A Framework for Modelling and Simulating Networks of Cells

Sara Montagna, Mirko Viroli

Alma Mater Studiorum–*Università di Bologna*
via Venezia 52, 47521 Cesena, Italy
{sara.montagna,mirko.viroli}@unibo.it

Abstract

Several complex biological phenomena are to be modelled in terms of a large and dynamic network of compartments, where the interplay between inter-compartment and intra-compartment events plays an essential role. Key examples are embryogenesis and morphogenesis processes, where autonomous internal dynamics of cells, as well as cell-to-cell interactions through membranes, are responsible for the emergent peculiar structures of the individual phenotype.
This paper introduces a practical framework for modelling and simulating these scenarios. This is based on *(i)* a computational model featuring networks of compartments and an enhanced model of chemical reaction addressing molecule transfer, *(ii)* a logic-oriented language to flexibly specify complex simulation scenarios, and *(iii)* a simulation engine based on the many-species/many-channels optimised version of Gillespie's direct method. As an example of application of our framework, we model the first stages of *Drosophila Melanogaster* development, which generate the early spatial pattern of gene expression, and we show the correctness of our model comparing the simulation results with real data of gene expression and spatial/temporal resolution acquired in free on-line sources.

Keywords: Formal methods, Multi-level model, Developmental biology

1 Introduction

Works on analysing biochemical networks are facing the need of tackling more and more complex biological scenarios. For instance, developmental biology aims at understanding the process of embryogenesis of multicellular organisms, studying the genetic processes that control cell proliferation and differentiation together with the signalling events among cells that co-ordinate the formation of embryo pattern and morphologies. Given the fact that the overall dynamics underlying these phenomena is extremely complex and very few biological data are readily available, the help of modelling techniques seems to acquire more and more importance—and this holds true especially when mixing together different embryogenetic mechanisms and their relations. These scenarios require tools that can support multi-scale models, where different cells form large-scale, dynamic networked systems – as e.g. in tissues of cells, organs, and even full embryos – and where both the biochemical reactions that occur inside each cell

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

and the molecules diffusion across membrane (mediating the interaction among cells) can be captured.

In this paper, we present a computational model and related framework for modelling and then simulating large networks of biological compartments (e.g. several hundreds of cells), as required by the study of phenomena like morphogenesis and embryogenesis. As such, our work differs from existing frameworks for computational systems biology like SPIM or Bio-PEPA [16,3]: although they can in principle tackle such large network scenarios, they would require additional tools (e.g. for automatic code creation) to make modelling and simulation more practical—see Section 5 for a more detailed discussion. On top of our framework, a logic-oriented specification language is used to flexibly specify simulation scenarios: on the one side it tightly focusses on biochemistry, by providing constructs to directly express biochemical reactions, compartments, compartment link topology, and reactions involving selective transfer through membranes; on the other side it relies on logic-based goal resolution and unification, achieving the expressiveness needed to easily handle size and complexity of the biochemical network. Behind the hood, such a specification is turned into an intermediate language (a sort of bytecode) that feeds a simulation engine implemented by adapting the optimised version of Gillespie's algorithm (described in [8]) to our computational model of biochemical cell networks.

To show the applicability of this framework we discuss a scenario of morphogenesis in embryo formation, where we conceive a biochemical system that – by the interplay of intra-compartment chemical reactions and inter-compartment chemical transfer – manifests the ability of regionalising a tissue of cells. In particular we model the *Drosophila Melanogaster*'s embryo development (*Drosophila* in short), reproducing the gene regulatory network that causes the early (stripes-like) regionalisation of gene expression in the anteroposterior axis [20,15].

The remainder of this paper is organised as follows: Section 2 presents – in terms of a stochastic calculus – the formal computational model that grounds our approach, Section 3 presents our logic specification language, Section 4 presents the case study, Section 5 reports about related works and finally Section 6 provides concluding remarks and discusses future works.

2 A Calculus of Biochemical Cell Networks

We start from defining the computational model that grounds our work. This is based on the well-known Gillespie's chemical model [10], extended with two concepts: *(i)* instead of a single compartment we actually have a graph-like network of compartments, where proximity is modelled by a link concept, and *(ii)* chemical laws can have in their right-hand side (namely as "products" of the chemical reaction) a so-called *firing molecule*, namely, a molecule that is ready to be sent through a link towards another compartment. As such, each link has a rate dictating the velocity of molecule transfer (hence modelling proximity).

Note that cells of a multicellular organism communicate and interact by signal molecules that are secreted in the extracellular matrix, or that remain to the surface of the signalling cell: these cues – modelled by our firing molecules

– bind specific cell-surface receptor proteins in the target cell and consequently activate an intracellular cascade there. The concepts of links and firing molecules precisely capture this mechanism: *(i)* first firing molecules are created in the source cell, *(ii)* then they are transferred to a target cell via a link – a link for a certain molecule reach cells that feature the corresponding receptor proteins, and the link rate dictates probability and speed of the transfer –, and *(iii)* finally they may activate some intracellular signalling proteins as usual.

Our model currently supports only static networks, which still allow to model interesting scenarios of embryogenesis as shown later in this paper—future works will be devoted to a coherent extension with multiply-nested compartments, and with action-molecules (similar to firing molecules) causing link repositioning, and compartment creation, splitting, joining, and so on.

