Formal Methods for Biochemical Signalling Pathways

Muffy Calder[1], Stephen Gilmore[2], Jane Hillston[2], and Vladislav Vyshemirsky[1]

[1] Department of Computing Science, University of Glasgow, Glasgow, Scotland `muffy,vvv@dcs.gla.ac.uk`.
[2] Laboratory for Foundations of Computer Science, University of Edinburgh, Edinburgh, Scotland `stg,jeh@inf.ed.ac.uk`.

1 Introduction

This chapter considers a different and novel application for quantitative formal methods, biochemical signalling pathways. The methods we use were developed for modelling *engineered* systems such as computer networks and communications protocols, but we have found them highly suitable for modelling and reasoning about *evolved* networks such as biochemical signalling pathways.

Biochemical signalling pathways are communication mechanisms for the control and coordination of cells in living organisms. Cells "sense" a stimulus and then communicate an appropriate signal to the nucleus, which makes a response. The response depends upon the way in which the signals are communicated in the pathway. Signalling pathways are complicated communication mechanisms, with feedback, and embedded in larger networks. Understanding how these pathways function is crucial, since their malfunction results in a large number of diseases such cancer, diabetes, and cardiovascular disease. Good *predictive* models can guide experimentation and drug development for pathway interventions.

Historically, pathway models either encode static aspects, such as which components in a pathway (proteins) have the potential to interact, or provide simulations of system dynamics using either ordinary differential equations (ODEs) [dJ02, Voi00] or stochastic simulations of individuals using Gillespie's algorithm [Gil77]. Here, we introduce a novel approach to analytic pathway modelling. The key idea is that *pathways have stochastic, computational content*. We consider pathways as *distributed systems*, viewing the components as *processes* which can interact with each other, via biochemical reactions. The reactions have *duration*, defined by (performance) *rates*; therefore we model using high level formal languages whose underlying semantics is continuous time Markov chains (CTMCs). A distinctive aspect of our work is that we do not model individual molecules, but *species* of molecules, i.e. we model molecular concentrations.

Biological modelling is complex and error-prone. We believe that high-level stochastic modelling languages can complement the efficient numerical methods currently in widespread use by computational biologists. Process algebras have a comprehensive theory for reasoning and verification. They are also supported by state-of-the-art tools for analysis which realise the theory mechanically and support ambitious modelling studies which include the essential representational detail demanded for physically accurate work.

We have developed models using two different high level formal languages: PEPA [Hil96] and PRISM [KNP02]. These languages allow us to concentrate on modelling behaviour at a high level of abstraction, focusing on compositionality, communication and interaction, rather than working at the low level detail of a CTMC or system of ODEs. Both languages have extensive toolsets and both are suited to modelling and analysis of biochemical pathways, but in different ways. The former is a process algebra, and so the models are easily and clearly expressed, using the built-in operators. Markovian analysis is supported by the toolset. The PRISM language represents systems using an imperative language of reactive modules. It has the capability to express a wide range of stochastic process models including both discrete- and continuous-time Markov chains and Markov decision processes. A key feature of both languages is multiway synchronisation, essential for our approach.

In the next section, we give a brief overview of background material, presenting only the essential details of stochastic process theory needed to appreciate what follows. In Section 3 we give an introduction to our modelling approach. In Section 4 we present the syntax and semantics of the stochastic process algebra which we use, PEPA, and discuss how individual reactions and reaction pathways are modelled. In Section 5 we present an example, the ERK signalling pathway. Biochemical pathways are commonly modelled using ODE models; we compare with these in Section 6. We relate the above to a method based on model checking properties in temporal logic in Section 7. Section 8 contains a discussion. Further and related work is presented in Section 9 and we conclude in Section 10.

Parts of the present work were previously presented in the papers [CGH05, CGH06, CVOG06].

2 Preliminaries

In this section we present background material on the stochastic processes which we use in our work, particularly Markov processes. The literature on Markov processes is vast but as introductory texts we would recommend [KS60] and [Nor97]. Introductions to the application of stochastic processes in the physical sciences include [Gil91] and [Gar04]. In this brief introduction we attempt to explain some of the fundamental assumptions on which Markovian analysis is based, such as “memorylessness”. By doing this we hope to guide the reader towards an understanding of the applicability of our results and their scope.

2.1 Continuous time Markov chains

A fundamental kind of quantitative model is a Continuous-Time Markov Chain (CTMC), a finite-state stochastic process which evolves in continuous time. CTMCs are widely used in quantitative modelling because the numerical procedures for working with them are widely-known [Ste94].

Such a system can be easily thought of as a labelled transition system (LTS) where the labels describing the transitions from one state to another record information such as the *rate* at which this transition can occur. Logically, one might think of such an LTS as a graph where the presence of an arc from vertex i to vertex j with label r indicates the possibility of moving from state i to state j with rate r and the absence of an arc indicates that it is impossible for the system to move directly from state i to state j. Algorithmically, it is more productive for numerical purposes to represent this information instead as a matrix where an entry r in position ij records the rate information and a zero at position ij indicates the impossibility of moving directly from state i to state j. In practice, for typical models these matrices are usually sparse with the zero entries greatly outnumbering the non-zeros. If the model of the system is expressed in this way then the calculation of performance measures such as availability and utilisation can be obtained by using procedures of numerical linear algebra to compute the steady-state probability distribution over the states of this finite-state system and calculating the measure of interest from this. More complex measures such response time require more sophisticated analysis procedures.

CTMCs associate an exponentially distributed random variable with each transition from state to state. The random variable expresses quantitative information about the rate at which the transition can be performed. Formally, a random variable is said to have an exponential distribution with parameter λ (where $\lambda > 0$) if it has the probability distribution function

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases}$$

The mean, or *expected value*, of this exponential distribution is $1/\lambda$. The time interval between successive events is $e^{-\lambda t}$.

The *memoryless property* of the exponential distribution is so called because the time to the next event is independent of when the last event occurred. The exponential distribution is the only distribution function which has this property.

A Markov process with discrete state space (x_i) and discrete index set is called a *Markov chain*. The future behaviour of a Markov chain depends only on its current state, and not on how that state was reached. This is the *Markov, or memoryless, property.* Every finite-state Markov process can be described by its *infinitesimal generator matrix*, Q. Q_{ij} is the total transition rate from state i to state j.

Stochastic models admit many different types of analysis. Some have lower evaluation cost, but are less informative, such as *steady-state analysis.* Steady-state analysis tells us only about the *stationary*, or *equilibrium*, probability distribution over all of the states of the system. This contains no information about probabilities of states near to the start of the time evolution of the system. Other types of analysis have higher evaluation cost, but are more informative, such as *transient analysis.* This tells us about the probability distribution over all of the states of the system at all time points, and measures such as *first passage times* can be computed from this. Passage times are needed for the calculation of response times between an input stimulus and its associated output response.

We will be dealing with *time-homogeneous* Markov processes, where time does not influence probability. These processes will also be *irreducible*, meaning that it is possible to get to any state from any other. Finally, the states of these processes will be *positive-recurrent*, meaning that every state can be visited infinitely often. A stationary probability distribution, $\pi(\cdot)$, exists for every time-homogeneous irreducible Markov process whose states are all positive-recurrent [KS60]. At equilibrium the probability flow into every state is exactly balanced by the probability flow out so the equilibrium probability distribution can be found by solving the *global balance equation*

$$\pi Q = 0$$

subject to the normalisation condition

$$\sum_i \pi(x_i) = 1.$$

From this probability distribution can be calculated performance measures of the system such as throughput and utilisation.

An alternative is to find the *transient* state probability row vector $\pi(t) = [\pi_0(t), \ldots, \pi_{n-1}(t)]$ where $\pi_i(t)$ denotes the probability that the CTMC is in state i at time t.

2.2 Continuous stochastic logic

CSL [BaHK00, ASSB00] is a continuous time logic that allows one to express a probability measure that a temporal property is satisfied, in either transient behaviours or in steady state behaviours. We assume a basic familiarity with the logic, which is based upon the computational tree logic CTL [CE81]. Properties are expressed by state or path formulae. The operators include the usual propositional connectives, plus the binary temporal operator *until* operator $\mathbf{U}$. The *until* operator may be time bounded or unbounded. Probabilities may also be bounded. $\star p$ specifies a bound, for example $P_{\star p}[\phi]$ is true in a state s if the probability that(state property) ϕ is satisfied by the paths from state s meets the bound $\star p$. Examples of bounds are > 0.99 and < 0.01. A special case of $\star p$ is no bound, in which case we calculate a probability.

Properties are *transient*, that is, they depend on time; or they are *steady state*, that is, they hold in the long run. Note that in this context, steady state solutions are not (generally) single states, but rather a network of states (with cycles) which define the probability distributions in the long run. Table 1 gives an overview of CSL, ϕ is a state formula.

We use the PRISM model checker [KNP02] to check the validity of CSL properties. In PRISM, we write $P_{=?}[\phi]$, to return the probability of the transient property ϕ, and $S_{=?}[\phi]$, to return the probability of the steady state property ϕ. The default is checking from the initial state, but we can apply a filter thus: $P_{=?}[\psi\{\phi\}]$, which returns the probability, from the (first) state satisfying ϕ, of satisfying ψ.

