Towards Probabilistic Model Checking on P Systems using PRISM

Francisco J. Romero-Campero[a], Marian Gheorghe[b],
Luca Bianco[c], Dario Pescini[d], Mario J. Pérez-Jiménez[a], Rodica Ceterchi[e]

[a]Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Seville, Avda. Reina Mercedes, 41012 Sevilla, Spain
Email: {fran,marper}@cs.us.es

[b]Department of Computer Science, The University of Sheffield
Regent Court, Portobello Street, Sheffield S1 4DP, UK
Email: M.Gheorghe@dcs.shef.ac.uk

[c]Department of Computer Science, University of Verona
Strada Le Grazie 15, 37134 Verona, Italy
Email:bianco@sci.univr.it

[d]Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
Email: pescini@disco.unimib.it

[e]University of Bucharest, Faculty of Mathematics and Computer Science
Academiei 14, 70109 Bucharest, Romania
Email: rc@funinf.cs.unibuc.ro

Abstract. In this paper it is presented the use of P systems and π-calculus to model interacting molecular entities and how they are translated into a probabilistic and symbolic model checker called PRISM.

1 Introduction

The complexity of biomolecular cell systems is currently the focus of intensive experimental research, nevertheless the enormous amount of data about the function, activity, and interactions of such systems makes necessary the development of models able to provide a better understanding of the dynamics and properties of the systems. A model, an abstraction of the real-world onto a mathematical/computational domain, highlights some key features while ignoring others that are assumed to be not relevant. A good model should have four properties: relevance, computability, understandability and extensibility, [20]. A model must be relevant capturing the essential properties of the phenomenon investigated; and computable so it can allow the simulation of its dynamic behaviour, and the qualitative and quantitative reasoning about its properties. An understandable model will correspond well to the informal concepts and ideas of molecular biology. Finally, a good model should be extensible to higher levels of organisations, like tissues, organs, organism, etc, in which molecular systems play a key role.

In this paper we will deal with models developed within the framework of membrane computing. Membrane computing is an emergent branch of natural computing introduced by G. Păun in [15]. This new model of computation starts from the assumption that the processes taking place in the compartmental structure of a living cell can be interpreted as computations. The devices of this model are called P systems. Roughly speaking, a P system consists of a cell-like membrane structure, in the compartments of which one places multisets of objects which evolve according to given rules.

Although most research in P systems concentrates on the computational power of the devices involved, lately they have been used to model biological phenomena within the framework of computational systems biology. In this case P systems are not used as a computing paradigm, but rather as a formalism for describing the behaviour of the system to be modelled. In this respect several P systems models have been proposed to describe oscillatory systems [8], signal transduction [17], gene regulation control [16], quorum sensing [12, 18, 21] and metapopulations [19]. These models differ in the type of the rewriting rules, membrane structure and the strategy applied to run the rules in the compartments defined by membranes. Some of these models using *metabolic algorithm* [5], *dynamical probabilistic P systems* [19] and *(multicompartmental) Gillespie Algorithm* [17] were applied in certain case studies.

As P systems are inspired from the structure and functioning of the living cell, it is natural to consider them as modelling tools for biological systems, within the framework of systems biology, being an alternative to more classical approaches like ordinary differential equations (ODEs) and to some recent approaches like Petri nets and π-calculus. Differential equations have been used successfully to model kinetics of conventional macroscopic chemical reactions where the main focus is on the average evolution of the concentration of chemical substances across the whole system. Nevertheless, there is an implicit assumption of continuously varying chemical concentration and deterministic dynamics. Two critical characteristics of this approach are that the number of molecules of each type in the reaction mix is large and that for each type of reaction in the system, the number of reactions is large within each observation interval, that is reactions are fast.

When the number of particles of the reacting species is small and reactions are slow, which is frequently the case in some biological systems, both of the previous assumptions are questionable and the deterministic continuous approach to chemical kinetics should be complemented by an alternative approach. In this respect, one has to recognise that the individual chemical reaction steps occur discretely and are separated by time intervals of random length. Stochastic and discrete approaches like the ones used with Petri nets [11], π-calculus [20] and P systems [17, 19] are more accurate in this situation. Nevertheless, these formalisms differs in some essential features that will be discussed briefly in this paper.

Most research in systems biology focuses on the development of models of different biological systems in order to be able to simulate them, accurate enough

such as to be able to reveal new properties that can be difficult or impossible to discover through direct experiments. One key question is what one can do with a model, other than just simulate trajectories? This question has been considered in detail for deterministic models, but less for stochastic models. Stochastic systems defy conventional intuition and consequently are harder to conceive. The field is widely open for theoretical advances that help us to reason about systems in greater detail and with finer precision.

An attempt in this direction consists in using model checking tools to analyse in an automatic way various properties of the model. There are previous studies investigating the use of model checking for P system specifications [2, 7].

Our current attempt uses a probabilistic symbolic model checking approach based on PRISM (Probabilistic and Symbolic Model Checker) [22] and investigates continuous time P systems with Gillespie dynamics using protein-protein interaction rules.

Systems consisting of interacting molecular entities have been modelled by using π-calculus formalism [20] explaining the principles of transforming the biological system into a π-calculus model in a coherent way.