For the sake of clarity, we provide a formalisation of this computational model in terms of a calculus, which resembles a process algebra though with a main difference with respect to standard process algebras like stochastic π-calculus [17]: similarly to works like the `nano` calculus [4] or Bio-PEPA [3], we model molecules directly as tokens (and not as processes), subject to reactions that resemble stochastic (Petri-net like) transitions.

2.1 Syntax

Let meta-variable σ range over compartment identifiers, M over molecule kinds, r over positive real numbers including 0, and n, m over natural numbers.
The syntax of the model (along with semantics, described later) is shown in Fig. 1.

An actual molecule entity E can be in two states, a normal molecule M or a firing molecule $\overset{\rightsquigarrow}{M}$ representing M just being sent outward the current compartment. L is a chemical law (or reaction), expressing the transformation of molecule set I (reactants) into O (products) by chemical rate r: reactants are sets of elements of kind $M\langle n\rangle$ (called "substance", representing n copies of M), and similarly products can also include firing molecules—n is also called *concentration*, though it is a discrete value. C is a compartment, made of a multiset of chemical substances and laws, while S is a whole biochemical system, which is as multiset of compartments $[\![C]\!]_\sigma$ (σ is the compartment identifier), and links. In our model a link $\sigma \overset{r,M}{\rightsquigarrow} \sigma'$ is unidirectional (from σ to σ'), and specifies the rate r at which a single molecules M can move through it. Note that many links have to be set up, each with its rate, if many molecule kinds are to be transferred out of a cell.

Fig. 1 also describes congruence (syntactic equivalence): other than stating standard properties of multiset composition operator "|", they state (in last line) that a chemical substance (even a firing one) can be either seen as joined into a single term $E\langle n\rangle$, or split in two (or recursively more) terms down to substances with concentration 1, namely single molecules—written $E\langle 1\rangle$ or simply E without risk of ambiguity.

Auxiliary infix and partial function $\oplus$ is introduced to extract the overall concentration of a substance into a solution. It takes a chemical substance $E\langle n\rangle$

Syntax:

$$
\begin{array}{lll}
E & ::= M \mid \overset{\rightsquigarrow}{M} & \text{Molecule and Firing Molecule} \\
I & ::= 0 \mid M\langle n\rangle \mid (I|I) & \text{Molecule set} \\
O & ::= 0 \mid E\langle n\rangle \mid (O|O) & \text{Firing molecule set} \\
L & ::= I \overset{r}{\mapsto} O & \text{Law} \\
C, D & ::= 0 \mid E\langle n\rangle \mid \{L\} \mid (C|C) & \text{Compartment} \\
S, R & ::= 0 \mid [C]_\sigma \mid \sigma \overset{r,M}{\rightsquigarrow} \sigma' \mid (S|S) & \text{System}
\end{array}
$$

Congruence:

$$
\begin{gathered}
0|S \equiv S \quad S|R \equiv R|S \quad (S|S')|S'' \equiv S|(S'|S'') \\
0|C \equiv C \quad C|D \equiv D|C \quad (C|C')|C'' \equiv C|(C'|C'') \\
E\langle n\rangle|E\langle m\rangle \equiv E\langle n+m\rangle \quad E\langle 0\rangle \equiv 0
\end{gathered}
$$

Auxiliary functions:

$$
E\langle n\rangle \oplus C = \begin{cases} E\langle n\rangle|C & \text{if } E\langle m\rangle \notin C \\ \bot & \text{otherwise} \end{cases}
$$

$$
G(I, M\langle m\rangle \oplus C) = \begin{cases} \binom{m}{n} * G(I', C) & \text{if } I = M\langle n\rangle \oplus I' \\ 1 & \text{if } I = 0 \end{cases}
$$

Transition rules:

$$
\begin{gathered}
S|R \xrightarrow{\ r\ } S|R' \quad \text{if } R \overset{r}{\to} R' \\
S \xrightarrow{\ r\ } S' \quad \text{if } S \equiv R, R \overset{r}{\to} R', R' \equiv S' \\
[\overset{\rightsquigarrow}{M}\langle n{+}1\rangle \oplus C]_\sigma \mid [D]_{\sigma'} \mid \sigma \overset{r,M}{\rightsquigarrow} \sigma' \xrightarrow{\ r(n+1)\ } [\overset{\rightsquigarrow}{M}\langle n\rangle \oplus C]_\sigma \mid [M|D]_{\sigma'} \mid \sigma \overset{r,M}{\rightsquigarrow} \sigma' \\
[I \mid \{I \overset{r}{\mapsto} O\} \mid C]_\sigma \xrightarrow{\ r*G(I,I|C)\ } [O \mid \{I \overset{r}{\mapsto} O\} \mid C]_\sigma
\end{gathered}
$$

Fig. 1. Computational Model

and a compartment C: if C does not include E it simply joins $E\langle n\rangle$ with C, otherwise it provides no result. Note that $\oplus$ is not a syntactic constructor (it is not part of the syntax), but it is simply a lookup function to be used through a match; for instance, the equation:

$$
\overset{\rightsquigarrow}{M}|M|M|M'|M\langle 5\rangle|M'\langle 4\rangle|M\langle 3\rangle|\{M \overset{r}{\mapsto} M'\} \equiv M\langle n\rangle \oplus C
$$

has only one solution, which gives $n = 10$ and $C = \overset{\rightsquigarrow}{M}|M'|M'\langle 4\rangle|\{M \overset{r}{\mapsto} M'\}$

Another auxiliary definition concerns function G, which takes the precondition of a law I and the content of a compartment C, and computes how many different combinations of molecules in I can be found in C—this function is key to properly compute chemical rates according to Gillespie's algorithm [10].