Operator	**CSL Syntax**
True	$true$
False	$false$
Conjunction	$\phi \wedge \phi$
Disjunction	$\phi \vee \phi$
Negation	$\neg\phi$
Implication	$\phi \Rightarrow \phi$
Next	$P_{\star p}[\mathbf{X}\phi]$
Unbounded Until	$P_{\star p}[\phi\mathbf{U}\phi]$
Bounded Until	$P_{\star p}[\phi\mathbf{U}^{\leq t}\phi]$
Bounded Until	$P_{\star p}[\phi\mathbf{U}^{\geq t}\phi]$
Bounded Until	$P_{\star p}[\phi\mathbf{U}^{[t_1,t_2]}\phi]$
Steady-State	$S_{\star p}[\phi]$

Table 1. Continuous Stochastic Logic operators

3 Modelling biochemical pathways

Biochemical pathways consist of proteins, which interact with each other through chemical reactions. The *pathway* is a sequence of reactions, which may have feedback and other dependencies between them. The *signal* is a high concentration, or abundance, of particular molecular species, and by the sequence of reactions the signal is carried down the pathway. In modelling terms, a reaction is an activity between reactants to produce products; so, reactants play the role of producer(s), and products the role of consumer(s).

From the description above we can see that we can view a pathway as a distributed system: all stages of the pathway may be activated at once. We associate a concurrent, computational process with each of the proteins in the pathway. In other words, in our approach *proteins are processes* and in the underlying CTMC, *reactions are transitions.* Processes (i.e. proteins) interact, or communicate with each other *synchronously*, by participating in reactions which build up and break down proteins. The availability of reactants plays a crucial role in the progress of the pathway. This is usually expressed in terms of the rate of reaction being proportional to the concentration of the reactants. In basic terms, a producer can participate in a reaction when there is enough species for a reaction, a consumer can participate when it is ready to be replenished. A reaction occurs only when all the producers and consumers are ready to participate.

It is important to note that we view the protein *species* as a process, rather than each *molecule* as a process. Thus the local states of each process reflect different levels of concentration for that species. This corresponds to a *population* type model (rather than an *individuals* type model) and more readily allows us to reflect the dynamics of the reactions. In traditional population models, species are represented as real valued molar concentrations. In our approach, the concentrations are discretised, each level representing an interval of concentration values. The granularity of the discretisation can vary; the coarsest possible being two values (representing, for example, enough and not enough, or *high* and *low*). Time is the only continuous variable, all others are discrete.

4 Modelling pathways in PEPA

We assume some familiarity with process algebra; a brief overview of the stochastic process algebra PEPA is below, see [Hil96] for further details.

4.1 Syntax of the language

As in all process algebras, PEPA describes the behaviour of a system as a set of processes or components, which undertake activities. All activities in PEPA are timed. In order to capture variability and uncertainty, the duration of each action is assumed to be a random variable which is exponentially distributed. For example, using the prefix combinator, the component $(\alpha, r).S$ carries out activity (α, r), which has action type α and an exponentially distributed duration with parameter r (average delay $1/r$), and it subsequently behaves as S. The component $P + Q$ represents a system which may behave either as P or as Q. The activities of both P and Q are enabled. The first activity to complete distinguishes one of them: the other is discarded. The system will behave as the derivative resulting from the evolution of the chosen component. The expected duration of the first activities of P and Q will influence the outcome of the choice (the *race policy*), so the outcome is probabilistic.

It is convenient to be able to assign names to patterns of behaviour associated with components. Constants are components whose meaning is given by a defining equation. The notation for this is $X \overset{def}{=} E$. The name X is in scope in the expression on the right hand side meaning that, for example, $X \overset{def}{=} (\alpha, r).X$ performs α at rate r forever. PEPA supports multiway cooperations between components: the result of synchronising on an activity α is thus another α, available for further synchronisation. We write $P \underset{L}{\bowtie} Q$ to denote cooperation between P and Q over L. The set which is used as the subscript to the cooperation symbol, the *cooperation set* L, determines those activities on which the *cooperands* are forced to synchronise. For action types not in L, the components proceed independently and concurrently with their enabled activities. We write $P \parallel Q$ as an abbreviation for $P \underset{L}{\bowtie} Q$ when L is empty. PEPA was developed originally for performance modelling of computer and communication systems. In this context it is assumed that sychronised activities respect the notion of *bounded capacity*: a component cannot perform an activity at a rate greater than its own specification for the activity. Therefore in PEPA the rate for the synchronised activities is the *minimum* of the rates of the synchronising activities. For example, if process A performs α with rate λ_1, and process B performs α with rate λ_2, then the rate of the shared activity when A cooperates with B on α is $min(\lambda_1, \lambda_2)$. We use the distinguished symbol $\top$, to indicate that a component is *passive* with respect to an activity, i.e. for all rates k, $min(k, \top) = k$.

4.2 Semantics of the language

PEPA has a structured operational semantics which generates a labelled (multi-)transition system for any PEPA expression. It is a multi-transition system because the multiplicity of activities is important and may change the dynamic behaviour of the model. Via the structured operational semantics, PEPA models give rise to CTMCs. The relationship between the process algebra model and the CTMC representation is the following. The process terms (P_i) reachable from the initial state of the PEPA model by applying the operational semantics of the language form the states of the CTMC. For every set of labelled transitions between states P_i and P_j of the model $\{(\alpha_1, r_1), \ldots, (\alpha_n, r_n)\}$ add a transition with rate r where r is the sum of $r_1, \ldots, r_n$. The activity labels (α_i) are necessary at the process algebra in order to enforce synchronisation points, but are no longer needed at the Markov chain level.

Algebraic properties of the underlying stochastic process become algebraic laws of the process algebra. We obtain an analogue of the *expansion law* of untimed process algebra [Mil89]:

$$(\alpha, r).P \parallel (\beta, s).Q = (\alpha, r).(P \parallel (\beta, s).Q) + (\beta, s).((\alpha, r).P \parallel Q)$$

only if the exponential distribution is used. Due to memorylessness we do not need to adjust the rate s to take account of the time which elapsed during this occurrence of α (and analogously for r and β).

The strong equivalence relation over PEPA models is a congruence relation as is usual in process algebras and is a bisimulation in the style of Larsen and Skou [LS91]. It coincides with the Markov process notion of *lumpability* (a lumpable partition is the only partition of a Markov process which preserves the Markov property [KS60]). This correspondence makes a strong and unbreakable bond between the concise and elegant world of process algebras and the rich and beautiful theory of stochastic processes.

The fact that the strong equivalence relation is a semantics-preserving congruence has practical applications also. The relation can be used to aggregate the state space of a PEPA model, accelerating the production of numerical results and allowing larger modelling studies to be undertaken [GHR01].

4.3 Reactions

As an example of how reactions are modelled, consider a simple single, reversible reaction, as illustrated in Fig. 1. This describes a reversible reaction between three proteins: $Prot1$, $Prot2$ and $Prot3$, with forward rate $k1$, and reverse rate $k2$.

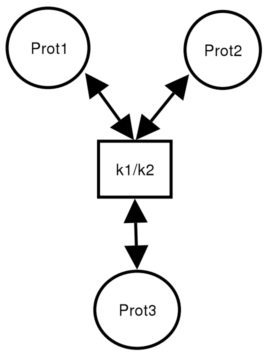

Fig. 1. Simple biochemical reaction

In the forward reaction (from top to bottom), $Prot1$, and $Prot2$ are the producers, $Prot3$ is the consumer; in the backward reaction, the converse is true. Using this example, we illustrate how proteins and reactions are represented in PEPA.

Consider the coarsest discretisation. We refer to the two values as *high* and *low* and subscript protein processes by H and L respectively. Thus when there are n proteins there are $2n$ equations. Assuming the forward reaction is called $r1$, and the reverse reaction $r2$, the equations are given in Fig. 2. We set the rate of consumers to be the passive rate $\top$.

$$\begin{array}{ll}
Prot1_H \stackrel{def}{=} (r1, k1).Prot1_L & Prot1_L \stackrel{def}{=} (r2, \top).Prot1_H \\
Prot2_H \stackrel{def}{=} (r1, k1).Prot2_L & Prot2_L \stackrel{def}{=} (r2, \top).Prot2_H \\
Prot3_H \stackrel{def}{=} (r2, k2).Prot3_L & Prot3_L \stackrel{def}{=} (r1, \top).Prot3_H
\end{array}$$

Fig. 2. Simple biochemical reaction in PEPA: model equations

The model configuration, given in Figure 3, defines the (multi-way) synchronisation of the three processes. Note that initially, the producers are *high* and the consumer is *low*.

$$Prot1_H \underset{\{r1,r2\}}{\bowtie} Prot2_H \underset{\{r1,r2\}}{\bowtie} Prot3_L$$

Fig. 3. Simple biochemical reaction in PEPA: model configuration

The model configuration defines a CTMC. Fig. 4 gives a graphical representation of the underlying CTMC, with the labels of the states indicating protein values.