In this paper it is shown how π-calculus and P systems can model systems consisting of reactions with biochemical entities. The specification is translated into PRISM and various properties are studied. Some simulations obtained using the the PRISM simulator as well as a P system simulator with Gillespie dynamics are presented.

The paper is organised as follows: in section 2 a brief overview of PRISM is presented; section 3 deals with P system specifications in PRISM, section 4 presents a case study representing the cell cycle in eukaryotes described using a P system specification and a π-calculus definition; both are then translated into PRISM and contrasted in section 5; conclusions are drawn in section 6.

2 PRISM

Probabilistic model checking is a formal verification technique. It is based on the construction of a precise mathematical model of a system which is to be analysed. Properties of this system are then expressed formally using temporal logic and analysed against the constructed model by a probabilistic model checker.

PRISM, the probabilistic and symbolic model checker in this study, supports three different types of probabilistic models, discrete time Markov chains (DTMC), Markov decision processes (MDP) and continuous time Markov chains (CTMC). PRISM supports systems specifications through two temporal logics, PCTL (probabilistic computation tree logic) for DTMC and MDP and CSL (continuous stochastic logic) for CTMC.

In order to construct and analyse a model with PRISM, it must be specified in the PRISM language, a simple, high level, state-based language based on the Reactive Modules formalism of [1].

Here we describe some aspects of the PRISM language through the following illustrative example taken from [22].

```
// N-place queue + server

ctmc

const int N = 10;
const double mu = 1/10;
const double lambda = 1/2;
const double gamma = 1/3;

module queue
      q : [0 .. N] init 0;

      [] q < N -> mu : (q' = q + 1);
      [] q = N -> mu : (q' = q);
      [serve] q > 0 -> lambda : (q' = q - 1);

endmodule

module server
      s : [0 .. 1] init 0;

      [serve] s = 0 -> 1 : (s' = 1 );
      [] s = 1 -> gamma : (s' = 0);

endmodule
```

The fundamental components of the PRISM language are modules and variables. A model is composed of a number of modules which can interact with each other. A module contains a number of local variables and commands.

The previous example consists of two modules; the first one represents a `queue` and the second one represents a `server`.

A module is specified as:

```
module ⟨name⟩

endmodule
```

Note that, in the example above, there are only two local variables, `q` in the `queue` module representing the size of the queue, and `s` in the `server` module which represents whether or not the server is busy. In the declaration of a variable its initial value and range must be specified. A variable declaration looks like:

```
name : [ lower-bound .. upper-bound ] init value;
```

The values of these variables at any given time constitute the states of the module. The space of reachable states is computed using the range of each variable and its initial value. The global state of the whole model is determined by the local state of all modules.

The behaviour of each module is described by a set of commands. A command takes the form:

$$\texttt{[action]}\ \mathtt{g} \rightarrow \lambda_{\mathbf{1}} : \mathbf{u_1} + \cdots + \lambda_{\mathbf{n}} : \mathbf{u_n};$$

The guard `g` is a predicate over all the variables of the model. Each update $\mathbf{u_i}$ describes the new values of the variables in the module specifying a transition of the module. The expressions $\lambda_{\mathbf{i}}$ are used to assign probabilistic information, rates, to transitions.

The label `action` placed inside the square brackets are used to synchronise the application of different commands in different modules. This forces two or more modules to make transitions simultaneously. The rate of this transition is equal to the product of the individual rates, since the processes are assumed to be independent.

In our example, in the `queue` module there are three commands; the first one allows a new client to join the queue with probability `mu` if the maximal size, `N`, has not been reached yet; otherwise the second command maintains the size of the queue constant with probability `mu`. The third command is synchronised with the first command of the `server` module and describes the situation when there are clients in the queue and the server is free; in this case with rate `lambda` the server is set to busy and one client is removed from the queue. Observe that the rate of this transition is equal to the product of the two individual rates ($1 \times$ `lambda` = `lambda`), this is a common technique, an action is *passive* with rate 1 and the other action *active* which actually defines the rate for the synchronised transition.

PRISM supports many other features like constants, expressions, process algebra operators, etc. For a detailed description of the tool we refer to [22].

3 Transforming P system Specification into PRISM

The main components of a P system are a membrane structure consisting of a number of membranes that can interact with each other, an alphabet of objects and a set of rules associated to each membrane. These components can easily be mapped into the components of the PRISM language using modules to represent membranes, variables to describe the alphabet and commands to specify the rules.

A P system is a construct

$$\Pi = (\Sigma, L, \mu, M_1, M_2, \ldots, M_n, R_1, \ldots, R_n)$$

where:

- Σ is a finite alphabet of symbols representing objects;
- L is a finite alphabet of symbols representing labels for the compartments[1];
- μ is a membrane structure containing $n \geq 1$ membranes labelled with elements from L;
- $M_i = (l_i, w_i)$, for each $1 \leq i \leq n$, is the initial configuration of membrane i with $l_i \in L$ and $w_i \in \Sigma^*$ a finite multiset of objects;
- R_i, for each $1 \leq i \leq n$, is a finite set of rules in membrane i of the form specified below with objects in Σ and labels in L.

The types of rules we will consider in this paper are those referred in the literature as protein-protein interaction rules.