Suppose I includes n different copies of M (in natural chemical systems it is often supposed that $n \leq 2$ [10], though our model is more general), and let m be the overall concentration of M in the current location, then the multiplicative contribution of M to the overall number of combinations is given by binomial $\binom{m}{n}$. For instance, we have:

$$G(M|M|M', M\langle 10\rangle|M'\langle 20\rangle) = G(M|M, M\langle 10\rangle) * G(M', M'\langle 20\rangle) = \frac{10 * 9}{2} * 20$$

2.2 Operational semantics

The operational semantics of this calculus is given in terms of a Continuous-Time Markov Chains model—following the works of Gillespie [10] and of existing stochastic languages for biochemistry [17]. A transition system $(S, \rightarrow, \mathbb{R}_0^+)$ is defined (bottom part of Fig. 1) where transitions are of the kind $S \xrightarrow{r} S'$, meaning that system S moves to S' with dynamics/likelihood expressed by markovian rate r.

The former rule is the classical one of interleaved semantics of process algebras, stating that in one computation step any network subpart is allowed to change state in isolation. The second rule, again, is the classical rule of congruence: given a system configuration S, let R be an equivalent one modulo "$\equiv$", then if R moves to R' and R' is equivalent to S', it follows that S can move to S' in one step. Third and fourth rules are instead specific to our model. Third rule handles movement of one molecule towards a neighbouring location. Let σ and σ' be two locations connected by a link for molecule M with rate r, and suppose that inside σ there is $\overset{\rightsquigarrow}{M}$ with overall concentration $n + 1$; then, a transition can occur which decreases such a concentration to n, while inside σ' we add a molecule M. The rate of this transition is $r * (n + 1)$, in that each single item of $\overset{\rightsquigarrow}{M}$ can move with rate r. As an example, starting from configuration

$$S_0 = [\overset{\rightsquigarrow}{M}\langle 5\rangle]_{\sigma_1}|[\overset{\rightsquigarrow}{M}\langle 10\rangle]_{\sigma_2}|[0]_{\sigma_3}|(\sigma_1 \overset{1.0,M}{\rightsquigarrow} \sigma_2)|(\sigma_1 \overset{9.5,M}{\rightsquigarrow} \sigma_3)$$

there are two available transitions, one moving M from σ_1 to σ_3, the other from σ_2 to σ_3, which are:

$$S_0 \xrightarrow{1.0*5} [\overset{\rightsquigarrow}{M}\langle 4\rangle]_{\sigma_1}|[\overset{\rightsquigarrow}{M}\langle 10\rangle]_{\sigma_2}|[M]_{\sigma_3}|(\sigma_1 \overset{1.0,M}{\rightsquigarrow} \sigma_2)|(\sigma_1 \overset{9.5,M}{\rightsquigarrow} \sigma_3)$$
$$S_0 \xrightarrow{9.5*10} [\overset{\rightsquigarrow}{M}\langle 5\rangle]_{\sigma_1}|[\overset{\rightsquigarrow}{M}\langle 9\rangle]_{\sigma_2}|[M]_{\sigma_3}|(\sigma_1 \overset{1.0,M}{\rightsquigarrow} \sigma_2)|(\sigma_1 \overset{9.5,M}{\rightsquigarrow} \sigma_3)$$

Given the rates of the two transitions, there is higher probability (precisely 95%) that the second transition will be selected next, and its average duration time will be 1/95 time units (i.e., seconds – if chemical rates are expressed as seconds^{-1} as usual).

Fourth rule handles execution of a chemical law inside a location. Let σ be a location including law $\{I \overset{r}{\mapsto} O\}$ as well as its required reactants I; then, a transition can occur which replaces I with O, and its rate is $r * G(I, I|C)$: according to [10], each single combination of the molecules of I is equally subject

```
constant T.                          % asserts T as a fact
molecule T.                          % declares molecule with id T
reaction T : Li --> Lo rate R.       % declares chemical reaction with id T
compartment T.                       % declares compartment with id T
link Ts >>> Td rate R molecule Tm. % declares link for molecule Tm, from comp Ts to Td
concentration N of Tm in Tc.         % sets initial conc. of Tm into comp Tc equal to N
place Tr into Tc.                    % places reaction Tr in comp Tc
final_time R.                        % sets overall simulation time
final_steps N.                       % sets overall simulation steps
sample_time R.                       % sets time between two observations
sample_steps R.                      % sets number of steps between two observations
out L.                               % prints a list of items as of below
out molecule(Tc,Lm).                 % prints a molecule concentration
out time.                            % prints elapsed time
out step.                            % prints number of steps so far
out end_of_line.                     % prints a carriage return
out string(T).                       % prints a string
```