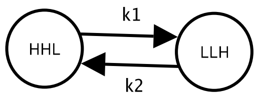

Fig. 4. CTMC for PEPA model of a simple biochemical reaction

4.4 Pathways and discretisation

A pathway involves many reactions, relating to each other in (typically) nonlinear ways. In PEPA, pathways are expressed by defining alternate choices for protein behaviours using the + operator. Consider extending the simple example. Currently, $Prot3$ is a consumer in $r1$. If it was also the consumer in another reaction, say $r3$, then this would be expressed by the equation:

$$Prot3_L \stackrel{def}{=} (r1, \top).Prot3_H + (r3, \top).Prot3_H$$

It is possible to model with finer grained discretisations of species, for example processes can be indexed by any countable set thus:

$$\begin{aligned}
Prot1_N &\overset{def}{=} (r1, N * k1).Prot1_{N-1} \\
Prot1_{N-1} &\overset{def}{=} (r1, (N-1) * k1).Prot1_{N-2} + (r2, \top).Prot1_N \\
&\ldots \\
Prot1_1 &\overset{def}{=} (r1, k1).Prot1_0 \qquad\qquad\quad + (r2, \top).Prot1_2 \\
Prot1_0 &\overset{def}{=} \qquad\qquad\qquad\qquad\qquad\quad (r2, \top).Prot1_1
\end{aligned}$$

Note that the rates are adjusted to reflect the relative concentrations in different states/levels of the discretisation. N need not be fixed across the model, but can vary across proteins, depending on experimental evidence and measurement techniques.

In a model with two levels of discretisation we specify non-passive rates (the known rate constant) for each occurrence of a reaction event in a producer process; since PEPA defines the rate of a synchronisation to be the rate of the slowest synchronising component, the rate for a given reaction will be exactly that rate constant. In the simple example, this means that initially, the three occurrences of $r1$ will synchronise, with rate $k1 = min(k1, k1, \top)$.

Finally, we note that for any pathway, in the model configuration, the synchronisation sets must include the pairwise shared activities of all processes. In the example configuration shown in Fig. 3, the two synchronisation sets are identical. This is rarely the case in a pathway, where each protein is typically involved in a different set of reactions.

5 An example: ERK signalling pathway

Cells are the fundamental structural and functional units of living organisms. We are interested in eukaryotic cells, which have an outer membrane and inner compartments including a nucleus. Cell signalling is the mechanism whereby cell behaviour is controlled and coordinated: signalling molecules are detected at the outer membrane of a cell, signals are conveyed to the nucleus via a pathway, and the cell makes a response.

Our example is one of the most important and ubiquitous cell signalling pathways: the ERK pathway [EE02] (also called Ras/Raf, or Raf-1/MEK/ERK pathway). This pathway is of great interest because abnormal functioning leads to uncontrolled growth and cancer. The overall, basic, behaviour of the pathway is signals are conveyed to the nucleus through a "cascade" of proteins (protein kinases) called Raf, MEK and ERK. The simple cascade is illustrated in Fig. 5.

When the first protein Raf is activated by the incoming signal, the key subsequent activity is the conveyance of the signal along the pathway; in this case the signal is conveyed by *phosphorylation*. Phosphorylation is the addition of a phosphate group to a protein, the source of the phosphoryl groups is ATP (adenosine triphosphate), which becomes ADP (adenosine diphosphate); fortunately ATP is in such abundance that we do not need to consider it explicitly. The Raf, MEK and ERK proteins are called kinases because they

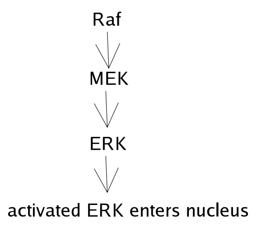

Fig. 5. Simpified ERK signalling pathway

transfer phosphoryl groups from ATP to each other. Proteins called phosphotases have the inverse function. Fig. 6 gives a high level overview of the process of phosphorylation.

Phosphorylation brings about a change in protein activity; in the context of a pathway, phosphorylation/dephosphorylation represents on/off behaviour, or the presence/absence of a signal. In the ERK pathway, the protein kinases become progressively phosphorylated, or activated, as the signal is conveyed to the nucleus.

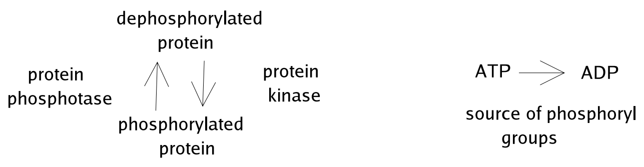

Fig. 6. Signalling

A phosphorylated protein is indicated by a (single or double) suffix, for example, ERK-P is singly phosphorylated ERK whereas ERK-PP is doubly phosphorylated ERK. Activated Raf is indicated by a “*” suffix.

The full behaviour of the ERK pathway is complex, here we focus on a portion of pathway behaviour: how an additional protein, called RKIP (Raf kinase inhibitor protein) inhibits, or regulates, the signalling behaviour of the pathway. Namely, RKIP interferes with the cascade by reacting with activated Raf.

The effect of RKIP is the subject of ongoing research, we have based on our model on the recent experimental results of Cho et al [CSK$^+$03].

A graphical representation of the pathway, taken from [CSK$^+$03] (with a small modification, see section 5.2) is given in Fig. 7. Each node is labelled by a protein *species*, that is, a molar concentration. The particular form of Raf in this investigation is called Raf-1; MEK, ERK and RKIP are as described above, and RP is an additional protein phosphotase. There are numerous

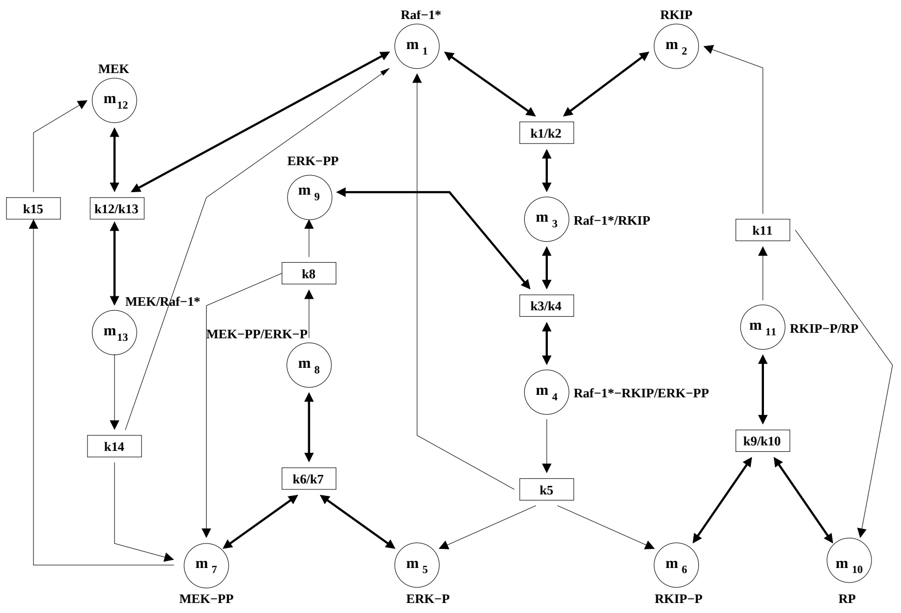

Fig. 7. RKIP inhibited ERK pathway

additional complex proteins built up from simpler ones; the complexes are named after their constituents. For example, Raf-1*/RKIP is a protein built up from Raf-1* and RKIP.

In Fig. 7 species concentrations for Raf-1* etc. are given by the variables $m1$ etc. Initially, all concentrations are unobservable, except for m_1, m_2, m_7, m_9, and m_{10} [CSK$^+$03]. Note that in this pathway, not all reactions are reversible, the non-reversible reactions are indicated by uni-directional arrows.

5.1 PEPA model

Fig. 8 gives the PEPA equations for the coarsest discretisation of the pathway, with the model configuration in Fig. 9.

There are two equations for each protein, for high and low behaviour. The pathway is highly non-linear, as indicated by numerous occurrences of the choice operator. For example, when Raf-1* is high, it can react with RKIP in the $k1react$ reaction, *or* it can react with MEK in the $k12react$ reaction. Some choices simply reflect the reversible nature of some reactions; for example RKIP is replenished by the $k2react$ reaction, which is the reverse of $k1react$.