- Transformation, complex formation and dissociation rules:

$$[\ a\ ]_l \xrightarrow{c} [\ b\ ]_l$$

$$[\ a,\ b\ ]_l \xrightarrow{c} [\ e\ ]_l \quad \text{where } a, b, e \in \Sigma \text{ and } l \in L$$

$$[\ a\ ]_l \xrightarrow{c} [\ b,\ e\ ]_l$$

These rules are used to specify chemical reactions taking place inside a compartment of type $l \in L$, more specifically they represent the transformation of a into b, the formation of a complex e from the interaction of a and b, and the dissociation of a complex a into b and e respectively. These types of rules are used for example in [5] to describe oscillations as a consequence of the interactions between different objects inside a single compartment.

- Diffusing in and out:

$$\begin{array}{l} [\ a\ ]_l \xrightarrow{c} a\ [\ \ ]_l \\ \qquad\qquad \text{where } a \in \Sigma \text{ and } l \in L \\ a\ [\ \ ]_l \xrightarrow{c} [\ a\ ]_l \end{array}$$

When chemical substances move or diffuse freely from one compartment to another one we use these types of rules, where a moves from or to a compartment of type l.
These rules are also used to model metapopulations [19] where individuals can move from one compartment to another one or signal molecules occurring in bacteria [17] using population P systems [4] as a model.

- Binding and debinding rules:

$$\begin{array}{l} a\ [\ b\ ]_l \xrightarrow{c} [\ e\ ]_l \\ \qquad\qquad \text{where } a, b, e \in \Sigma \text{ and } l \in L \\ [\ a\ ]_l \xrightarrow{c} b\ [\ e\ ]_l \end{array}$$

[1] Two membranes with the same label will be considered to be of the same *type*

Using rules of the first type we can specify reactions expressing the binding of a ligand swimming in one compartment to a receptor placed in the membrane surface of another compartment. The reverse reaction, debinding of a substance from a receptor, can also be described by using the second rule. These rules were used to model signalling at the cell surface in [17].

– Recruitment and releasing rules:

$$\begin{array}{l} a\,[\,b\,]_l \xrightarrow{c} e\,[\;\;]_l \\ \\ e\,[\;\;]_l \xrightarrow{c} a\,[\,b\,]_l \end{array} \qquad \text{where } a, b, e \in \Sigma \text{ and } l \in L$$

With these rules we represent the interaction between two chemicals in different compartments whereby one of them is recruited from its compartment by a chemical on the other compartment, and then the new complex remains in the latter compartment. In a releasing rule a complex, e, located in one compartment can dissociate into a and b, with a remaining in the same compartment as e, and b being released into the other compartment. In [17], these rules were used to describe the signal transduction between environmental concentrations of signal molecules and the cytoplasmic concentrations of different kinases.

Here, in order to capture the features of all these rules, we consider generic rules of the form:

$$u\,[\,v\,]_l \xrightarrow{c} u'\,[\,v'\,]_l \tag{1}$$

with u, v, u', v' some finite multisets of objects and l the label of a membrane. These rules are multiset rewriting rules that operate on both sides of the membranes, that is, a multiset u placed outside a membrane labelled by l and a multiset v placed inside the same membrane can be simultaneously replaced by a multiset u' and a multiset v' respectively. In this way, we are able to capture in a concise way the features of both communication rules (diffusion, binding, debinding etc ...) and transformation rules considered before. This generic type of rules was referred as *boundary rules* in [3].

We also associate to each rule a stochastic constant, c, which will be used to compute the probability of applying a rule in a given configuration, see [17]. This is necessary to characterise the *reality* of the phenomenon to be modelled. The necessity of taking into account these quantitative aspects has been made clear in a few recent studies regarding the use of P systems to model biological systems.

In what follows we will describe how to specify P systems models in the PRISM language.

First of all, since we work with continuous time P systems with Gillespie dynamics our model will be declared as a CTMC using the key word `stochastic`.

The membranes occurring in the membrane structure will be represented using modules and the topology according to which membranes communicate or interact will be coded in the commands of each module.

Each module will describe the behaviour of one membrane by representing the rules associated to it using commands and the objects placed in it using local variables.

More specifically, given $\Pi = (\Sigma, L, \mu, M_1, M_2, \ldots, M_n, R_1, \ldots, R_n)$ a P system as before, each membrane in μ will be uniquely identified with an identifier i, $1 \leq i \leq n$.

- Each membrane i will be specified using a module which will be called `compartment_i`.
- For each object $o \in \Sigma$ that can be present inside the compartment defined by membrane i a local variable `o_i` will be declared in module `compartment_i`. The initial value of the variable will be given by the corresponding initial multiset w_i; its value range will be determined experimentally or it will be derived from the literature.
  A constant `o_i_bound` representing the upper bound of the object `o_i` will be declared to specify the value range.
- To describe the rules in R_i commands will be used. We will focus on the generic type of rule in (1). In general, these rules need two membranes to interact in a synchronised way to exchange objects. In this case the two modules representing the corresponding membranes will synchronise the application of two different commands by using the label `rule_k`, where k is the index of the rule being specified.

