Constraint-based simulation of biological systems described by Molecular Interaction Maps

Luca Bortolussi
Dept. of Mathematics and Computer Science, University of Trieste, Italy.
luca@dmi.units.it

Simone Fonda
Dept. of Computer Science, University of Pisa, Italy.
simone.fonda@gmail.com

Alberto Policriti
Dept. of Mathematics and Computer Science, University of Udine, Italy
Istituto di Genomica Applicata, Udine, Italy
alberto.policriti@dimi.uniud.it

Abstract

We present a method to simulate biochemical networks described by the graphical notation of Molecular Interaction Maps within stochastic Concurrent Constraint Programming. Such maps are compact, as they represent implicitly a wide set of reactions, and therefore not easy to simulate with standard tools. The encoding we propose is capable to stochastically simulate these maps implicitly, without generating the full list of reactions.

1 Introduction

The aim of this work is the simulation of biological regulatory networks described by the graphical notation of Molecular Interaction Maps (MIM) [11]. In order to achieve this goal, we define an encoding of such maps in stochastic Concurrent Constraint Programming (sCCP) [1], a stochastic, concurrent, and constraint-based programming language.

In scientific literature, many mathematical tools have been proposed to describe, model, and simulate the behaviors of biological systems, all sharing the common goal of organizing and analyzing the available knowledge and understanding of such systems [9]. As happens with computer programming languages, one formalism could be better suited than another for a specific purpose, depending on the features it provides to its users. When choosing a formalism many parameters must be taken into account: continuous or discrete representation of the entities, expressivity, availability of mathematical analysis techniques, possibility of a computer aided simulation and visualization, and so on. Other important properties of a modeling language concern the process of writing and maintaining a model: compositional languages are usually preferable for systems composed of many interacting parts, like biological ones [12]. Formalisms can also differ in the size of produced models; generally, the more compact the better. Moreover, the same system can be modeled at different levels of detail; for example, a cell can be described as a functional black box or specifying its internal behaviors and mechanisms in detail.

As the focus of modeling in biology is to understand dynamical properties of biological systems, most of the modeling languages have a semantic based on ordinary differential equations [15] or stochastic processes [16]. Stochastic processes, usually continuous-time Markov Chains [16], have been used in biochemistry and biology since a long time, especially after the publishing of a simple and efficient simulation algorithm by Gillespie [8]. These models are generally more realistic than the ones based on Ordinary Differential Equations (ODEs), as they represent molecules as discrete quantities (compared to the continuous approximation of ODEs) and, moreover, they have a noisy evolution (compared to the determinism of ODEs), an important issue in biology [16].

Generally, both stochastic and differential models can be obtained in a canonical way when the list of reactions of the system is fully specified [16] (i.e., we need to specify all

the possible ways of reacting of all the possible reactants). A big problem with this approach is the *cost* of the notation: in some cases, the effort required for a complete specification can be overwhelming; this is true especially for bio-regulatory networks, due to the central role played by multi-molecular complexes and protein modifications [10, 11, 6]. In fact, even a small set of proteins can potentially generate a big number of different complexes, of which only a few different kinds may be present at a certain time. Manually listing all these complexes can be a very hard task.

In order to tackle this problem, we need a formalism that can work at an higher level of abstraction, representing entities and behaviors in an implicit way. One possibility is offered by the Molecular Interaction Maps [11]: they are a compact graphical notation proposed by Kohn, with the goal to be clear, easy to use, compact, unambiguous, and (hopefully) widespread. In the author's expectations, the equivalent of electronic circuit diagrams for biologists.

The contribution of this work is the definition of an encoding of MIM in sCCP, a stochastic extension of Concurrent Constraint Programming [1], already used to model biological systems at different degrees of complexity [3, 2]. The crucial ingredient of such an encoding will be the *implicit* representation of complexes and reactions. Essentially, molecular complexes will be represented by graphs, modified locally by reactions. This work is related to κ-calculus [5], a ruled based language describing complexation implicitly, and to β-binders [4], an extension of π-calculus where π-processes are encapsulated in boxes that can be complexed together. Before presenting the encoding, however, we need to discuss in more detail Molecular Interaction Maps (Section 2). Then, after briefly recalling sCCP (Section 3), we present the main ideas of the implicit simulation of MIMs in sCCP (Section 4). Finally, Section 5 presents some experimental results, while conclusions and future works can be found in Section 6.