Unlike our simple reaction example, in the pathway model, the synchronisation sets in the model configuration are all distinct. Although we derived the synchronisation sets by hand, it would be possible to do so algorithmically, by inspection of the model equations.

$$\begin{aligned}
\text{Raf-1}^*_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k1react}, k_1).\text{Raf-1}^*_{\mathrm{L}} + (\mathit{k12react}, k_{12}).\text{Raf-1}^*_{\mathrm{L}} \\
\text{Raf-1}^*_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k5product}, \top).\text{Raf-1}^*_{\mathrm{H}} + (\mathit{k2react}, \top).\text{Raf-1}^*_{\mathrm{H}} \\
&\quad + (\mathit{k13react}, \top).\text{Raf-1}^*_{\mathrm{H}} + (\mathit{k14product}, \top).\text{Raf-1}^*_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{RKIP}_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k1react}, k_1).\text{RKIP}_{\mathrm{L}} \\
\text{RKIP}_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k11product}, \top).\text{RKIP}_{\mathrm{H}} + (\mathit{k2react}, \top).\text{RKIP}_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{MEK}_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k12react}, k_{12}).\text{MEK}_{\mathrm{L}} \\
\text{MEK}_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k13react}, \top).\text{MEK}_{\mathrm{H}} + (\mathit{k15product}, \top).\text{MEK}_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{MEK/Raf-1}^*_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k14product}, k_{14}).\text{MEK/Raf-1}^*_{\mathrm{L}} + (\mathit{k13react}, k_{13}).\text{MEK/Raf-1}^*_{\mathrm{L}} \\
\text{MEK/Raf-1}^*_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k12react}, \top).\text{MEK/Raf-1}^*_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{MEK-PP}_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k6react}, k_6).\text{MEK-PP}_{\mathrm{L}} + (\mathit{k15product}, k_{15}).\text{MEK-PP}_{\mathrm{L}} \\
\text{MEK-PP}_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k8product}, \top).\text{MEK-PP}_{\mathrm{H}} + (\mathit{k7react}, \top).\text{MEK-PP}_{\mathrm{H}} \\
&\quad + (\mathit{k14product}, \top).\text{MEK-PP}_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{ERK-PP}_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k3react}, k_3).\text{ERK-PP}_{\mathrm{L}} \\
\text{ERK-PP}_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k8product}, \top).\text{ERK-PP}_{\mathrm{H}} + (\mathit{k4react}, \top).\text{ERK-PP}_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{ERK-P}_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k6react}, k_6).\text{ERK-P}_{\mathrm{L}} \\
\text{ERK-P}_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k5product}, \top).\text{ERK-P}_{\mathrm{H}} + (\mathit{k7react}, \top).\text{ERK-P}_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{MEK-PP/ERK-P}_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k8product}, k_8).\text{MEK-PP/ERK-P}_{\mathrm{L}} + (\mathit{k7react}, k_7).\text{MEK-PP/ERK-P}_{\mathrm{L}} \\
\text{MEK-PP/ERK-P}_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k6react}, \top).\text{MEK-PP/ERK-P}_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{Raf-1}^*\text{/RKIP}_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k3react}, k_3).\text{Raf-1}^*\text{/RKIP}_{\mathrm{L}} + (\mathit{k2react}, k_2).\text{Raf-1}^*\text{/RKIP}_{\mathrm{L}} \\
\text{Raf-1}^*\text{/RKIP}_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k1react}, \top).\text{Raf-1}^*\text{/RKIP}_{\mathrm{H}} + (\mathit{k4react}, \top).\text{Raf-1}^*\text{/RKIP}_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{Raf-1}^*\text{/RKIP/ERK-PP}_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k5product}, k_5).\text{Raf-1}^*\text{/RKIP/ERK-PP}_{\mathrm{L}} \\
&\quad + (\mathit{k4react}, k_4).\text{Raf-1}^*\text{/RKIP/ERK-PP}_{\mathrm{L}} \\
\text{Raf-1}^*\text{/RKIP/ERK-PP}_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k3react}, \top).\text{Raf-1}^*\text{/RKIP/ERK-PP}_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{RKIP-P}_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k9react}, k_9).\text{RKIP-P}_{\mathrm{L}} \\
\text{RKIP-P}_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k5product}, \top).\text{RKIP-P}_{\mathrm{H}} + (\mathit{k10react}, \top).\text{RKIP-P}_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{RP}_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k9react}, k_9).\text{RP}_{\mathrm{L}} \\
\text{RP}_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k11product}, \top).\text{RP}_{\mathrm{H}} + (\mathit{k10react}, \top).\text{RP}_{\mathrm{H}}
\end{aligned}$$

$$\begin{aligned}
\text{RKIP-P/RP}_{\mathrm{H}} &\overset{\mathit{def}}{=} (\mathit{k11product}, k_{11}).\text{RKIP-P/RP}_{\mathrm{L}} + (\mathit{k10react}, k_{10}).\text{RKIP-P/RP}_{\mathrm{L}} \\
\text{RKIP-P/RP}_{\mathrm{L}} &\overset{\mathit{def}}{=} (\mathit{k9react}, \top).\text{RKIP-P/RP}_{\mathrm{H}}
\end{aligned}$$

Fig. 8. PEPA model definitions for the reagent-centric model

$$\begin{array}{l}
(\text{Raf-1}^*_{\text{H}} \underset{\{k1react,k2react,k12react,k13react,k5product,k14product\}}{\bowtie} \\
\quad (\text{RKIP}_{\text{H}} \underset{\{k1react,k2react,k11product\}}{\bowtie} \\
\quad\quad (\text{Raf-1}^*/\text{RKIP}_{\text{L}} \underset{\{k3react,k4react\}}{\bowtie} \\
\quad\quad\quad (\text{Raf-1}^*/\text{RKIP/ERK-PP}_{\text{L}}) \underset{\{k3react,k4react,k5product\}}{\bowtie} \\
\quad\quad\quad\quad (\text{ERK-P}_{\text{L}} \underset{\{k5product,k6react,k7react\}}{\bowtie} \\
\quad\quad\quad\quad\quad (\text{RKIP-P}_{\text{L}} \underset{\{k9react,k10react\}}{\bowtie} \\
\quad\quad\quad\quad\quad\quad (\text{RKIP-P/RP}_{\text{L}} \underset{\{k9react,k10react,k11product\}}{\bowtie} \\
\quad\quad\quad\quad\quad\quad\quad (\text{RP}_{\text{H}} \parallel \\
\quad\quad\quad\quad\quad\quad\quad\quad (\text{MEK}_{\text{L}} \underset{\{k12react,k13react,k15product\}}{\bowtie} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad (\text{MEK/Raf-1}^*_{\text{L}} \underset{\{k14product\}}{\bowtie} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (\text{MEK-PP}_{\text{H}} \underset{\{k8product,k6react,k7react\}}{\bowtie} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (\text{MEK-PP/ERK-P}_{\text{L}} \underset{\{k8product\}}{\bowtie} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (\text{ERK-PP}_{\text{H}})))))))))))))
\end{array}$$

Fig. 9. PEPA model configuration for the reagent-centric model

5.2 Analysis

There are two principal reasons to apply formal languages to describe systems and processes. The first is the avoidance of ambiguity in the description of the problem under study. The second, but not less important, is that formal languages are amenable to automated processing by software tools. We used the PEPA Workbench [GH94] to analyse the model.

First, we used the Workbench to test for deadlocks in the model. A deadlocked system is one which cannot perform any activities (in many process algebras this is denoted by a constant such as *exit* or *stop*). In our context, signalling pathways should not be able to deadlock, this would indicate a malfunction. Initially, there were several deadlocks in our PEPA model: this is how we discovered an incompleteness in the published description of [CSK$^+$03], with respect to the treatment of MEK. After correspondence with the authors, we were able to correct the omission and develop a more complete, and deadlock free model.

Second, when we had a deadlock-free model, we used the Workbench to generate the CTMC (28 states) and its long-run probability distribution. The steady-state probability distribution is obtained using a number of routines from numerical linear algebra. The distribution varies as the rates associated with the activities of the PEPA model are varied, so the solution of the model is relative to a particular assignment of the rates. Initially, we set all rates to unity (1.0). Our main aim was to investigate how RKIP affects the production of phosphorylated ERK and MEK, i.e. if it reduces the probability of having a high level of phosphorylated ERK or MEK.