  Therefore, assuming that compartment i is contained in compartment j and given a rule of the form

$$a_1, \ldots, a_p \, [\, b_1, \ldots, b_r \,]_i \xrightarrow{c} c_1, \ldots, c_s [\, d_1, \ldots, d_t \,]_i$$

  The command in module `compartment_j` will be:

```
[rule_k] a1_j > 0 & ... & ap_j > 0 &
         c1_j < c1_j_bound & ... & cs_j < cs_j_bound ->
         c * a1_j * ... *ap_j :
          (a1_j' = a1_j - 1) & ... & (ap_j' = ap_j - 1) &
          (c1_j' = c1_j + 1) & ... & (cs_j' = cs_j + 1);
```

  The command in module `compartment_i` will be:

```
[rule_k] b1_i > 0 & ... & br_i > 0 &
         d1_i < d1_i_bound & ... & dt_i < dt_i_bound ->
         b1_i * ... * br_i :
          (b1_i' = b1_i - 1) & ... & (br_i' = br_i - 1) &
          (d1_i' = d1_i + 1) & ... & (dt_i' = dt_i + 1);
```

  Observe that these two commands are applied when the guards hold, that is, if and only if there are some reactants in the corresponding membranes and

the products have not reached the upper bounds determined experimentally. Also note that the rate of this transition is the product of the individual rates (`c * a1_j * ... * ap_j`) * (`b1_i * ... * br_i`). Here we assume that the objects a's and b's are different, if we have an object with multiplicity greater than one present on the left hand side of the rule the rate associated to the command will be different and it will be computed as it is explained in [10].
When this transition is performed the local variables representing the reactants are decreased by one and the variables representing the products are increased by one.
Finally, note that although the rules of the general form in (1) require synchronisation between two modules representing membranes, in the particular case of transformation, complex formation and dissociation rules only one membrane is involved and no synchronisation is needed. Given a rule of type:

$$[\ a_1, \ldots, a_p\ ]_l \xrightarrow{c} [\ b_1, \ldots, b_r\ ]_l \tag{2}$$

the PRISM specification will be as follows:

```
[] a1_i > 0 & ... & ap_i > 0 &
   b1_i < b1_i_bound & ... & br_i < br_i_bound ->
      c * a1_i * ... * ap_i :
        (a1_i' = a1_i - 1) & ... & (ap_i' = ap_i - 1) &
        (b1_i' = b1_i + 1) & ... & (br_i' = br_i + 1);
```

4 Cell cycle in Eukaryotes - a case study

In this section two different models of the cell cycle in eukaryotes will be presented and contrasted using PRISM and our simulator of P systems with Gillespie dynamics available from [25]. The first model is expressed using a P system specification with a single compartment whereas the second one [13] uses a π-calculus approach.

The cell division cycle in eukaryotes is a coordinated set of processes whereby a cell replicates all its components and divides into two nearly identical daughter cells. These processes are controlled by a complex network consisting of cyclin-dependent kinase (CDK) and its corresponding cyclin. Kinases, cdc14, and phosphatases, cdh1, regulate CDK activity. There are also stoichiometric inhibitors (CKI), that sequester cyclin-CDK dimers inhibiting their activity.

4.1 A P system model

Our P system model is given by,

$$\Pi = (\Sigma, \{1\}, [\ ], (1, w), R)$$

where:

- Σ contains all the protein and complexes of proteins involved in the system:
  $\Sigma = \{cdk,\ cyclin,\ cdk.cyclin,\ cdk.degc,\ cki, cdk.cyclin.cki,\ cdh1,$
  $cdh1off,\ cdc14,\ cdc14off\}$
- the membrane structure has only one component that will be labelled by 1.
- w is the initial multiset of objects (molecules) which determines the initial configuration of the system,

$$w = cdk^{100}\ cyclin^{200}\ cki^{100}\ cdh1^{100}\ cdc14^{200}$$

- the rules of R describe the interactions taking place in the cell cycle control[2]:
  $r_1 : [\ cdk,\ cyclin\ ]_1 \xrightarrow{c_1} [\ cdk.cyclin\ ]_1 \quad c_1 = 0.5$
  cdk is activated upon binding with its corresponding $cyclin$ producing the dimer $cdk.cyclin$.
  $r_2 : [\ cdh1,\ cdk.cyclin\ ]_1 \xrightarrow{c_2} [\ cdh1,\ cdk.degc\ ]_1 \quad c_2 = 0.005$
  $r_3 : [\ cdk.degc\ ]_1 \xrightarrow{c_3} [\ cdk\ ]_1 \quad c_3 = 0.001$
  These two rules describe how $cdh1$ regulates the activity of the dimers $cdk.cyclin$ by degrading the bound cyclin.
  $r_4 : [\ cdk.cyclin,\ cki\ ]_1 \xrightarrow{c_4} [\ cdk.cyclin.cki\ ]_1 \quad c_4 = 0.003$
  $r_5 : [\ cdk.cyclin.cki\ ]_1 \xrightarrow{c_5} [\ cdk.cyclin,\ cki\ ]_1 \quad c_5 = 0.3$
  cki also inhibits the activity of the dimers $cdk.cyclin$ by forming the triplets $cdk.cyclin.cki$.
  $r_6 : [\ cdh1,\ cdk.cyclin\ ]_1 \xrightarrow{c_6} [\ cdh1off,\ cdk.cyclin\ ]_1 \quad c_6 = 0.005$
  $cdh1$ can also be inhibited by $cdk.cyclin$ dimers.
  $r_7 : [\ cdh1off,\ cdc14\ ]_1 \xrightarrow{c_7} [\ cdh1,\ cdc14off\ ]_1 \quad c_7 = 0.009$
  $r_8 : [\ cdh1,\ cdc14off\ ]_1 \xrightarrow{c_8} [\ cdh1,\ cdc14\ ]_1 \quad c_8 = 0.009$
  $cdh1$ and $cdc14$ regulate each other activity by inhibition and activation.