Fig. 2. Surface Language Constructs

to the chemical law with rate r. As an example, starting from configuration

$$S_0 = [M\langle 5\rangle \mid M'\langle 10\rangle \mid \{M \overset{100.0}{\longmapsto} 0\} \mid \{M|M' \overset{10.0}{\longmapsto} M\}]_\sigma$$

there are two available transitions (one per chemical law):

$$S_0 \xrightarrow{5*100.0} [M\langle 4\rangle \mid M'\langle 10\rangle \mid \{M \overset{100.0}{\longmapsto} 0\} \mid \{M|M' \overset{10.0}{\longmapsto} M\}]_\sigma$$
$$S_0 \xrightarrow{10*5*10.0} [M\langle 5\rangle \mid M'\langle 9\rangle \mid \{M \overset{100.0}{\longmapsto} 0\} \mid \{M|M' \overset{10.0}{\longmapsto} M\}]_\sigma$$

3 A Framework

Although the calculus is rather simple, it allows to model even complex simulation scenarios, featuring large networks of compartments (one can easily think at biological scenarios with several thousands of cells), with specific topological structures (lattices, torus, scale-free networks), and where each cell can have its own peculiar behaviour and initial state. However, as far as pragmatics of modelling and simulation is concerned, it is quite important to devise a surface language that can, on the one hand, provide suitable constructs to easily express the above cases, and on the other hand, intuitively "compile" into the computational model above—namely, the relationship between computational model and surface language would be the same as e.g. the well-known one between stochastic π-calculus and the SPIM language [16]. Moreover a simulator for language is required, which can properly turn the system specification into one that can be efficiently simulated by the Gillespie's algorithm.

In this section, we first described the proposed surface language, and then the main details of the implemented simulation engine

3.1 Surface language

The language we adopt is basically a description language on top of Prolog. It allows one to describe facts specifying which cells, links, reactions, and chemical substances initially exist in the system; moreover, such facts can include logic variables, and side-conditions can be specified that properly constraint their

```
constant size(50).

molecule M where (M in [pump,field]).
reaction r(pump)  : [pump] --> [pump,field] rate 10.0.
reaction r(diff)  : [field] --> [field,firing(field)] rate 0.2.
reaction r(decay) : [field] --> [] rate 0.1.

compartment c(X,Y) where (size(N), X in 1..N, Y in 1..N).
link c(X,Y) >> c(X,Y+1) rate 10000.0 molecule field.
link c(X,Y) >> c(X,Y-1) rate 10000.0 molecule field.
link c(X,Y) >> c(X-1,Y) rate 10000.0 molecule field.
link c(X,Y) >> c(X+1,Y) rate 10000.0 molecule field.
concentration 1 of pump in c(M,M) where (size(N), M is N/2).
place _ into _.

final_steps 500000.
sample_steps 50000.
out [time,end_of_line].
out S where ( compartment c(X,Y), S1 = molecule(c(X,Y),field)),
              (X=N,S=[S1,end_of_line]);S=S1
).
```

Fig. 3. Code for a 50x50 field diffusion scenario

binding—this acts as a flexible mechanism to expand concise specifications into large system configurations.

Let `T` be any first-order term, `R` any real number, `N` any natural number, and `L` any list of terms (with usual syntax `[T1,..,Tn]`), the language provides the declaration constructs shown in Fig. 2, each optionally providing a `where` condition that binds logic variables used in the declaration—such conditions can be any Prolog goal, with an additional syntactic sugar such that "`X in [1,2,3,4]`" and "`X in 1 .. 4`" unify `X` with 1,2,3,4, iteratively.

For the sake of space, we explain the semantics of these declarations informally by one simple yet interesting example. We consider a square grid of 50x50 cells, named `c(1,1)`,..,`c(50,50)`, each connected with the 4 adjacent ones (except for those in boundary positions of the grid); the cell in central position pumps a chemical substance that diffuses around and is subject to decay (as in radioactive decay); we want to simulate 5′000′000 steps and print, one each 500′000, a matrix visualising concentration of relevant substances in each cell, preceded by the elapsed time. The corresponding specification is provided in Fig. 3. First line asserts fact `size(50)`, which declares the grid size. Molecules `pump` and `field` are then declared: notice that our compiler interprets this line as a Prolog goal of the kind "`molecule(M) :- M in [pump,field]`" which yields two solutions, binding `M` to `pump` and then to `field`. More generally, each declaration can be seen as a rule $p(t_1, \ldots, t_n) : -w_1, \ldots, w_k$, where each t_i is a term and $w_1, \ldots w_k$ is the possibly-void list of goals in the `where` condition: the interpreter solves goals $w_1, \ldots w_k$, and each resulting substitution is applied to the fact $p(t_1, \ldots, t_n)$, yielding a ground command for the engine—by this mechanism, the actual "population" of molecules, cells and links is instantiated out of the front-end specification.

Considering again the example, three chemical reactions are defined, one that pumps molecules of `field` if molecule `pump` is present, one that diffuses a copy of `field` in some neighbouring compartment, and one to decay `field` substance.

Declaration `compartment` defines the 50x50 grid: note that, due to Prolog resolution, they are ordered as follows:

`c(1,1),..,c(1,50),c(2,1),...,c(50,50)`. Then, links are declared for `field` (the compiler automatically excludes links escaping the grid). Instruction `concentration` is used to place one molecule of `pump` in the center of the grid, while all other concentrations are set to 0 by default. Declaration `place` is used to place all defined reactions in all compartments. Finally, we define total number of simulation steps, number of steps in between two observations, and printing commands: the last `out` declaration emits the value of `field` in each compartment in the proper order, also producing an end-of-line at the end of each row—recall that in Prolog, ";" stands for logical disjunction.