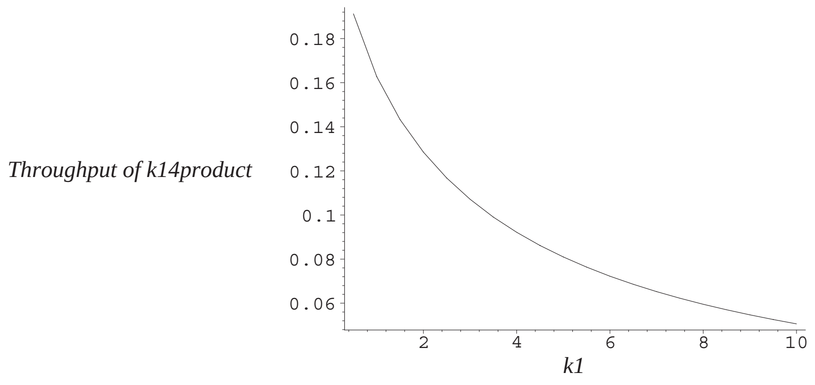

Fig. 10. Plotting the effect of $k1$ on $k14$*product*

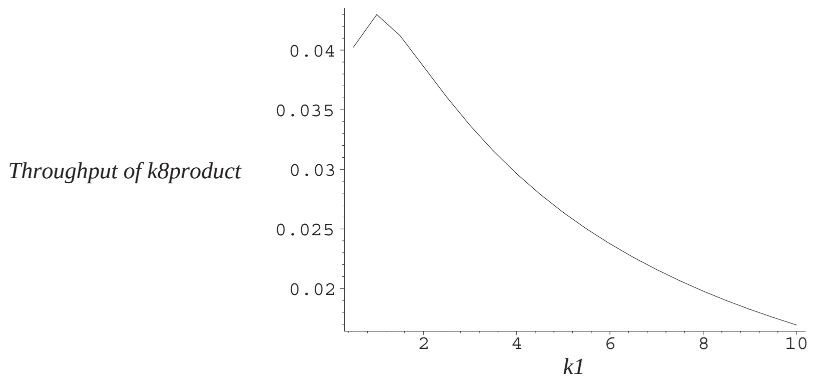

Fig. 11. Plotting the effect of $k1$ on $k8$*product*

We used the PEPA state-finder to aggregate the probabilities of all states when ERK-PP is high, or low, for a given set of rates. That is, it aggregated the probabilities of states whose (symbolic) description has the form $* \bowtie$ ERK-PP$_{\mathrm{H}}$ where $*$ is a wildcard standing for any expression. We then repeated this with a different set of rates and compared results. We observed that the probability of being in a state with ERK-PP$_{\mathrm{H}}$ *decreases* as the rate $k1$ is increased, and the converse for ERK-PP$_{\mathrm{L}}$ *increases*. For example, when $k1 = 1$, the probability of ERK-PP$_{\mathrm{H}}$ is .257, when $k1 = 100$, it drops to .005. We can also plot throughput (rate $\times$ probability) against rate. Fig. 10 and Fig. 11 show two sub-plots which detail the effect of increasing the rate $k1$ on the $k14$*product* and $k8$*product* reactions – the production of (doubly) phosphorylated MEK and (doubly) phosphorylated ERK, respectively. These are obtained by scaling $k1$, keeping all other rates to be unity. The graphs show that increasing the rate of the binding of RKIP to Raf-1* dampens down the $k14$*product* and $k8$*product* reactions, and they quantify this information. The efficiency of the reduction is greater in the former case: the graph falls away more steeply. In the latter case the reduction is more gradual and the throughput of $k8$*product* peaks at $k1 = 1$. Note that since $k5$*product* is on the same (sub)pathway as $k8$*product*, both ERK-PP and ERK-P are similarly affected. Thus we conclude that the rate at which RKIP binds to Raf-1* (thus suppressing phosphorylation of MEK) affects the ERK pathway, as predicted (and observed); RKIP does indeed regulate the ERK pathway.

In the next section we give an overview of traditional pathway models, and how they relate to our approach.

6 Modelling pathways with differential equations

The algebraic formulation of the PEPA model makes clear the interactions between the pathway components. There is a direct correspondence between topology and the model, models are easy to derive and to alter. This is not apparent in the traditional pathway models given by sets of ODEs. In these models, equations define how the concentration of each species varies over time, according to mass action kinetics. There is one equation for each protein species. The overall rate of a reaction depends on both a rate (constant) and the concentration masses. Both time and concentration variables are continuous. ODEs do not give an indication of the structure, or topology of the pathway, and consequently the process to define them is often error prone. Set against this, efficient numerical methods are available for the numerical integration of ODEs even in the difficult quantitative setting of chemically reacting systems which are almost always stiff due to the presence of widely differing timescales in the reaction rates.

Fortunately, the ODEs can be derived from PEPA models — in fact, from models which distinguish only the coarsest discretisation of concentration. The high/low discretisation is sufficient because we need to know only when a reaction *increases* or *decreases* the concentration of a species. Moreover the PEPA expressions reveal which species are required in order to complete an activity.

An example illustrates the relationship between ODEs and the PEPA model. In the PEPA equations for the ERK pathway, we can easily observe that Raf-1* increases (low to high, second equation) with rates (from synchronisations) k_5, k_2, k_{13} and k_{14}; it decreases (high to low, first equation) with rates k_1 and k_{12}. Mass action kinetics shows how the rates and masses affect the amount of Raf-1*. The equation for Raf-1*, given in terms of the (continuous) concentration variables m_1 etc. is

$$\frac{dm_1}{dt} = (k_5 \cdot m_4) + (k_2 \cdot m_3) + (k_{13} \cdot m_{13}) + (k_{14} \cdot m_{13}) - (k_1 \cdot m_1 \cdot m_2) - (k_{12} \cdot m_1 \cdot m_{12}) \tag{1}$$

This equation defines the change in Raf-1* by how it is *increased*, i.e. the positive terms, and how it is *decreased*, i.e. the negative terms. These differential equations can be derived directly and automatically from the PEPA model. Algorithms to do so are given in [CGH05].

While the style of modelling in the stochastic process algebra approach embodied by PEPA is concise and elegant, the bounded capacity, *min* style semantics for synchronisation means we have not been able to represent accurate rates for mass action kinetics. For example, in a PEPA model for a reaction between two producers and one consumer, the overall rate of a synchronised transition is the *minimum* of the three given rates. In mass action kinetics, the rate would be a (function of the) *product* of the given rates. We could overcome this by defining a very large number of constants, representing every possible rate $\times$ mass product combination; alternatively, we turn to the PRISM high level language for CTMCs, which implements synchronisation rates by products. This formalism, which is state based, is less concise for modelling, but it affords additional analysis by way of model checking properties expressed using continuous stochastic logic (CSL).

7 Modelling pathways in PRISM

The PRISM language [KNP02] represents systems using an imperative language of reactive modules. It has the capability to express a wide range of stochastic process models including both discrete- and continuous-time Markov chains and Markov decision processes. A key feature is multiway synchronisation, essential for our approach. In (CTMC) PRISM models, activities are called transitions. (Note, the PRISM name denotes both a modelling language and the model checker, the intended meaning should be clear from the context.) These correspond directly to CTMC transitions and they are labelled with performance rates and (optional) names. For each transition, like PEPA, the rate is defined as the parameter of an exponential distribution of the transition duration. PRISM is state-based; *modules* play the role of PEPA processes and define how state variables are changed by transitions. Like PEPA, transitions with common names are synchronised; transitions with distinct names are not synchronised.

7.1 Reactions

Similar to our PEPA models, proteins are represented by PRISM modules and reactions are represented by transitions. Below, we give a brief overview of the language, illustrating each concept with reference to the simple reaction example from Fig. 1; the reader is directed to [KNP02] for further details of PRISM.

The PRISM model for the simple example is given in Fig. 12. The first thing to remark is that in the PRISM model, the discretisation of concentration is an explicit parameter, denoted by N. In this example, we set it to 3. K is simply a convenient abbreviation for N^{-1}.

Second, consider the first three modules which represent the proteins $Prot1$, $Prot2$ and $Prot3$. Each module has the form: a state variable which denotes the protein concentration (we use the same name for process and variable, the type can be deduced from context) followed by a nondeterministic choice of transitions named $r1$ and $r2$. A transition has the form *precondition* $\rightarrow$ *rate: assignment*, meaning when the precondition is true, then perform the assignment at the given *rate, i.e.* rate is the parameter of an exponential distribution of the transition duration. The assignment defines the value of a state variable *after* the transition. The new value of a state variable follows the usual convention – the variable decorated with a single quote.

In this model, the transition rates have been chosen carefully, to correspond to mass action kinetics. Namely, when the transition denotes consumer behaviour (decrease protein by 1) the protein is multiplied by K, when the transition denotes producer behaviour (increase protein by 1), the rate is simply 1. These rates correspond to the fact that in mass action kinetics, the overall rate of the reaction depends on a rate constant and the concentrations of the reactants *consumed* in the reaction. (We will discuss this further in Section 7.2.) Note that unlike PEPA where the processes are recursive, here PRISM modules describe the circumstances under which transitions can occur.

The fourth module, *Constants*, simply defines the constants for reaction kinetics. These were obtained from experimental evidence [CSK+03]. the module contains a "dummy" state variable called x, and (always) enabled transitions which define the rates.

The four modules run concurrently, as given by the system description in Fig. 13. In PRISM, the rate for the synchronised transition is the *product* of the rates of the synchronising transitions. For example, if process A performs α with rate λ_1, and process B performs α with rate λ_2, then the rate of α when A is synchronised with B is $\lambda_1 \cdot \lambda_2$. To illustrate how this determines the rates in the underlying CTMC, consider the first reaction from the initial state of the example, i.e. reaction $r1$. There are four enabled (i.e. precondition is true) transitions with the name $r1$. They will all synchronise, and when they do, the resulting transition has rate

$$(Prot1 \cdot K) \cdot (Prot2 \cdot K) \cdot 1 \cdot (k1 \cdot N) \tag{2}$$

Since $Prot1$ and $Prot2$ are initialised to N ($Prot3$ is initialised to 0), this equates to

$$(N \cdot K) \cdot (N \cdot K) \cdot 1 \cdot (k1 \cdot N) = k1 \cdot N \tag{3}$$

```
const int N = 3;
const double K = 1/N;

module Prot1
    Prot1: [0..N] init N;
    [r1] (Prot1>0) -> Prot1*K: (Prot1' = Prot1 - 1);
    [r2] (Prot1<N) -> 1: (Prot1' = Prot1 + 1);
endmodule

module Prot2
    Prot2: [0..N] init N;
    [r1] (Prot2>0) -> Prot2*K: (Prot2' = Prot2 - 1);
    [r2] (Prot2<N) -> 1: (Prot2' = Prot2 + 1);
endmodule

module Prot3
    Prot3: [0..N] init 0;
    [r1] (Prot3 < N) -> 1: (Prot3' = Prot3 + 1);
    [r2] (Prot3>0) -> Prot2*K: (Prot3' = Prot3 - 1);
endmodule

module Constants
    x: bool init true;
    [r1] (x=true) -> k1*N: (x'=true);
    [r2] (x=true) -> k2*N: (x'=true);

endmodule
```

Fig. 12. Simple biochemical reaction in PRISM: modules

```
system
   Prot1 || Prot2 || Prot3 || Constants
endsystem
```

Fig. 13. Simple biochemical reaction in PRISM: system

The reaction $r1$ can occur N times, until all the $Prot1$ and $Prot2$ has been consumed. Fig. 14 gives a graphical representation of the underlying CTMC when $N = 3$. Again, the state labels indicate protein values, i.e. $x_1x_2x_3$ denotes the state where $Prot1 = x_1$, $Prot2 = x_2$, etc. Note that the transition rates decrease as the amount of producer decreases.