2 Molecular Interaction Maps

We briefly introduce now the MIM notation; the interested reader is referred to [11] for a detailed presentation. A MIM is essentially a graph, where nodes correspond to different (basic) molecular species, linked by different kinds of connecting lines, see Figure 1 for an example. The notation follows few general principles: to keep the diagram compact an *elementary molecular species*, like A, B or C in Figure 1(a), generally occurs in *only one place* on a map. Different interactions between molecular species, instead, are distinguished by different lines and arrowheads; for instance, the double line in Figure 1(a) represents a covalent modification of B (in this case a *phosphorylation*, i.e. the attachment of a phosphate group in a specific place in the protein), while the single double-barbed arrows de-

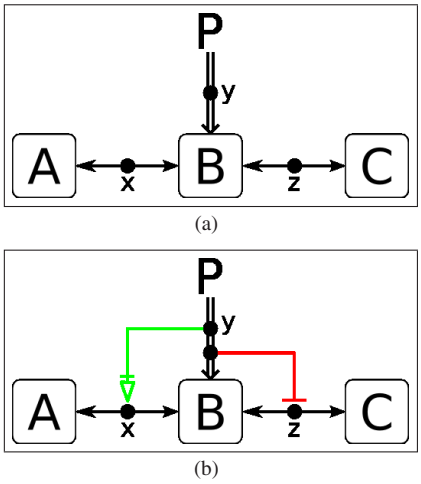


Figure 1. Two very simple MIMs

note complexation operations. *Complex molecular species*, or simply complexes, are created as a consequence of interactions and are indicated by small circles on the corresponding interaction line; e.g. in Figure 1(a) x represents the complex $A : B$, the result of a complexation between A and B, while y represents the phosphorylated B molecule (denoted hereafter with pB).

In general, there are two types of interaction lines: *reactions* and *contingencies*. The former operate on molecular species, the latter on reactions or other contingencies. The line with a T-shaped end of Figure 1(b) is an *inhibition* line; it states that phosphorylated B (indicated by y) cannot bind to C, (in fact the line terminates on the barbed arrow connecting B to C). Another contingency symbol of Figure 1(b) is the arrow with a bar preceding its empty arrowhead, terminating in x. This line represents *requirement*: B must be phosphorylated in order to bind to A. Note that multiple nodes on an interaction line represent exactly the same molecular species.

In order to explain the differences between Figure 1(a) and Figure 1(b), we need to introduce the *interpretations* of MIMs. The MIM notation, in fact, can be equipped with two different interpretations: *explicit* and *combinatorial*.
In the explicit interpretation, an interaction line applies only to the molecular species directly connected to it. Looking again at Figure 1(a), we can say that B can bind to A (forming the complex $A : B$) or to C (forming the complex $B : C$) but there is no way the complex $A : B : C$ will be formed. Moreover if B is phosphorylated, it cannot bind neither to A nor to C.
In the combinatorial interpretation, an interaction line represents a functional connection between domains or sites that (unless otherwise indicated) is independent of the modifications or bindings of the directly interacting species. In

this way the map of Figure 1(a) states that A can bind to B, independently of the state of B. For instance, B could be phosphorylated and thus forming the $A : pB$ complex, or it could be bound to C, resulting in the $A : B : C$ complex. Moreover, the combinatorial interpretation represents implicitly a large number of molecules: a single complexation arrow usually handles, on both sides, a big set of reactants. Consider, for example, the arrow connecting A to B in Figure 1(a); in the combinatorial interpretation it represents four reactions, namely the complexation of A with B, pB, $B : C$ and $pB : C$.[1]

In principle, it is always possible to create an explicit MIM with the same behaviors of a combinatorial MIM, introducing more reaction arrows and contingencies.[2] Explicit MIMs can be easily translated into a set of ODEs or into an explicit stochastic model for computer simulation, see [10]. Unfortunately, making explicit a combinatorial MIM is a non-trivial job, due to the combinatorial explosion of reaction arrows and contingencies needed in the map (making explicit map of Figure 1(a) requires nine more arrows). In addition, an explicit ODE-based (or stochastic) simulation, like the one described in [10], requires to consider all possible molecular complexes that can be generated in the system as reactants or products of some reaction. However, at a given time, usually only a small subset of these complexes is present, hence the effort needed to generate explicitly this large number of complexes is largely unmotivated.