The reader should notice that the resulting specification combines two key aspects. On the one hand, there is a tight connection between programming constructs and the computational model defined in previous section (a formal encoding is not reported for brevity): `compartment` and `link` populate the system configuration with elements $[C]_\sigma$ and $\sigma \overset{r,M}{\rightsquigarrow} \sigma'$, while `concentration` and `place` fills the content of all compartments—other constructs basically define auxiliary specification. On the other hand, preconditions can be flexibly structured to incorporate any complex scenario—by embedding proper Prolog code, e.g., one could specify a random topology of cells, or load initial concentrations in each cell from an external source—as we do in this paper.

3.2 Overview of implementation

Here we provide a brief sketch of our simulator's internals. This is basically structured in two standalone, command-line tools: a front-end compiler for the surface language, and a back-end simulation engine producing output results as a text file.

The front-end is a gnuProlog program: it receives a specification file as described in previous section, parses it, checks it for correctness, and generates an intermediate file containing a list of commands to initialise the back-end engine. The intermediate file is obtained by "compiling away" `where` conditions, namely, turning each universal declaration (a declaration with variables, and possibly preconditions) into a list of ground commands, one per correct instantiation of the declaration—as described in Section 3.1. For instance, link declaration in the example of previous section gets compiled into the commands:

```
link c(1,1) c(1,2) 10000.0 field
link c(1,1) c(2,1) 10000.0 field
...
link c(49,50) c(50,50) 10000.0 field
```

As this file can grow to more than some hundreds kilobytes in large networks, future works will be devoted to make this intermediate file more compact, namely, deferring some parts of the Prolog resolution process to the initialisation module of the engine.

The back-end engine is a simulator (written in C) for the computational model shown in Section 2: it receives the intermediate file, initialises all the proper data structures, and the proceeds with the simulation process, which is

based on the optimised Gillespie's algorithm described in [8] and used in [19], properly adapted to tackle our computational model; namely:

- A list of available actions is maintained over time, each representing a possible transition in the system, namely, one chemical reaction into each compartment of the network, and additionally, all the actions involving transfer through links.
- A dependency graph is built that links each actions a to those whose rate should change when a is executed. This extends the basic definition in [8] with the idea that a link action possibly affects also chemical reactions in the target compartment.
- A binary search tree over actions is used to select one action given a random number in between 0 and the sum of action rates. Each node of the tree keeps the sum of rates relative to the left subtree, that is used to perform left-right switch when searching the action to be executed. In particular, this tree structure guarantees that the performance of one-step simulation is $O(logN)$ where N is the number of actions, namely, it is logarithmic in the number of compartments.

The engine produces an output text file according to `out` declarations in the specification, which can be used to produce charts using standard tools like spreadsheets, `gnuplot`, or `Matlab`, as in the command-line version of SPIM [16].

4 A Model for the Morphogenesis of Drosophila

Developmental biology is a branch of life science that studies the process by which organisms develop, focussing on the genetic control of cell growth, differentiation and movement [9]. A main problem in this context is understanding the mechanisms that make the process of vertebrates' embryo regionalisation so robust, making it possible that from one cell (the zygote), the organism evolves acquiring the same morphologies each time. This phenomenon involves at the same time the dynamics of – at least – two levels including both cell-to-cell communication and intracellular phenomena: they work together, and influence each other in the formation of complex and elaborate patterns that are peculiar to the individual phenotype. This happens according to the principles of *downward* and *upward* causation, where the behaviour of the parts (down) is determined by the behaviour of the whole (up), and the emergent behaviour of the whole is determined by the behaviour of the parts [18].

One notable example of pattern formation during morphogenesis is given by the patterning along the anteroposterior axis of the fruit fly *Drosophila Melanogaster* [15,12] – as shown on the right of Fig. 6 – which in this section we take as a case study.

4.1 Biological background

The egg of *Drosophila* is already polarised by differently localised mRNA molecules which are called *maternal effects*. The first nine nuclear divisions

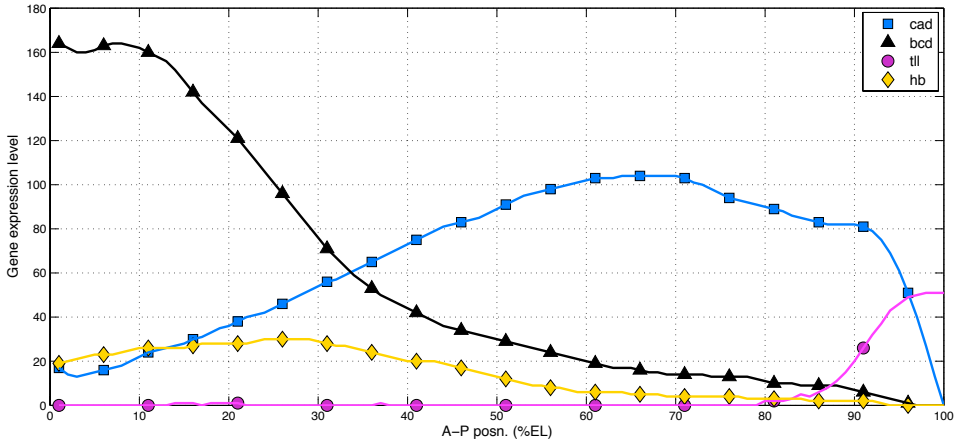

Fig. 4. Experimental data at cleavage cycle 11 of genes with non-zero concentration *bcd, cad, tll, hb*

generate a set of nuclei, most of which move from the middle of the egg towards the surface, where they form a monolayer called *syncytial blastoderm*. After other four nuclear divisions, plasma membranes grow to enclose each nucleus, converting the syncytial blastoderm into a *cellular blastoderm* consisting of about 6000 separate cells.