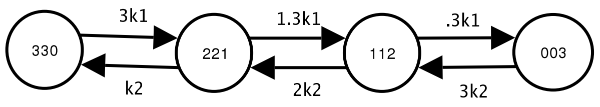

Fig. 14. CTMC for PRISM model of simple biochemical reaction

7.2 Reaction kinetics

In this section we show how the PRISM model implements mass action kinetics. Consider the mass action kinetics for $Prot3$ in the simple example, given by the ODE

$$\frac{dm_3}{dt} = (k_1 \cdot m_1 \cdot m_2) - (k_2 \cdot m_3) \tag{4}$$

The variables m_1 etc. are continuous, denoting concentrations of $Prot1$, etc. Integrating equation 4 by the simplest method, Euler's method, defines a new value for m_3 thus:

$$m_3' = m_3 + (k1 \cdot m_1 \cdot m_2 \cdot \Delta t) - (k_2 \cdot m_3 \cdot \Delta t) \tag{5}$$

In our discretisation, concentrations can only increase in units of one molar concentration, i.e. $1/N$, so

$$\Delta t = \frac{1}{N \cdot ((k1 \cdot m_1 \cdot m_2) - (k_2 \cdot m_3))} \tag{6}$$

Recall that PRISM implements rates as the memoryless negative exponential, that is for a given rate λ, $P(t) = 1 - e^{-\lambda t}$ is the probability that the action will be completed before time t. Taking λ as $\frac{1}{\Delta t}$, in this example we have

$$\lambda = (N \cdot k1 \cdot m_1 \cdot m_2) - (N \cdot k_2 \cdot m_3) \tag{7}$$

The continuous variables m_1 etc. relate to the PRISM variables $Prot1$ etc., as follows:

$$m_1 = Prot1 \cdot K \tag{8}$$

$$m_2 = Prot2 \cdot K \tag{9}$$

etc.

So, substituting into equation (7) yields

$$\lambda = (N \cdot k1 \cdot (Prot1 \cdot K) \cdot (Prot2 \cdot K)) - (N \cdot k_2 \cdot (Prot3 \cdot K)) \tag{10}$$

Given the initial concentrations of $Prot1$, $Prot2$ and $Prot3$, this equates to

$$\lambda = (N \cdot k1 \cdot (N \cdot K) \cdot (N \cdot K)) - (N \cdot k_2 \cdot (0 \cdot K)) \tag{11}$$

which simplifies to the rate specified in the PRISM model, equation (3) in Section 7.1.

Comparison of ODE and CTMC models

It is important to note that the ODE model is deterministic, whereas our CTMC models are stochastic. How do the two compare? We investigated simulation traces of both, over 200 data points in the time interval $[1 \ldots 100]$, using MATLAB for the the former. While PRISM is not designed for simulation, we were able to derive simulation traces, using the concept of *rewards* (see [KNP02]).

When comparing the two sets of traces, the accuracy of the CTMC traces depends on the choice of value for N. Intuitively, as N approaches infinity, the stochastic (CTMC) and deterministic (ODE) models will converge. For many pathways, including our example pathway, N can be surprisingly small (e.g. 7 or 8), to yield very good simulations, in reasonable time (few minutes) on a state of art workstation. The two are indistinguishable for practical purposes. More details about simulation results and comparison between the stochastic and deterministic models are given in [CVOG06].

One advantage of our approach is the modeller chooses the granularity of N. Usually this will depend on the accuracy of and confidence in experimental data or knowledge. In many cases, it is sufficient to use a high/low model, particularly when the data or knowledge is incomplete or very uncertain.

The full PRISM model for the example pathway is given in the Appendix. We now turn our attention to analysis of the example pathway using a temporal logic.

7.3 Analysis of example pathway using the PRISM model checker

Temporal logics are powerful tools for expressing properties which may be generic, such as state reachability, or application specific in which case they represent application characteristics. Here, we concentrate on the latter, specifically considering properties of biological significance.

The two properties we consider are: what is the probability that a protein concentration reaches a certain level, and then remains at that level thereafter, and what is the probability that one protein "peaks" before another? The former is referred to as *stability* (i.e. the protein is stable), the latter as *activation sequence.*

Since we have a stochastic model, we employ the logic CSL (Continuous Stochastic Logic) (see section 2.2) and the symbolic probabilistic model checker PRISM [PNK04] to compute steady state solutions and check validity. Using PRISM we can analyse open formulae, i.e. we can perform *experiments*

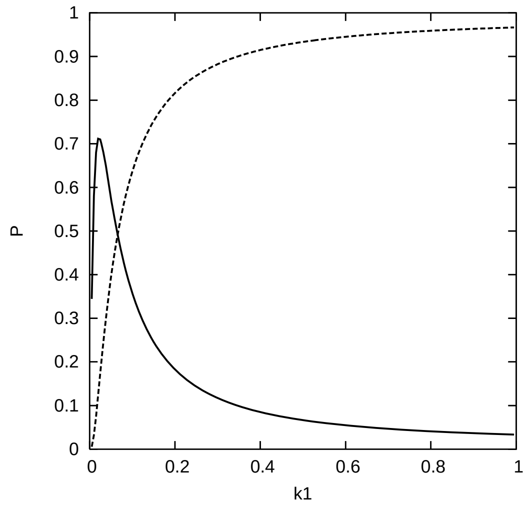

Fig. 15. Stability of Raf-1* at levels {2,3} and {0,1}

as we vary instances of variables in a formula expressing a property. Typically, we will vary reaction rates or concentration levels. We consider two properties below, the first is a steady state property and we vary a reaction rate, the second is a transient property and we vary a concentration. All properties were checked within a few minutes on a state of art workstation; hence run times are omitted.

Protein stability

Stability properties are useful during model fitting, i.e. fitting the model to experimental data. As an example, consider the stability of Raf-1* as the reaction rate $k1$ (the rate of $r1$ which binds Raf-1* and RKIP) varies over the interval $[0 \ldots 1]$. Let stability in this case be defined as concentration 2 or 3. The stability property is expressed by:

$$S_{=?}[(\text{Raf-1}^* \geq 2) \wedge (\text{Raf-1}^* \leq 3)] \tag{12}$$

Now consider the probability that Raf-1* is stable at concentrations 0 and 1; the formula for this is:

$$S_{=?}[(\text{Raf-1}^* \geq 0) \wedge (\text{Raf-1}^* \leq 1)] \tag{13}$$

Fig. 15 gives results for both these properties, when $N = 5$. From the graph, we can see that the likelihood of property (12) (solid line) is greatest when $k1 = 0.03$ and then it decreases; the likelihood of property (13) (dashed line) increases dramatically, becoming very likely when $k1 > 0.4$.

We note that the analysis presented in section 5.2 is for stability. For example, assuming $N = 1$, the probability that ERK-PP is high would be expressed in PRISM by $S_{=?}[\text{ERK-PP} \geq 1)]$.

Activation sequence

As an example of activation sequence, consider the two proteins Raf-1*/RKIP and Raf-1*/RKIP/ERK-PP, and their two peaks C and M, respectively. Is it possible that the (concentration of the) former peaks before the latter? This property is given by:

$$P_{=?}[(\text{Raf-1}^{*}/\text{RKIP/ERK-PP} < M)\ \mathbf{U}\ (\text{Raf-1}^{*}/\text{RKIP} = C)] \qquad (14)$$

The results, for C ranging over $0, 1, 2$ and M ranging over $1 \ldots 5$ are given in Fig. 16: the line with steepest slope represents $M = 1$, the line which is nearly horizontal is $M = 5$. For example, the probability Raf-1*/RKIP reaches concentration level 2 before Raf-1*/RKIP/ERK-PP reaches concentration level 5 is more than 99%, the probability Raf-1*/RKIP reaches concentration level 2 before RAF1/RKIP/ERK-PP reaches concentration level 2 is almost 96%.