Now we will specify the previous P system in the PRISM language using the algorithm described in section 3. Since Π is a continuous time P system with Gillespie dynamics, our model will be declared as being `stochastic`.

The PRISM model will have a single module, `compartment`, representing the only membrane present in the membrane structure of Π.

```
module compartment
    ⋮
endmodule
```

In this module we will describe the alphabet Σ and the initial multiset w using local variables. For example, cdk, $cyclin$ and the dimer $cdk.cyclin$ are specified as follows:

```
cdk_1 : [ 0 .. cdk_1_bound ] init 100;
cyclin_1 : [ 0 .. cyclin_1_bound ] init 200;
cdk.cyclin_1 : [ 0 .. cdk.cyclin_1_bound] init 0;
```

[2] The time units of the stochastic constants associated with the rules are minutes.

Finally the rules from R will be described using commands. We observe that since the model consists only of rules of the form in (2) no sychronisation is required in this case study.

$[cdk,\ cyclin]_1 \xrightarrow{c_1} [cdk.cyclin]_1 \quad c_1 = 0.5$

```
[] cdk_1 > 0 & cyclin_1 > 0  &
   cdk.cyclin_1 < cdk.cyclin_1_bound ->
       c1*cdk_1*cyclin_1 :
         (cdk_1' = cdk_1 - 1) & (cyclin_1' =  cyclin_1 - 1) &
         (cdk.cyclin_1' = cdk.cyclin_1 + 1);
```

$[cdk.cyclin.cki]_1 \xrightarrow{c_5} [cdk.cyclin,\ cki]_1 \quad c_5 = 0.3$

```
[] cdk.cyclin.cki_1 > 0 &
   cdk.cyclin_1 < cdk.cyclin_1_bound & cki_1 < cki_1_ bound ->
       c5*cdk.cyclin.cki_1 :
         (cdk.cyclin.cki_1' = cdk.cyclin.cki_1  - 1) &
         (cki_1' = cki_1 + 1) &
         (cdk.cyclin_1' = cdk.cyclin_1 + 1);
```

We have run simulations of the previous P system and PRISM specifications using our simulator of P system with Gillespie dynamics and the PRISM simulator. Next we depict the evolution of the number of objects representing the dimer $cdk.cyclin$, $cdh1$ and $cdc14$.

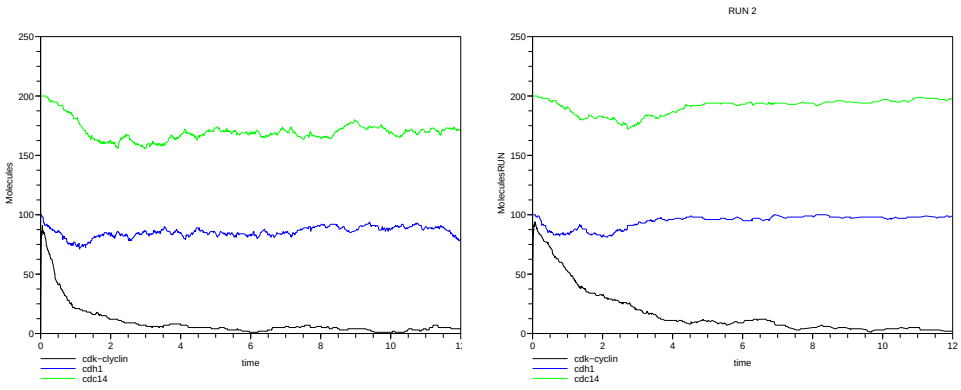

Fig. 1. Simulations using the PRISM simulator (first graph) and our simulator of P systems with Gillespie dynamics (second graph)

Observe, that although both simulators use the same strategy for the evolution of the two different models the simulations are not exactly the same. This is due to the fact that we are dealing with stochastic approaches.

Nonetheless, both runs show a sudden increase of the dimer *cdk.cyclin* reaching a peak of almost 100 molecules in less than a minute then the number of dimers decay steadily to zero. The evolution of *cdh*1 and *cdc*14 is similar, at the beginning both decay slightly and then they increase reaching a number of molecules approximately equal to the initial one.

4.2 A π-calculus model

Within the framework of π-calculus a system of interacting molecular entities is described and modelled by a system of concurrent communicating processes. Communication occurs on complementary channels, that are identified by specific names. Each molecule in the molecular system is described by a process and modifications due to chemical interactions are represented using communication and message passing between different processes through different channels, [20].

In what follows we comment on some fragments of a π-calculus specification of the same network of the cell cycle specified before using P systems. These fragments were taken from [13].

1. `SYSTEM ::= CYCLIN | CDK | CDH1 | CDC14 | CKI`

   First the molecular population is specified as a system of concurrent processes each one representing a molecule; `CYCLIN`, `CDK`, `CDH1`, `CDC14` and `CKI`. The following two fragments are examples of how chemical interactions are specified in π-calculus.