Up to the cellular blastoderm stage, development depends largely – although not exclusively – on maternal mRNAs and proteins that are deposited in the egg before fertilisation. After cellularisation the transcription increases dramatically. Once cellularisation is completed the gene expression regionalisation is already observable.

The building blocks of anterior-posterior axis patterning are laid out during egg formation thanks to the maternal effects. *Bicoid* and *caudal* are the maternal effect genes that are most important for patterning of anterior parts of the embryo in this early stage. They are transcription factors that drive the expression of *gap genes* such as *hunchback* (hb), *Krüppel* (Kr), *knirps* (kni) and *giant* (gt), as shown in the diagram of Fig. 5 where *tailess* (tll) also appears as gap genes whose regulation we do not represent here [1,9].

4.2 The model

The model aims at reproducing the expression pattern of the gap genes, before the pair-rule genes are activated. In [15] this phenomenon is modelled through a mathematical model – the reaction-diffusion partial differential equation – and experimental data are used in order to estimate the model parameters.

We used the experimental data available online in the FlyEx database [1]. The data contains quantitative wild-type concentration profiles for the protein products of the seven genes – *bcd, cad, hb, Kr, kni, gt, tll*—during cleavage cycles 11 up to 14A which constitute the blastoderm stage of *Drosophila* development. These data are used to validate the model dynamic. Expression data from cleavage cycle 11 are used as the initial condition—see Fig 4. The concentration

[1] `http://flyex.ams.sunysb.edu/flyex/index.jsp`

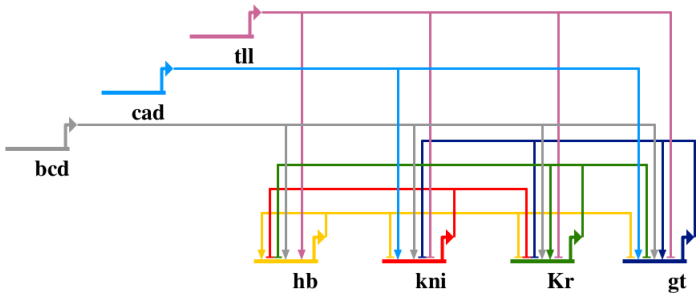

Fig. 5. Regulatory relationship as in our model and in [15,12]

of proteins are unitless, ranging from 0 to 255, at space point x, ranging from 0 to 100 % of embryo length.

Reaction rates for the gene regulatory network and for the protein synthesis and degradation has been taken from the Unc-GC model described in [15].

4.2.1 Intracellular reactions

The reactions that model the intracellular behaviour directly implement the graph of Fig. 5 which provides a snapshot of the regulatory interaction among genes in each cell. We use 0 or 1 as suffix in the genes name to express the genes activity – inactive and active respectively – and "p" or "g" prefix for representing the protein or gene form of the molecule type. The entities involved are expressed by instruction:

```
molecule M where (M in [pBcd, pCad, pHb, pKr, pGt, pKni, pTll, gHb0, gKr0, gGt0,
                        gKni0, gHb1, gKr1, gGt1, gKni1]).
```

The protein synthesis is assumed to be an atomic events, so that transcription and translation are modelled with one reaction only. Gene activation is modelled through a reaction that change the state of the gene from 0 to 1 once the activating protein is available. The other way round causes the gene inhibition. The model of *hb* dynamics is expressed by code (and similarly for the others):

```
% gene hb regulation
reaction r(gHbAct00)   : [pBcd, gHb0] --> [pBcd, gHb1] rate 0.1114.
reaction r(gHbAct01)   : [pTll, gHb0] --> [pTll, gHb1] rate 00144.
reaction r(gHbAct02)   : [pHb, gHb0] --> [pHb, gHb1] rate 0.0293.
reaction r(gHbDeAct00)  : [pKni, gHb1] --> [pKni, gHb0] rate 0.3903.
reaction r(gHbDeAct01)  : [pKr, gHb1] --> [pKr, gHb0] rate 0.0124.

% protein Hb synthesis and degradation
reaction r(pHbSynth) : [gHb1] --> [gHb1, pHb] rate 32.03.
reaction r(pHbDegr) : [pHb] --> [] rate 0.136.
```

4.2.2 Cell graph and cell-to-cell communication

We performed experiments with a 10x100 grid built as shown in Fig. 3, which allows molecules to diffuse in both the x and y axis. The horizontal axis represents the A-P position, while the vertical axis represents a portion of the D-V position, ranging from 45% to 55% of embryo's width.