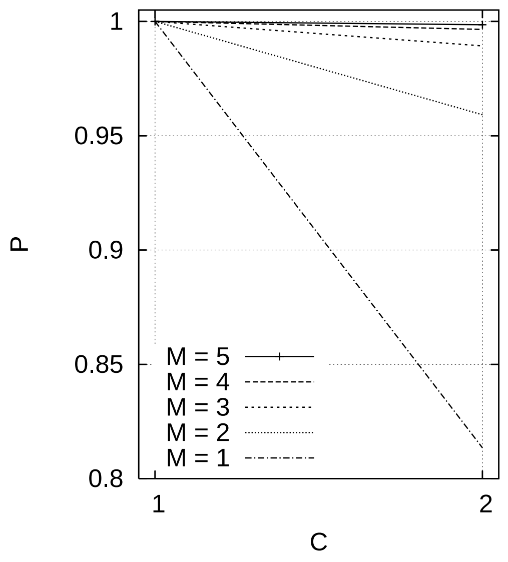

Fig. 16. Activation sequence

7.4 Further properties

Examples of further temporal properties concerning the accumulation (or diminution) of proteins, illustrate the use of bounds. Full details of their analysis can be found in [CVOG06].

The (accumulation) property

$$P_{=?}[(true)\ \mathbf{U}^{\leq 120}\ (Protein > C)\{(Protein = C)\}] \tag{15}$$

expresses the possibility that *Protein* can reach a level higher than C, within a time bound, once it has reached concentration C.

The (diminution) property

$$P_{\geq 1}[(true)\ \mathbf{U}\ ((Protein = C) \wedge (P_{\geq 0.95}[\mathbf{X}(Protein = C - 1)]))] \tag{16}$$

expresses the high likelihood of decreasing *Protein*, i.e. the concentration reaches C and after the next step it is very likely to be $C - 1$.

8 Discussion

Modelling biochemical signalling pathways has previously been carried out using sets of nonlinear ordinary differential equations (ODEs) or stochastic simulation based on Gillespie's algorithm. These can be seen as contrasting approaches in several respects. The ODE models are deterministic and present a *population* view of the system. This aims to characterise the average behaviour of large numbers of individual molecules of each species interacting, capturing only their concentration. Alternatively, in Gillespie's approach each molecule is modelled explicitly and stochastically, capturing the probabilities with which reactions occur, based on the likelihood of molecules of appropriate species being in close proximity. This gives rise to a CTMC, but one whose state space is much too large to be solved explicitly. Hence simulation is the only option, each realisation of the simulation giving rise to one possible behaviour of the system. Thus the results of many runs must be aggregated in order to gain insight into the typical behaviour.

Our approach represents a new alternative which develops a representation of the behaviour of the system which is intermediate between the previous techniques. We retain the stochastic element of Gillespie's approach but the CTMC which we give rise to can be considerably smaller because we model at the level of *species* rather than *molecules*. Keeping the state space manageable means that we are able to solve the CTMC explicitly and avoid the repeated runs necessitated by stochastic simulation. Moreover, in addition to the quantitative analysis on the CTMC, as illustrated here with PEPA, we are able to conduct model checking of stochastic properties of the model. This provides more powerful reasoning mechanisms than stochastic simulation.

In our models the continuous variable, or *concentration*, associated with each species is discretised into a number of *levels*. Thus each component representing a species has a distinct local state for each level of concentration. The more levels that are incorporated into the model, i.e. the finer the granularity of the discretisation, the closer the results of the CTMC will be to the ODE model. However, finer granularity also means that there will be more states in the CTMC. Thus we are faced with a trade-off between accuracy and tractability. Since not all species must have the same degree of discretisation we may choose to represent some aspects of the pathway in finer detail than others.

8.1 Scalability

Whilst being of manageable size from a solution perspective, the CTMCs we are dealing with are too large to contemplate constructing manually. The use of high level modelling languages such as PEPA and PRISM to generate the underlying CTMC allows us to separate system structure from performance. Our style of modelling, focussed on species, or molar concentrations thereof, rather than molecules, means that most reactions involve three or more components. The multi-way synchronisation of PEPA and PRISM is ideally suited to this approach. Ultimately, we will encounter state space explosion, arising from either the granularity of the discretisation or the number of species, but it has not been a problem for the pathways we have studied thus far (with up to approx. 20 species).

8.2 Relationship between PEPA and PRISM

PEPA and PRISM have provided complementary formalisms and toolsets for modelling and reasoning with CTMCs. While models are easily and clearly expressed in PEPA, it is difficult to represent reaction rates accurately. This is not surprising, given PEPA was designed for modelling performance of (bounded capacity) computer systems, not biochemical reactions. PRISM provides a better representation of reaction rates and the facility to check CSL properties. A key feature of both languages is multiway synchronisation, essential for our approach. To some extent the source language is not important, since a PRISM model can be derived automatically from any PEPA model, using the PEPA workbench, though it is necessary to handcode the rates (because PEPA implements synchronisation by minimum, PRISM by product). Here, we have handcoded the PRISM models, to make the concentration variable explicit. This has enabled us to perform experiments easily over a wide range of models.

9 Related and Further Work

Work on applying formal system description techniques from computer science to biochemical signalling pathways was initially stimulated by [GP98, Reg02, RSS01, PRSS01]. Subsequently there has been much work in which the stochastic π-calculus is used to model biological systems, for example [CCDM04] and elsewhere. This work is based on a correspondence between molecules and processes. Each *molecule* in a signalling pathway is represented by a component in the process algebra representation. Thus, in order to represent a system with populations of molecules, many copies of the process algebra components are needed. This leads to underlying CTMC models with enormous state spaces — the only possible solution technique is simulation based on Gillespie's algorithm.

In our approach we have proposed a more abstract correspondence, between *species* and processes (c.f. modelling classes rather than individual objects). Now the components in the process algebra model capture a pattern of behaviour of a whole set of molecules, rather than the identical behaviour of thousands of molecules having to be represented individually. From such models we are able to generate underlying models, suitable for analysis, in a number of different ways. When we consider populations of molecules, considering only two states for each species (*high* and *low*) we are able to generate a set of ODEs from a PEPA model. With a moderate degree of granularity in the discretisation of the concentration we are able to generate an underlying CTMC explicitly. This can then be subjected to steady state or transient numerical analysis, or model checking of temporal properties expressed in CSL, as we have seen. Alternatively, interpreting the high/low model as establishing a pattern of behaviour to be followed by each molecule, we are able to derive a stochastic simulation based on Gillespie's algorithm.

In the recent work by Heath *et al.* [HKN$^+$06, KNP$^+$06], the authors use PRISM to model the FGF signalling pathway. However, they model individuals and do not appear to have a representation of population dynamics.

10 Conclusions

Mathematical biologists are familiar with applying methods based on reaction rate equations and systems of coupled first-order differential equations. They are familiar too with the stochastic simulation methods in the Gillespie family which have their roots in physically rigorous modelling of the phenomena studied in statistical thermodynamics. However, the practice in the field of computational biology is often either to code a system of differential equations directly in a numerical computing platform such as Matlab, or to run a stochastic simulation.

It might be thought that differential equations represent a direct mathematical formulation of a chemical reacting system and might be more straightforward to use than mathematical formulations derived from process algebras.

Set against this though is the absence of a ready apparatus for reasoning about the correctness of an ODE model. No equivalence relations exist to compare models and there is no facility to perform even simple checks such as deadlock detection, let alone more complex static analysis such as liveness or reachability analysis. The same criticisms unfortunately can also be levelled at stochastic simulation.

We might like to believe that there was now sufficient accumulated expertise in computational biological modelling with ordinary differential equations that such mistakes would simply not occur, or we might think that they would be so subtle that modelling in a process algebra such as PEPA or a state-based modelling language such as PRISM could not uncover them. We can however point to at least one counterexample to this. In a recent PEPA modelling study we found an error in the analysis of a published and widely cited ODE model. The authors of [SEJGM02] develop a complex ODE model of epidermal growth factor (EGF) receptor signal pathways in order to give insight into the activation of the MAP kinase cascade through the kinases Raf, MEK and ERK-1/2. Our formalisation in [CDGH06] was able to uncover a previously unexpected error in the way the ODEs had been solved, which led to the production of misleading results. In essence, the error emerged because through the use of a high-level language, we were able to compare different analyses of the same model, and then do some investigations to discover the cause of discrepencies between them.

High-level modelling languages rooted in computer science theory add significantly to the analysis methods which are presently available to practicing computational biologists, increasing the potential for stronger and better modelling practice leading to beneficial scientific discoveries by experimentalists making a positive contribution to improving human and animal health and quality of life. We believe that the insights obtained through the principled application of strong theoretical work stand as a good advertisement for the usefulness of high-level modelling languages for analysing complex biological processes.

