus information is monotonically added to the store until all agents have evolved. The final store is a refinement of the initial one and it is the result of the computation. The concurrent agents do not communicate directly with each

other, but only through the shared store, by either checking if it entails a given constraint (*ask* operation) or adding a new constraint to it (*tell* operation).

For the CCP's syntax, we have that P is the class of programs, F is the class of sequences of procedure declarations (or clauses), A is the class of agents, c ranges over constraints, and x is a tuple of variables. Each procedure is defined (at most) once, thus nondeterminism is expressed via the + combinator only. We also assume that, in $p(x) :: A$, we have $vars(A) \subseteq x$, where $vars(A)$ is the set of all variables occurring free in agent A. In a program $P = F.A$, A is the initial agent, to be executed in the context of the set of declarations F. This corresponds to the language considered in [11] which allows only guarded nondeterminism. The syntax is the following:

$P ::= F.A$
$F ::= p(x) :: A \mid F.F$
$A ::= success \mid fail \mid tell(c) \ \rightarrow A \mid E \mid A \| A \mid \exists_x A \mid p(x)$
$E ::= ask(c) \rightarrow A \mid E + E$

4 Stochastic Concurrent Constraint Programming

sCCP [3] is obtained by adding a stochastic duration to the instruction interacting with the constraint store C, i.e. *ask* and *tell*. The most important feature added in the sCCP is the continuous random variable T, associated with each instruction. It represents the time needed to perform the corresponding operations in the store. T is exponentially distributed, and its probability density function is $f(\tau) = \lambda e^{-\lambda\tau}$ where λ is a positive real number (rate of the exponential random variable) representing the expected frequency per unit of time. The duration of an ask or a tell can depend on the state of the store at the moment of the execution.

The main difference of sCCP with classical cc, is the presence of two different actions with temporal duration, *ask* and *tell*, identified by a rate function λ: $tell_\lambda(c)$ and $ask_\lambda(c)$, following the probability law. It means that the reaction occurs in a stochastic time T, $f(\tau) = \lambda e^{-\lambda\tau}$ whose mean is $1/\lambda$; i.e. $tell_\infty$ is an instantaneous execution while $tell_0$ never occurs.
Other functions are the same that in CCP, except for the variables, that in CCP are rigid, in the sense that, whenever they are instantiated, they keep that value forever. Time-varying variables (called stream variables) can be easily modeled in sCCP as growing lists with a unbounded tail: $X = [a_1, ..., a_n|T]$. When the quantity changes, we simply need to add the new value, say b, at the end of the list by replacing the old tail variable with a list containing b and a new tail variable: $T = [b|T']$. When we need to know the current value of the variable X, we need to extract from the list, the value immediately preceding the unbounded tail. The stream variables are denoted with assignment \$=.

We model the biochemical equation [5], in sCCP, with the following recursively defined method:

$react(XIIa, XIa, K_M, V_0) :-$
$ask_{r_{MM}(K_M,V_0,XIIa)}(XIIa > 0).$
$(tell_\infty(XIIa \$= XIIa - 1) || tell_\infty(XIa \$= XIa + 1)).$
$react(XIIa, XIa, K_M, V_0)$
Where the rate λ of the *ask*, is computed by the Michaelis-Menten kinetics: $r_{MM(K_M,V_0,XIIa)} = \frac{V_0 \times XIIa}{XIIa + K_M}$. Roughly the program inserts in the store the current value for the variables $K_M, V_0, XIIa$; it checks the value of the factor $XIIa$, then, with an immediate effect, it updates the values for the factors $XIIa$ (reagent) and XIa (product) with the new values. Subsequently it executes a new instance of the program.

5 Hybrid cc

Hybrid concurrent constraint programming languages (Hybrid cc [1]) is a powerful framework for modeling, analyzing and simulating hybrid systems, i.e., systems that exhibit both discrete and continuous change. It is an extension of Timed Default cc [12] over continuous time. One the major difficulty in the original cc framework is that cc programs can detect only the presence of information, not the absence [2]. Default cc extends cc by a negative *ask* combinator ($if\ a\ else\ A$) which imposes the constraint a at the program A.