Our goal is precisely to define a simulation operating directly at the level of the combinatorial interpretation of MIMs, hence this is the interpretation we will consider in the rest of the paper. The advantages of this choice are clear: during each stage of a simulation, we need to represent only the complexes present in the system at that time. Moreover, models can be defined using the compact notation of combinatorial MIMs, hence the resulting map is usually smaller, thereby easier to build and understand.

The biggest obstacle towards an implicit simulation of MIMs is the fact that its interpretations are *ambiguous*. We tackled this problem defining a set of *graph rewriting rules* that disambiguate the maps, in what we deem a biologically plausible way. This is a delicate point, as these rules need to be able to disambiguate each possible input in an automatic way. We are still working on this question, which can be seen as a first step towards the definition of a formal semantics for MIMs.

Another crucial issue for the simulation of MIMs is the knowledge of the rates of reactions. In general, these values are not known, but they need to be inferred or guessed in some way. In fact, MIMs usually encode just the structure of the network, although rates are fundamental to the understanding of dynamical behavior. However, this limit afflicts all modeling techniques and it is not the focal point of this work.

[1] The combinatorial interpretation considers these reactions as happening uniformly, i.e. with the same rate whatever the context. This problem can be circumvented using specific contingency arrows, like stimulation arrows.

[2] Contingencies are used in the combinatorial interpretation to restrict behaviors. For instance, in the map of Figure 1(b), the explicit and the combinatorial interpretations coincide.

3 Stochastic Concurrent Constraint Programming

Concurrent Constraint Programming (CCP [13]) is a process algebra having two distinct entities: agents and constraints. Constraints are interpreted first-order logical formulae, stating relationships among variables (e.g. $X = 10$ or $X + Y < 7$). Agents in CCP, instead, have the capability of adding constraints (`tell`) into a "container" (the *constraint store*) and checking if certain relations are entailed by the current configuration of the constraint store (`ask`). The communication mechanism among agents is therefore asynchronous, as information is exchanged through global variables. In addition to `ask` and `tell`, the language has all the basic constructs of process algebras: nondeterministic choice, parallel composition, procedure call, plus the declaration of local variables. Moreover, constraints of the store can be defined using the computational machinery of Prolog [14, 13].

The stochastic version of CCP (sCCP [1, 2, 3]) is obtained by adding a stochastic duration to all instructions interacting with the constraint store $\mathcal{C}$, i.e. `ask`, `tell`. Each instruction has an associated random variable, exponentially distributed with rate given by a function associating a real number to each configuration of the constraint store: $\lambda : \mathcal{C} \to \mathbb{R}^+$.

The underlying semantic model of the language (defined via structural operational semantic, cf. [1, 2]) is a Continuous Time Markov Chain [16] (CTMC), i.e. a stochastic process whose temporal evolution is a sequence of discrete jumps among states in continuous time. States of the CTMC correspond to configurations of the sCCP-system, consisting in the current set of processes and in the current configuration of the constraint store. The next state is determined by a stochastic race among all active instructions such that the fastest one is executed.

Time-varying quantities, an important ingredient to deal with biological systems [3], are modeled as *stream variables*, i.e. growing lists with an unbounded tail.

In [2, 3] we argued that sCCP can be conveniently used for modeling a wide range of biological systems, like biochemical reactions, genetic regulatory networks, the formation of protein complexes, and the process of folding of a protein. In fact, while maintaining the compositionality of process algebras, the presence of a customizable constraint

store and of variable rates gives a great flexibility to the modeler.