2. `CYCLIN ::= (v bb) BINDING-SITE`
   `BINDING-SITE ::=` $(\overline{lb}\langle$`bb`$\rangle$, $\mathtt{R}_4$`), CYCLIN-BOUND`
   `CDK ::= (lb(cbb),` $\mathtt{R}_4$`). CDK-CATALYTIC`

   The process `CYCLIN` is defined as another process `BINDING-SITE` with communication channel `bb`. The process `CDK` has the complementary channel to `BINDING-SITE`, $\overline{lb}\langle$`bb`$\rangle$, and so they can communicate, with rate $\mathtt{R}_4$, to produce two new processes `CYCLIN-BOUND` and `CDK-CATALYTIC`. This fragment correspond to rule r_1 of the P system specification.

3. `CKI ::= DEGRCKI + BINDCYC`
   `BINDCYC ::= (bind(x),`$\mathtt{R}_{11}$`).0`
   `CYC-CDK-CKI ::=` $(\overline{bind}\langle$`bb`$\rangle$, $\mathtt{R}_{11}$`).TRIM`

   This fragment is similar to the previous one. `CKI` can be replaced either by the process `DEGRCKI` or `BINDCYC` nondeterministically. The process `BINDCYC` can communicate with `CYC-CDK-CKI` to produce `TRIM`, process representing the trimer. This fragment correspond to rule r_4 of the P system specification.

The above π-calculus specification may be translated into PRISM [22]. Below the parts corresponding to the fragments of the specification presented before

are given. The π-calculus specification was mapped into PRISM using modules to describe processes, commands labels correspond to communication channels, variables value represent the number of different processes and commands specify the effect of a communication.

```
module cyclin
    cyclin : [0..CYCLIN] init CYCLIN;
    cyclin_bound : [0..CYCLIN] init 0;
    trim : [0..CYCLIN] init 0;
    [lb] cyclin>0 & cyclin_bound<CYCLIN
        -> cyclin : (cyclin_bound'=cyclin_bound+1) &
                    (cyclin'=cyclin-1);
    [bind] cyclin_bound>0 & trim<CYCLIN
        -> cyclin_bound : (trim'=trim+1) &
                          (cyclin_bound'=cyclin_bound-1);
endmodule
```

The module above describes the processes representing molecules of type cyclin. The variables `cyclin`, `cyclin_bound` and `trim` represent the number of processes `CYCLIN`, `CYCLIN_BOUND` and `TRIM` of the previous π-calculus specification. Below the modules describing the molecules of type `cdk` and `cki` are presented.

Observe that the command labels `lb` and `bind` specify the communication channels and the effect of these communications are described in the corresponding commands. For example, the label `lb` synchronise the first commands in modules `cyclin` and `cdk`, where the number of processes of type `cyclin` is decreased by one and the number of processes of type `cyclin_bound` and `cdk_cat` are increased by one representing the removal of a process `CYCLIN` and the creation of two new processes `CYCLIN_BOUND` and `CDK-CATALYTIC`.

```
module cdk
    cdk : [0..CDK] init CDK;
    cdk_cat : [0..CDK] init 0;
    [lb] cdk>0 & cdk_cat<CDK
        -> cdk : (cdk_cat'=cdk_cat+1) & (cdk'=cdk-1);
endmodule
```

The fragment 3 of the π-calculus specification is described in the next module and the creation of a process `TRIM` is represented using the commands labelled by `bind` which decrease by one the variable `cki` representing the number of `CKI` processes whereas in the module `cyclin` the variable `trim` is increased by one.

```
module cki
    cki : [0..CKI] init CKI;
    [bind] cki>0 -> cki : (cki'=cki-1);
endmodule
```

Using the PRISM simulator simulations of the previous π-calculus model can be obtained [22]. The evolution of the number of processes `CYCLIN`, `CDH1` and `CDC14` is depicted below. Note the similarity between this graph and the ones in figure (1).

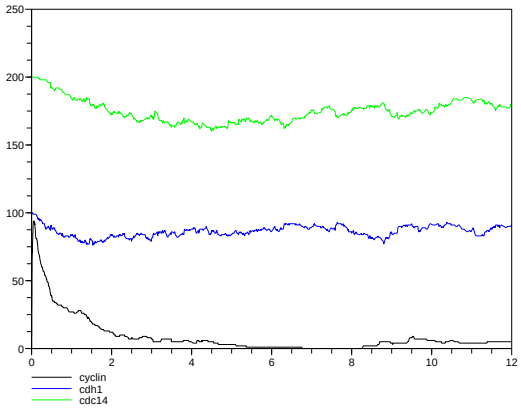

Fig. 2. A Simulation of the π-calculus model

5 Some Results and Discussions

We deal with systems of interacting biochemical entities. In this section we will show how these systems are modelled using π-calculus and P systems formalism and how these are represented in PRISM.

From the previous case study modelled by using π-calculus and P systems approaches we can identify some general principles of representing in the formalisms different aspects of the modelling process.

In P systems molecules are represented using objects from a finite alphabet and therefore the molecular population present in the system is specified by multisets of objects. In π-calculus a concurrent and communicating process abstracts the behaviour of a molecule and the whole molecular system is specified as a system of concurrent and communicating processes.

Compartments are explicitly specified in P systems as regions delimited by membranes. Although in π-calculus there is no component that corresponds directly with a biological compartment, processes representing molecules in the

same compartment share certain exclusive communication capabilities (private communication channels), that are inaccessible to processes representing molecules in other compartments.