The cc paradigm has no concept of timed execution. For modeling discrete, reactive systems, introduced the idea (from synchronous programming) that the environment reacts with a system (program) at discrete time ticks. At each time tick, the program executes a cc program, outputs the resulting constraint, and sets up another program for execution at the next clock tick. Concretely, this led to the addition of two control constructs to the language *next A* (execute A at the next time instant), and *always A* (execute A at every time instant). Thus, intuitively, the discrete timed language was obtained by uniformly extending the untimed language (cc or Default cc) across (integer) time [12].

To model Biological systems in Hybrid cc, we used the following schema [2]:

Biology	Hybrid cc
reaching thresholds	discrete events
time, concentration	continuous variables
kinetics	differential equations
gene interaction	concurrency
stochastic behevior	random numbers

Authors in [12] allows constraints expressing initial value (integration) problem, e.g. constraints of the form $init(X = 0)$; $cont(dot(X) = 1)$ read as follows: the initial value of X is 0; the first derivative of X is 1; from these we can infer at time t that $X = t$. Additionally they adds to the untimed Default cc a single temporal control construct: *hence A*. Declaratively, *hence A* imposes the constraints of A at every time instant after the current one. Operationally, if *hence A* is invoked at time t, a new copy of A is invoked at each instant in $(t; 1)$.

The current implementation of Hcc [12], supports two types of constraints, which are handled by interval methods: ordinary differential equations and nonlinear algebraic constraints [2].

```
e = 10,
s = 5,
es = 0.01,
p = 0,
   always { k1 = 1, km1 = 0.1, k2 = 0.01,
     cont(e), cont(s),
     if (e >= 0.000000001) then {
       e' = ((km1+k2) * es) - (k1 * e * s),
       s' = (km1 * es) - (k1 * e * s) ,
       es' = (k1 * e * s) - ((km1+k2) * es),
       p' = k2 * es }
                              else { e' = 0, s' = 0, es' = 0, p' = 0 }

         }
```

We can observe that in the first row, we have the initial conditions with the quantity of Enzyme (100), Substrate (10), Enzyme-Substrate (0), Product (0), and the rate of the three reactions: $k_1 = 1$ for the first one, $k_{-1} = 0.1$ for the second one and $k_2 = 0.01$ for the third reaction. With the syntax $cont(e, s)$ we assert that the rates are continuous. Subsequently we control the current values of e and s then we can start the reaction; the = operators, has the usual meaning of "equal". We can easily obtain the quantity of e, s, es, p in the next time instant. If the "if condition" doesn't hold, we obtain the previous amount of factors.

6 Biochemical Abstract Machine

Biochemical Abstract Machines (BIOCHAM [7]) is a software environment for modeling complex cell processes, making simulations (i.e. in silico experiments), formalizing the biological properties of the system know from real experiments, checking them and using them as specification when refining a model. BIOCHAM is based on two aspects: the analysis and simulation of boolean, kinetic and stochastic model and the simulation of biological proprieties in temporal logic. For kinetics model, BIOCHAM can search for appropriate parameter values in order to reproduce a specific behavior observed in experiments and formalized in temporal logic.

We can use the Michaelis-Menten kinetics to represent the first enzymatic reaction of blood coagulation. To explain the language used to model the reaction in the BIOCHAM [4] language, we can translate the syntax in the following way:

```
(k1*[E]*[S],km1*[ES]) for E + S <=> ES.
k2*[ES] for ES => E + P.
```

```
parameter(k1,1).
parameter(km1,0.1).
parameter(k2,0.01).
present(E,100).
present(S,10).
absent(ES).
absent(P).
```

There are two different syntax operator, used to model the different kinds of reaction: <=> and =>. The first one model the reversible reaction, involved in the ES formation, this reaction can be reversible. The second one model the irreversible reaction which produce the P factor. The for operator show us for which substances, the reaction is performed: the first for is for the $E+S \rightleftharpoons^{k_1}_{k_{-1}} E:S$ reaction, the second one for the $E:S \rightarrow^{k_2} E+P$ reaction.