4 Encoding MIMs in sCCP

The starting point for the direct simulation of MIM in sCCP is the definition of a suitable representation of molecules and complexes. Actually, when compared to other process algebras like π-calculus, sCCP offers a crucial ingredient in this direction: the presence of the constraint store. In fact, the store is customizable and every kind of information can be represented by the use of suitable constraints, i.e. logical predicates. Manipulating and reasoning on such information can be performed in a logic programming style [13]. These ingredients make the constraint store an extremely flexible tool that can be naturally used to represent the data structures needed to operate on MIMs.

The idea to simulate MIMs is simple: we operate directly on the graphical representation. Of course, a MIM contains all possible interactions of the system, hence the graphical representation used in the simulation must be specialized to single molecules and complexes. Complexes, in particular, can be seen as *graphs*, with *nodes representing the basic molecules* (i.e. the proteins) of the complex, and with *edges representing the chemical bonds* tying them together. Such graphs will be referred in the following as *complex-graphs*.

Recalling Figure 1 and the combinatorial interpretation of Section 2, we can easily convince ourselves that molecular species are defined by the collection of their interaction sites. In our encoding, these sites are represented by *ports* (or better, *port-types*), i.e. terminating points of one single arrow in the MIM. Port-types are characterized by a unique identifier, called *port_type_id*, and by a boolean variable, INH_p, storing the state of the port. In fact, each port can be active ($INH_p = false$), meaning that it can take part to the corresponding reaction, or inhibited ($INH_p = true$) by biological mechanisms specified in the map.
Molecular-types correspond to nodes of the MIM or to points in the middle of an arrow (i.e., terminating points of reaction arrows). They consist of an unique identifier, *molecular_type_id*, of a list of port-types, implicitly determining all the possible reactions the molecule can be involved into, and of a list of contingencies starting from it (we present the treatment of contingencies at the end of the section). For instance, in Figure 1(a), A and x nodes define two distinct molecular-types.

Each graph of a complex can contain, in principle, several instances of the same molecular-type, just think of the case in which two copies of the same molecule are bound together (the so-called homodimers). These different copies are distinguished by an unambiguous naming system inside each complex: each node of a complex-graph is numbered by an integer, local to that complex, called *mol_id*. Moreover, each different complex graph that can be constructed according to the prescription of the MIM, identifies a *complex-type*; complex-types are also given an unique id, *complex_type_id*, assigned at run-time whenever a new complex-type is created.

Complex-graphs and molecular-types can be easily represented in sCCP. In fact, we just need to store all the characterizing information in suitable logical predicates.
Another important class of predicates, crucial for the run-time engine, are those counting the number of objects of a certain type. Specifically, at run-time we need to count how many complexes we have for each different complex-type, and how many active ports we have, for any port-type.

The reason for updating the number of ports or complexes at run-time, lies in the definition of the stochastic model for the simulation of MIMs. We adopt a classical approach, defining the speed of a reaction according to the *principle of mass action* [8]: the speed of each reaction is proportional to the quantity of each reactant. In our encoding, each reaction involves one or two port-types, hence its speed will be proportional to the number of active instances of such port-types.

The sCCP program associated to a MIM can be seen as a loop composed of 4 basic steps:

1. choose the next reaction to execute;
2. choose the reactants;
3. create the products;
4. apply contingency rules to products.

It gives rise to a stochastic model (a Continuous Time Markov Chain) that can be simulated by a classical Monte Carlo algorithm like Gillespie's direct method [8] (see [2] for details on a meta-interpreter for sCCP).
We give now some details on the sCCP program. The choice of the next reaction can be seen as a stochastic race among all the enabled reactions. In sCCP, this effect is obtained associating an agent to each reaction arrow of the molecular interaction map, called *reaction agent*. This agent tries to execute at a rate defined according to the principle of mass action. If the agent wins the competition, it needs to identify the actual reactants. In fact, a reaction involves one or two complexes with available ports of the required port-type. However, as different complex-types can have such ports available, we must identify the ones really involved in the reaction. Essentially, we need to pick, with uniform probability, one complex among all those having an active port of the required type. This operation is performed by *port managers*: there is one agent for each port-type, keeping an updated list of all complex types containing active ports of its type and choosing one of them upon request from a reaction agent.

When reactants have been chosen by port agents, the reaction agent generates the products of the reaction. For instance, in case of a complexation reaction, the agent has to merge two complexes into one, adding nodes and edges to the complex description and marking as bound the ports involved in the reaction (i.e. removing them from lists of the corresponding port agents).