Molecules diffuse crossing the cell's membrane and going into one of the four neighbouring cells piked up probabilistically. An example of such reactions is:

```
reaction r(pHbMove)   : [pHb] --> [firing(pHb)] rate 2.25.
```

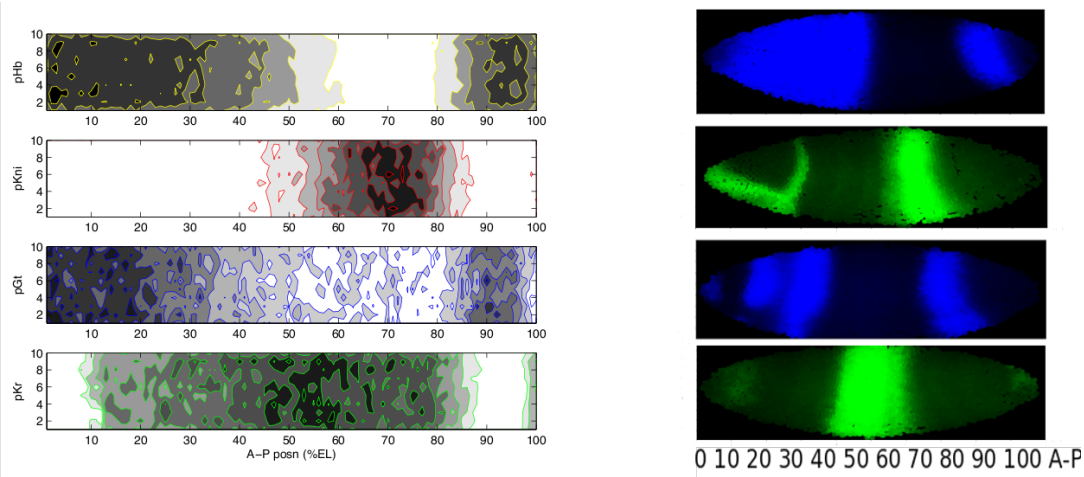

Fig. 6. Simulation results for the four gap genes *hb, kni, gt, Kr* at a simulation time equivalent to the eighth time step of cleavage cycle 14A (left) and the corresponding experimental data (right)—% A-P length on the x and % D-V width on the y

4.3 Simulation results

Results charted in the 2D grid are shown in Fig. 6 (left) for expression of *hb, kni, gt, Kr* at the eighth time step of cleavage cycle 14A. Experimental data are also provided in Fig. 6 (right) with 2D Atlas reconstructing the expression level of the four genes in A-P sections of the embryo. A qualitative comparison shows that the expression pattern of genes *hb, kni, gt* and *Kr* nicely fit the spatial distribution shown in the experimental data: *hb* is expressed in the extreme left pole until about 45% of embryo length and on the right between about 85% and 95%; *kni* is expressed mostly between 65% and 75%; *gt* is correctly reproduced on the left, but it loses precision on the right where its expression slightly overlaps *hb*; while *Kr* properly appears between 40% and 60%. This shows that the proposed framework smoothly allows to check the qualitative validity of our working model against the sought embryogenesis phenomenon.

5 Related Works

A good deal of work in the research field of Computational System Biology (CSB) has been moving towards the ability of addressing the scenarios described in this paper, as we witness a trend moving from the single, global solution idea of e.g. stochastic π-calculus [17] and κ-calculus [6,5] , to mechanisms and constructs tackling the multi-compartment scenario. In [19] the $S\pi@$ process calculus is introduced to deal with the notion of compartments (possibly with variable volumes), by adding to the stochastic π calculus the idea that process-molecules are situated into a location. In [3] Bio-PEPA has been extended for expressing hierarchies of locations with different sizes so that to model compartments, membrane and cell intra-compartment and inter-compartment reactions. In *Beta-binders* and its extension called BlenX, systems are modelled as a set of boxes representing biological entities at different levels – proteins, cells – and are simulated on top of BetaWB [7]. A model in *Membrane computing* [14], formally called P systems, consists of a membrane that contains a multi-set of objects

(representing chemical substances) that evolve according to given evolution rules (representing reactions).

The ability of modelling biochemical networks is also offered by simulation tools for kinetic modelling of biochemistry – see a survey in [2] – which provide a direct view of chemical reactions (differently from the indirect one of process algebras, which use process channels). They tackle the problem mostly at the level of Graphical User Interfaces, with little support to flexibility in expressing large-scale and dynamic networks at the language level. Most notably, an advantage of process calculi is that they are designed to retain Turing expressiveness, hence potentially allowing to structure any complex simulation scenario—though practically this can be hard, requiring a higher level language to be compiled into the process language. Moreover, they are based on mathematical models (ODE, PDE), and are mainly focused on intracellular networks, with few exceptions such as CellDesigner, CellWare, COPASI, Dizzy, GEPASI, JDesigner, Virtual Cell, MesoRD which support the representation of intracellular compartments.

It is worth reminding that the mentioned languages and frameworks are not conceived to address systems composed by a huge number of interacting cells. In particular, why it is still possible to use e.g. SPIM or Bio-PEPA to model scenarios like the one studied in this paper (featuring a network of 1000 cells), it would require a huge specification that is simply impractical to produce by hand. The work in this paper proceeds precisely in this direction: our front-end compiler can be considered as a tool useful to automatically create by expansion all the structures needed to feed a simulation engine with a large and complex biochemical network. Other simulators, like e.g. SPIM, Bio-PEPA, or BetaWB, could have been considered as an alternative target engine for our approach, but the advantages in doing so are currently unclear, though they are subject of current investigations.