Regarding biochemical reactions, in P systems they are specified using rewriting rules according to which objects representing reactants are replaced with objects representing products. In the π-calculus approach different processes representing molecules interact through complementary communication channels; this communication and the changes triggered by it describe a chemical interaction between the molecules represented using these processes.

The translocation of a molecule from one compartment to another one is modelled in P system using a particular type of rewriting rule, called boundary rule, where the compartments involved in the movement are specified. In the case of π-calculus mobile communication in which communication channel names are sent as messages is used to describe the movement of a process from one compartment to another using the extrusion of a private channel's scope.

In the following two tables we sum up how biomolecular systems are specified in P systems and in π-calculus and how these specifications can be mapped into PRISM so that we can perform probabilistic model checking on them.

Biomolecular entity	**P system entity**	**π-calculus entity**
Molecule	Object	Process
Molecular Population	Multiset of objects	System of concurrent processes
Compartment	Region defined by a membrane	Private Communication channels
Biochemical Transformation	Re-writing rule	Communication through channels
Compartment Translocation	Boundary rule	Extrusion of a private channel

Biomolecular entity	**P system PRISM**	**π-calculus PRISM**
Molecule	Variable	Module
Molecular Population	Variable values	Varibales in modules
Compartment	Module	Command Labels
Biochemical Transformation	Command	Sychronisation between commands
Compartment Translocation	Synchronisation between commands	Sychronisation between commands

One of the key features of our approach is the explicit stochastic behaviour of our models. As mentioned before stochastic systems defy conventional intuition and consequently are harder to conceive. The main stream methodology

to get the statistics of a stochastic system is the simulation of many trajectories. Nevertheless, simulation is the exploration of finite behaviours over given time intervals, whereas probabilistic model checking allows us to investigate the truth or otherwise of temporal queries expressed in temporal logics over possibly infinite sets of behaviours over possibly unbounded time intervals [6].

In what follows we use CSL (Continuous Stochastic Logic) and PRISM to formulate and check three temporal biological queries against our model of the cell cycle in eukaryotes in order to illustrate what kind of properties may be checked.

First, we study the expected number of molecules of *cdk.cyclin*, *cdh*1 and *cdc*14. For this we associate to each state a reward representing the value of the variable which specifies the corresponding protein. Then we formulate the query regarding the expected value of that reward at the time instant `T`:

```
R = ? [ I = T ]
```

In the graph below we have plotted these expected values as T varies.

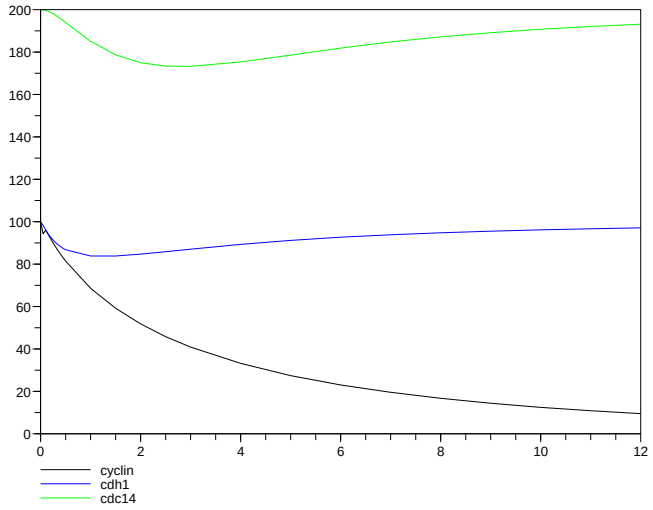

Fig. 3. Expected evolution of the P system specification

Observe that the expected number of molecules of dimers *cyclin.cdk* decreases steadily to low numbers whereas *cdh*1 and *cdc*14 decrease slightly during the first two minutes to start increasing gradually afterwards to reach approximately the initial number of molecules.

In order to get a finer grain analysis of the monotonic decrease of *cyclin.cdk* we investigate the probability that the dimers `cdk.cyclin` at time `T` is greater than a given threshold `t`. This property is specified by the CSL formula:

```
P = ? [ true U[T,T] cdk.cyclin >= t ]
```

In figure 4 it is depicted the probability that the number of dimers `cdk.cyclin` is greater than 90, 80 and 70 as time varies for the P system specification.

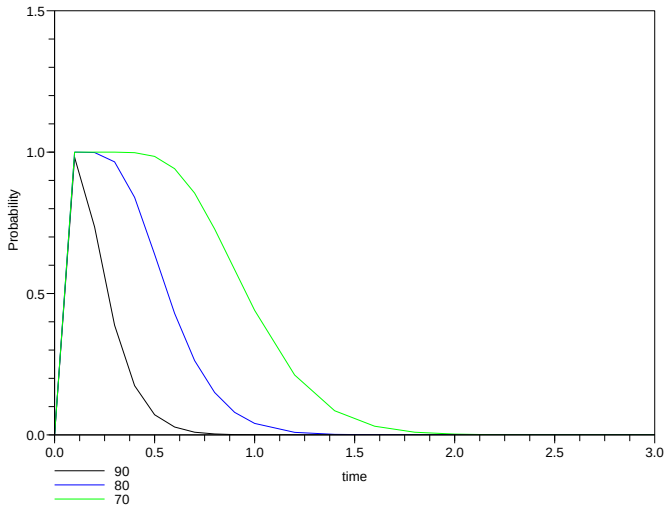

Fig. 4. Probability that the number of dimers cdk.cyclin is greater than a given threshold in the P system specification