If, after these operations, a new (i.e. not present in the system) complex-type is obtained, then all the necessary predicates are added to the constraint store. A bookkeeping of the predicates counting complexes and ports is then performed.

The last point of the simulation algorithm consists in the application of contingencies. These are the inhibition and requirement arrows, briefly introduced in Section 2. To grasp the rationale behind their implementation, consider again Figure 1(b). The inhibition arrow from y (the head of the contingency rule) to A-B complexation line (the tail) was used to forbid complexation between phosphorylated B and A. This essentially means that, if B is phosphorylated (i.e., the corresponding edge e is in the complex description), then B cannot bind to A, and so the port p_{B-A} connecting B to A, must become inactive.
This example suggests that contingencies are nothing but logical implications of the form:

IF (a set of edges E is in the complex)
THEN (some ports must become active/inactive)

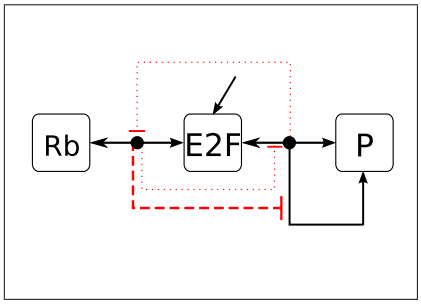
Rules of this kind are stored in each molecular-type descriptor, so that each complex-type has associated a set of contingency rules potentially applicable.
When a new complex type is created in a reaction, then the reaction agent checks what rules among those listed in the complex can be applied, and it modifies accordingly the value of INH_p of ports involved and their associated global counters.

5 Experimental Results

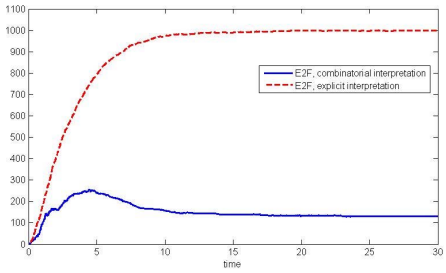
In this section we present some preliminary tests of the framework just presented. Regarding the implementation, currently we have a first prototype of sCCP, based on SICStus prolog [7]. Essentially, we have a general purpose meta-interpreter for the language, coded using SICStus's Constraint Logic Programming libraries to manage the constraint store, cf. [2] for further details. To test the encoding of MIMs in sCCP, we simply defined the templates of all agents and predicates of the encoding in such a way that each MIM can be described simply by the collection of agents associated to reactions and ports and by the description of molecular types.

The example we consider is a very simple MIM taken from [10], where Kohn first introduced the MIM notation in order to study the mammalian G1/S cell-cycle phase transition, considering networks of increasing complexity. Here we consider the simplest system, composed by three proteins. One is the transcription factor $E2F$, involved in the regulation of several genes active in the replication of DNA (the S phase of the cell cycle) and feeded externally. The second is a protein Rb, belonging to the Retinoblastoma family, inhibiting $E2F$ activity when complexed to it. The third is a Protease P involved in the degradation of $E2F$. The corresponding map, as taken from [10], is shown in Figure 2(a) (continuous lines). In [10] the explicit interpretation makes the bindings of $E2F$ exclusive. In the combinatorial interpretation, instead, these two bindings can coexist in the same complex, although in both maps the degradation of $E2F$ can happen when $E2F$ is bound only to protease P. This choice is made unequivocal in Figure 2(a) by the addition of the dashed inhibition arrow. Moreover, the explicit interpretation can be reproduced introducing the dotted inhibitions arrows.

The differences between the two interpretations are not confined to the topology of the maps, but they obviously propagate to the dynamical behavior. In Figure 2(b), the two interpretations are compared: both the dynamics and the



(a) MIM for a subnetwork of G1/S phase transition



(b) sCCP Simulation of MIM

Figure 2. Molecular Interaction Maps for a simple subnetwork of the G1/S phase transition network, and its simulation w.r.t. combinatorial and explicit interpretations.

stable values of the complex Rb:E2F are different. In fact, in the combinatorial interpretation, part of the $Rb : E2F$ complex is also bound to protease P, hence the stationary value of pure $Rb : E2F$ is lower.