6 Conclusions

Developmental biology calls for modelling and simulation tools that can effectively and efficiently support the analysis of biochemical systems featuring multiply-nested, dynamic networks of compartments. As this is a long term goal for the CSB research field, we believe that it is key to start considering computer frameworks that not only support simulation, but also focus on finding suitable computer languages to express systems that, in fact, can become quite as complex as software can be.

Accordingly, this paper provides the following contributions: *(i)* a simple and coherent computational model to structure biochemical networks of compartments, *(ii)* a language to express articulated systems in a simple and flexible way, *(iii)* a corresponding simulation engine based on known optimisation techniques [8]. Although this framework is limited to the case of static networks, it already allows to experiment on rather large scenarios of embryo- and morphogenesis in a way which we believe is more expressive and flexible with respect to existing tools.

In order to demonstrate the framework applicability, we studied the phe-

nomenon of pattern formation during *Drosophila* embryo development, modelling the interactions between maternal factors and gap genes that originate the early regionalisation of the embryo. The possibility to model both the reactions taking place inside the cells that regulate the gene expressions, and the molecules diffusion that mediates the cell-to-cell communication, allows the reproduction of the interplay between these two levels in order to verify its fundamental role in the spatial self-organisation characteristic of such a kind of phenomenon. The results presented show the formation of a precise spatial pattern which have been successfully compared with observations acquired from the real embryo gene expressions.

Future works will be devoted to better estimate parameters (also relying on evolutionary techniques as in [7]), so as to promote a more precise mimic – even from the quantitative point of view – of the expected behaviour, and extend exploration to other stages of embryogenesis, up to the long-term goal of simulating/predicting larger portions of embryogenesis. Moving towards larger time windows will require a dynamic topology of the cell network: the assumption made in this work that few and mostly non-relevant topological changes occur during the considered period – so that it is reasonable to assume a fixed topology – will in fact decay. Therefore future works will go in the direction of making it possible to simulate large scale dynamic network of cells, where they can move, divide and die. Unfortunately even if the language is ready for such an extension thanks to its flexibility [13], implementing dynamic topologies is quite a complex task at the engine level, and will require an important effort. As the model will become more and more complex, it will then be necessary to consider other extensions of the basic Gillespie's SSA – such as tau-leaping [11] – in order to maintain the good performance of the engine. Finally future works will be devoted to develop our tool up to a public release.

References

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland Science Textbooks. Garland Science, 4th edition, June 2002.

[2] R. Alves, F. Antunes, and A. Salvador. Tools for kinetic modeling of biochemical networks. *Nature Biotechnology*, 24(6):667–672, June 2006.

[3] F. Ciocchetta, A. Duguid, and M. L. Guerriero. A compartmental model of the cAMP/PKA/MAPK pathway in Bio-PEPA. *CoRR*, abs/0911.4984, 2009.

[4] A. Credi, M. Garavelli, C. Laneve, S. Pradalier, S. Silvi, and G. Zavattaro. nanok: A calculus for the modeling and simulation of nano devices. *Theoretical Computer Science*, 408(1):17–30, 2008.

[5] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling of cellular signalling. In *CONCUR*, pages 17–41, 2007.

[6] V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[7] L. Dematté, C. Priami, A. Romanel, and O. Soyer. Evolving blenx programs to simulate the evolution of biological networks. *Theoretical Computer Science*, 408(1):83–96, 2008.

[8] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, March 2000.

[9] S. F. Gilbert. *Developmental Biology, Eighth Edition*. Sinauer Associates Inc., Eighth edition, Mar. 2006.

[10] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[11] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[12] V. V. Gursky, J. Jaeger, K. N. Kozlov, J. Reinitz, and A. M. Samsonov. Pattern formation and nuclear divisions are uncoupled in drosophila segmentation: comparison of spatially discrete and continuous models. *Physica D: Nonlinear Phenomena*, 197(3-4):286–302, October 2004.

[13] S. Montagna and M. Viroli. A computational framework for modelling multicellular biochemistry. In *IEEE CEC09 Preceedings*, Trondheim, Norway, 18–21 May 2009.

[14] G. Paun. *Membrane Computing: An Introduction.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[15] T. J. Perkins, J. Jaeger, J. Reinitz, and L. Glass. Reverse engineering the gap gene network of *Drosophila Melanogaster*. *PLoS Computational Biology*, 2(5):e51, 05 2006.

[16] A. Phillips. The Stochastic Pi Machine (SPiM), 2007. Version 0.05 available online at http://research.microsoft.com/~aphillip/spim/.

[17] C. Priami. Stochastic pi-calculus. *The Computer Journal*, 38(7):578–589, 1995.

[18] A. M. Uhrmacher, D. Degenring, and B. Zeigler. Discrete event multi-level models for systems biology. In C. Priami, editor, *Transactions on Computational Systems Biology I*, volume 3380 of *Lecture Notes in Computer Science*, pages 66–89. Springer, 2005.

[19] C. Versari and N. Busi. Efficient stochastic simulation of biological systems with multiple variable volumes. *Electronic Notes in Theoretical Computer Science*, 194(3):165–180, 2008.

[20] D. Yamins and R. Nagpal. Automated global-to-local programming in 1-d spatial multi-agent systems. In *7th International Joint Conference on Agents and Multi-Agent Systems (AAMAS-08)*, pages 615–622, Estoril, Portugal, 12–16May 2008. IFAAMAS.