The result of this simulation, in BIOCHAM generate a graph to the plot a relation graph and a table of results in relation to time variation. In the graph (fig.2), we have the plot of four kinds of curves, referring to the four substances involved in the reaction: the initial decreasing of the Enzyme and of the Substrate (violet and red curves respectively), the formation to the Enzyme-Substrate complex (green curve) and finally the Product formation (sky blue line).

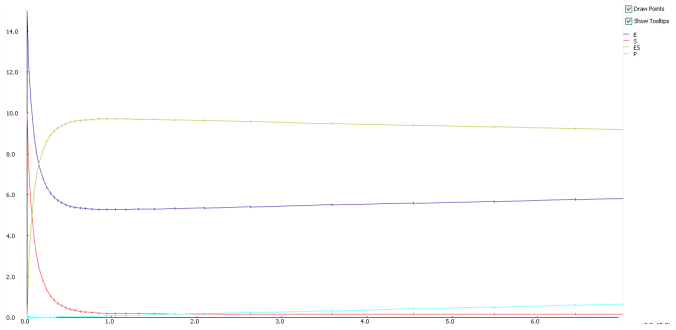

Fig. 2. Graphic

7 Related and future works

Most important features are represented in [10, 8]. In [10] the authors suggests to model biomolecular process i.e. protein networks, by using the pi-Calculus, while in [8] is shown that there are two formalisms for mathematically describing the time behavior of a spatially homogeneous chemical systems: the deterministic approach and the stochastic one. The first regards the time evolution as a continuous and predictable process which is governed by a set of ordinary

differential equations (the "reaction-rate equations"), while the seconds regards the time evolution as a kind of random-walk process which is governed by a single differential-difference equation (the "master equation").

From the application point of view, the examined languages allows the biologist to model biological systems in a high-level and declarative way, using different kinds of applications and languages construct that capture directly a variety of biological phenomena. We are interest in the non Deterministic process Calculus (ntcc [9]) because it is a concurrent constraint programming which include time process with a graphic formation, in order to describe our biochemical reactions.

The aim of many researches in the bioinformatics field is to improve the modeling features of data in order to describe biological behaviors in a more accurate way, such that they can be used in in silico modeling in the drug sperimentations (preclinic) area; we aim to save time and money in the last sperimental phase on humans. Biological reactions are the first step towards a description of cycles cellular-based mechanisms.

References

1. Panos J. Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors. *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*. Springer, 1995.
2. Alexander Bockmayr and Arnaud Courtois. Using hybrid concurrent constraint programming to model dynamic biological systems. In *ICLP*, pages 85–99, 2002.
3. Luca Bortolussi and Alberto Policriti. Modeling biological systems in stochastic concurrent constraint programming. *Constraints*, 13(1-2):66–90, 2008.
4. Nathalie Chabrier-Rivier, François Fages, and Sylvain Soliman. The biochemical abstract machine biocham. In *Proc. CMSB*, pages 172–191, 2004.
5. Thomas Delvin. *Texbook of biochemistry with clinical correlations*. McGraw Hill Book co., 2001.
6. Schaft A.J. van der and J.M. Schumacher. *Introduction to Hybrid Dynamical Systems*. Springer-Verlag., 1999.
7. François Fages. Temporal logic constraints in the biochemical abstract machine biocham. In Patricia M. Hill, editor, *LOPSTR*, volume 3901 of *Lecture Notes in Computer Science*, pages 1–5. Springer, 2005.
8. Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. In *The Journal of Physical Chemistry*, pages 2340–2352, 1977.
9. Julian Gutiérrez, Jorge A. Pérez, Camilo Rueda, and Frank D. Valencia. Timed concurrent constraint programming for analysing biological systems. *Electr. Notes Theor. Comput. Sci.*, 171(2):117–137, 2007.
10. Aviv Regev, William Silverman, and Ehud Y. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Proc. Pacific Symposium on Biocomputing*, pages 459–470, 2001.
11. Vijay A. Saraswat. The concurrent constraint programming research programmes. In *CP*, page 588, 1995.
12. Vijay A. Saraswat, Radha Jagadeesan, and Vineet Gupta. Timed default concurrent constraint programming. *J. Symb. Comput.*, 22(5/6):475–520, 1996.
13. Williams. *Williams Hematology*. McGraw Hill Book co., 2006.