This example, despite its simplicity, shows two things: the feasibility of our encoding, which is able to simulate the maps implicitly, and the fact that defining possible interaction in combinatorial MIMs is a matter of forbidding unwanted or unrealistic behaviors.

6 Conclusions

In this paper we presented an implementation in stochastic Concurrent Constraint Programming of an algorithm to simulate biological regulatory networks defined by means of Molecular Interaction Maps. This notation is implicit, as each edge in such a diagram represents a set of reactions potentially very large. Our sCCP-simulation, instead of generating the full list of reactions, is able to simulate the map implicitly, generating only those complexes that are actually present in the system at run-time. This is achieved using a graph-based representation of complexes, so that new complexes are dynamically constructed merging and splitting other complex-graphs. A prototype implementation has been written in SICStus prolog [7] and used to perform some preliminary tests.

The choice of sCCP as a language to describe (implicitly) such maps is motivated by different reasons. First of all, the details of the stochastic evolution are dealt automatically by its (stochastic) semantics. In addition, the power of constraints allows to represent and reason directly on the graph-based representation of complexes, separating this description from the definition of agents performing the simulation. Another important motivation is that the model built in such way is compositional w.r.t. the addition of new edges (and nodes) in a MIM. Finally, the size of an sCCP program associated to a MIM *scales linearly* with the description of the map (i.e. with the number of symbols needed in the description).

As the Prolog prototype lacks efficiency, we are planning to realize a more efficient implementation, interfacing it with graphical tools to design MIMs, in order to analyze big bio-regulatory networks.

References

[1] L. Bortolussi. Stochastic concurrent constraint programming. In *Proceedings of QAPL 2006, ENTCS*, volume 164, pages 65–80, 2006.

[2] L. Bortolussi. *Constraint-based approaches to stochastic dynamics of biological systems*. PhD thesis, PhD in Computer Science, University of Udine, 2007. Available at `http://www.dmi.units.it/~bortolu/files/reps/Bortolussi-PhDThesis.pdf`.

[3] L. Bortolussi and A. Policriti. Modeling biological systems in concurrent constraint programming. To appear in *Constraints*, 2007.

[4] F. Ciocchetta, C. Priami, and P. Quaglia. Modeling kohn interaction maps with beta-binders: an example. *Transactions on computational systems biology*, LNBI 3737:33–48, 2005.

[5] V. Danos and C. Laneve. Formal molecular biology. *Theor. Comput. Sci.*, 325(1):69–110, 2004.

[6] J.R. Faeder, M.L. Blinov, B. Goldstein, and W.S. Hlavacek. Rule-based modeling of biochemical networks. *Complexity*, 10:22–41, 2005.

[7] Swedish Institute for Computer Science. Sicstus prolog home page.

[8] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. of Physical Chemistry*, 81(25), 1977.

[9] H. Kitano. Computational systems biology. *Nature*, 420:206–210, 2002.

[10] K. W. Kohn. Functional capabilities of molecular network components controlling the mammalian g1/s cell cycle phase transition. *Oncogene*, 16:1065–1075, 1998.

[11] K. W. Kohn, M. I. Aladjem, J. N. Weinstein, and Y. Pommier. Molecular interaction maps of bioregulatory networks: A general rubric for systems biology. *Molecular Biology of the Cell*, 17(1):1–13, 2006.

[12] C. Priami and P. Quaglia. Modelling the dynamics of biosystems. *Briefings in Bioinformatics*, 5(3):259–269, 2004.

[13] V. A. Saraswat. *Concurrent Constraint Programming*. MIT press, 1993.

[14] L. Shapiro and E. Y. Sterling. *The Art of PROLOG: Advanced Programming Techniques*. The MIT Press, 1994.

[15] S. H. Strogatz. *Non-Linear Dynamics and Chaos, with Applications to Physics, Biology, Chemistry and Engeneering*. Perseus books, 1994.

[16] D. J. Wilkinson. *Stochastic Modelling for Systems Biology*. Chapman & Hall, 2006.