Besides transient analysis we can also study steady state properties using the operator `S`. In order to verify that *cdh*1 and *cdc*14 return to their initial values in the long term we have verified the following temporal queries:

```
S >= 0.9 [ cdh1 = 100  ]
S >= 0.9 [ cdc14 = 200  ]
```

6 Conclusions and Future Work

In this paper we have discussed how P systems and π-calculus can model systems of interacting biochemical entities. A simple case study regarding the cell cycle has been used to illustrate the different methodologies. We have also shown how P systems and π-calculus specifications can be translated into PRISM allowing

us to perform probabilistic model checking; in this respect specific questions were addressed for the P system specification.

Future work will aim to compare through more complex case studies some differences between P system specifications and other modelling paradigms - Petri nets, cellular automata, ambient calculus, in order to understand advantages and limitations offered by these methods and their suitability for different problems.

References

1. Alur, R., Henzinger, T.A. (1999) Reactive Modules. *Formal Methods in System Design* **15** 7–48.
2. Andrei, O., Ciobanu, G., Lucanu, D. (2005) Executable Specifications of P Systems. In: *Proc. 5th Workshop on Membrane Computing.*
3. Bernardini, F., Manca, V. (2003) P Systems with Boundary Rules. *Lecture Notes in Computer Science*, **2597**,107–118.
4. Bernardini, F., Gheorghe, M. (2004) Population P Systems. *J. UCS* **10(5)** 509–539.
5. Bianco, L., Fontana, F., Manca, V. (2006) P Systems with Reaction Maps, *International Journal of Foundations of Computer Science*, **17** (1) 27–48.
6. Calder, M., Vyshemirsky, V., Gilbert, D, Orton, R. Analysis of Signalling Pathways using Continuous Time Markov Chains, *Transactions on Computational Systems Biology*, to appear.
7. Dang, Z., Ibarra, O.H., Li, C., Gaoyan, X. Decidability of Model-Checking P Systems. *Journal of Automata, Languages and Combinatorics,* 126–145.
8. Fontana, F., Bianco, L., Manca, V. (2005) P Systems and the Modeling of Biochemical Oscillations, Workshop on Membrane Computing, 199 – 208.
9. Gillespie, D.T. (1976). A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *J Comput Physics*, **22**, 403–434.
10. Gillespie, D.T. (1977). Exact Stochastic Simulation of Coupled Chemical Reactions. *The Journal of Physical Chemistry*, **81**, 25, 2340–2361.
11. Goss, P.J.E., Peccoud, J. (1998) Quantitative modeling of stochastic systems in molecular biology using stochastic Petri nets. *Proc. Natl. Acad. Sci. USA*, **95**, 6750–6755.
12. Krasnogor, N., Gheorghe, M., Terrazas, G., Diggle, S., Williams, P., Camara, M. (2005) An appealling Computational Mechanism Drawn from Bacterial Quorum Sensing. *Bulletin of the EATCS*, **85**, 135–148.
13. Lecca, P., Corrado, P. Cell Cycle Control in Eukaryotes: A BioSpi Model, *Electronic Notes in Theoretical Computer Science*, to appear.
14. Novak, B., Csikasz-Nagy, A., Gyorffy, B., Nasmyth, K., Tyson, J.J. Model Scenarios for Evolution of the Eukaryotic Cell Cycle, (1998) *Phil. Trans. R. Soc. Lond.*, **353**, 2063–2076.
15. Păun, Gh. (2000). Computing with Membranes, *Journal of Computer and System Sciences*, **61**(1) 108 – 143.
16. Pérez-Jiménez, M.J., Romero-Campero, F.J. Modelling Gene Expression Control Using P Systems: The Lac Operon, A Case Study, submitted.

17. Pérez-Jiménez, M.J., Romero-Campero, F.J. (2006) P Systems, a New Computationl Modelling Tool for Systems Biology, *Transactions on Computational Systems Biology*, to appear.
18. Pérez-Jiménez, M.J., Romero-Campero, F.J. A Model of the Quorum Sensing System in Vibrio fischeri using P Systems, submitted.
19. Pescini, D., Besozzi, D., Mauri, G., Zandron, C. (2006) Dynamical probabilistic P systems, *International Journal of Foundations of Computer Science*, **17** (1) 183–195.
20. Regev, A., Shapiro, E. The π-calculus as an abstraction for biomolecular systems. In Gabriel Ciobanu and Grzegorz Rozenberg, editors, *Modelling in Molecular Biology*. Springer 2004.
21. Terrazas, G., Krasnogor, N., Gheorghe, M., Bernardini, F., Diggle, S., Camara, M. (2005) An Environment Aware P-System Model of Quorum Sensing. *CIE 2005, Lecture Notes in Computer Science*, **3526**, 473–485.
22. PRISM Web Site: `http://www.cs.bham.ac.uk/~dxp/prism/`
23. The P Systems Web Site: `http://psystems.disco.unimib.it`
24. SciLab Web Site `http://scilabsoft.inria.fr/`
25. `http://www.dcs.shef.ac.uk/~marian`