Appendix: PRISM model of example pathway

The system description is omitted - it simply runs all modules concurrently. The rate constants are taken from [CSK$^+$03].

```
const int N = 7;
const double M = 2.5/N;

module RAF1
    RAF1: [0..N] init N;
    [r1]  (RAF1 > 0) -> RAF1*M: (RAF1' = RAF1 - 1);
    [r12] (RAF1 > 0) -> RAF1*M: (RAF1' = RAF1 - 1);
    [r2]  (RAF1 < N) -> 1: (RAF1' = RAF1 + 1);
    [r5]  (RAF1 < N) -> 1: (RAF1' = RAF1 + 1);
    [r13] (RAF1 < N) -> 1: (RAF1' = RAF1 + 1);
    [r14] (RAF1 < N) -> 1: (RAF1' = RAF1 + 1);
endmodule

module RKIP
    RKIP: [0..N] init N;
    [r1]  (RKIP > 0) -> RKIP*M: (RKIP' = RKIP - 1);
    [r2]  (RKIP < N) -> 1: (RKIP' = RKIP + 1);
    [r11] (RKIP < N) -> 1: (RKIP' = RKIP + 1);
endmodule

module RAF1/RKIP
    RAF1/RKIP: [0..N] init 0;
    [r1]  (RAF1/RKIP < N) -> 1: (RAF1/RKIP' = RAF1/RKIP + 1);
    [r2]  (RAF1/RKIP > 0) -> RAF1/RKIP*M:
              (RAF1/RKIP' = RAF1/RKIP - 1);
    [r3]  (RAF1/RKIP > 0) -> RAF1/RKIP*M:
              (RAF1/RKIP' = RAF1/RKIP - 1);
    [r4]  (RAF1/RKIP < N) -> 1: (RAF1/RKIP' = RAF1/RKIP + 1);
endmodule

module ERK-PP
    ERK-PP: [0..N] init N;
    [r3]  (ERK-PP > 0) -> ERK-PP*M: (ERK-PP' = ERK-PP - 1);
    [r4]  (ERK-PP < N) -> 1: (ERK-PP' = ERK-PP + 1);
    [r8]  (ERK-PP < N) -> 1: (ERK-PP' = ERK-PP + 1);
endmodule

module RAF1/RKIP/ERK-PP
    RAF1/RKIP/ERK-PP: [0..N] init 0;
    [r3]  (RAF1/RKIP/ERK-PP < N) -> 1:
              (RAF1/RKIP/ERK-PP' = RAF1/RKIP/ERK-PP + 1);
    [r4]  (RAF1/RKIP/ERK-PP > 0) ->
              RAF1/RKIP/ERK-PP*M:
                  (RAF1/RKIP/ERK-PP' = RAF1/RKIP/ERK-PP - 1);
    [r5]  (RAF1/RKIP/ERK-PP > 0) ->
              RAF1/RKIP/ERK-PP*M:
                  (RAF1/RKIP/ERK-PP' = RAF1/RKIP/ERK-PP - 1);
endmodule

module ERK
    ERK: [0..N] init 0;
    [r5]  (ERK < N) -> 1: (ERK' = ERK + 1);
    [r6]  (ERK > 0) -> ERK*M: (ERK' = ERK - 1);
    [r7]  (ERK < N) -> 1: (ERK' = ERK + 1);
endmodule

module RKIP-P
    RKIP-P: [0..N] init 0;
    [r5]  (RKIP-P < N) -> 1: (RKIP-P' =RKIP-P + 1);
    [r9]  (RKIP-P > 0) -> RKIP-P*M: (RKIP-P' =RKIP-P - 1);
    [r10] (RKIP-P < N) -> 1: (RKIP-P' =RKIP-P + 1);
endmodule

module RP
    RP: [0..N] init N;
    [r9]  (RP > 0) -> RP*M: (RP' = RP - 1);
    [r10] (RP < N) -> 1: (RP' = RP + 1);
    [r11] (RP < N) -> 1: (RP' = RP + 1);
endmodule

module MEK
    MEK: [0..N] init N;
    [r12] (MEK > 0) ->  MEK*M: (MEK' = MEK - 1);
    [r13] (MEK < N) -> 1: (MEK' = MEK + 1);
    [r15] (MEK < N) -> 1: (MEK' = MEK + 1);
endmodule

module MEK/RAF1
    MEK/RAF1: [0..N] init N;
    [r14] (MEK/RAF1> 0) ->  MEK/RAF1*M: (MEK/RAF1' = MEK/RAF1 - 1);
    [r15] (MEK/RAF1> 0) ->  MEK/RAF1*M: (MEK/RAF1' = MEK/RAF1 - 1);
    [r12] (MEK/RAF1 < N) -> 1: (MEK/RAF1' = MEK/RAF1 + 1);
endmodule

module MEK-PP
    MEK-PP: [0..N] init N;
    [r6]  (MEK-PP > 0) ->  MEK-PP*M: (MEK-PP' = MEK-PP - 1);
    [r15] (MEK-PP > 0) ->  MEK-PP*M: (MEK-PP' = MEK-PP - 1);
    [r7]  (MEK-PP < N) -> 1: (MEK-PP' = MEK-PP + 1);
    [r8]  (MEK-PP < N) -> 1: (MEK-PP' = MEK-PP + 1);
    [r14] (MEK-PP < N) -> 1: (MEK-PP' = MEK-PP + 1);
endmodule

module  MEK-PP/ERK
    MEK-PP/ERK: [0..N] init 0;
    [r7]  (MEK-PP/ERK > 0) ->  MEK-PP/ERK*M:
              (MEK-PP/ERK' =  MEK-PP/ERK - 1);
    [r8]  (MEK-PP/ERK > 0) ->  MEK-PP/ERK*M:
              (MEK-PP/ERK' =  MEK-PP/ERK - 1);
    [r6]  (MEK-PP/ERK < N) -> 1: (MEK-PP/ERK' =  MEK-PP/ERK + 1);
endmodule

module RKIP-P/RP
    RKIP-P/RP: [0..N] init 0;
    [r9]  (RKIP-P/RP < N) -> 1: (RKIP-P/RP' = RKIP-P/RP + 1);
    [r10] (RKIP-P/RP > 0) -> RKIP-P/RP*M:
              (RKIP-P/RP' = RKIP-P/RP - 1);
    [r11] (RKIP-P/RP > 0) -> RKIP-P/RP*M:
               (RKIP-P/RP' = RKIP-P/RP - 1);
endmodule

module Constants
    x: bool init true;
    [r1]  (x) -> 0.53/M: (x' = true);
    [r2]  (x) -> 0.0072/M: (x' = true);
    [r3]  (x) -> 0.625/M: (x' = true);
    [r4]  (x) -> 0.00245/M: (x' = true);
    [r5]  (x) -> 0.0315/M: (x' = true);
    [r6]  (x) -> 0.8/M: (x' = true);
    [r7]  (x) -> 0.0075/M: (x' = true);
    [r8]  (x) -> 0.071/M: (x' = true);
    [r9]  (x) -> 0.92/M: (x' = true);
    [r10] (x) -> 0.00122/M: (x' = true);
    [r11] (x) -> 0.87/M: (x' = true);
    [r12] (x) -> 0.05/M: (x' = true);
    [r13] (x) -> 0.03/M: (x' = true);
    [r14] (x) -> 0.06/M: (x' = true);
    [r15] (x) -> 0.02/M: (x' = true);
endmodule
```

References

[ASSB00] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model checking continuous time Markov chains. *ACM Transactions on Computational Logic*, 1:162–170, 2000.

[BaHK00] C. Baier, B. Haverkort and. Hermanns, and J.-P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *Computer Aided Verification*, pages 358–372, 2000.

[CCDM04] D. Chiarugi, M. Curti, P. Degano, and R. Marangoni. VICE: A VIrtual CEll. In *Proceedings of the 2nd International Workshop on Computational Methods in Systems Biology*, Paris, France, April 2004. Springer.

[CDGH06] M. Calder, A. Duguid, S. Gilmore, and J. Hillston. Stronger computational modelling of signalling pathways using both continuous and discrete-state methods. In *To appear in Computational Methods in Systems Biology 2006*, LNCS. Springer-Verlag, 2006.

[CE81] E.M. Clarke and E.A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *Logics of Programs: Workshop*, volume 131 of *Lecture Notes in Computer Science*, Yorktown Heights, New York, May 1981. Springer-Verlag.

[CGH05] M. Calder, S. Gilmore, and J. Hillston. Automatically deriving ODEs from process algebra models of signalling pathways. In *Computational Methods in Systems Biology 2005*, pages 204–215. LFCS, University of Edinburgh, 2005.

[CGH06] M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. *Transactions on Computational Biology*, pages 1–23, 2006.

[CSK$^+$03] K.-H. Cho, S.-Y. Shin, H.-W. Kim, O. Wolkenhauer, B. McFerran, and W. Kolch. Mathematical modeling of the influence of RKIP on the ERK signaling pathway. In C. Priami, editor, *Computational Methods in Systems Biology (CSMB'03)*, volume 2602 of *LNCS*, pages 127–141. Springer-Verlag, 2003.

[CVOG06] M. Calder, V. Vyshemirsky, R. Orton, and D. Gilbert. Analysis of signalling pathways using continuous time Markov chains. *Transactions on Computational Biology*, pages 44–67, 2006.

[dJ02] H. de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[EE02] W.H. Elliott and D.C. Elliott. *Biochemistry and Molecular Biology.* Oxford University Press, 2002.

[Gar04] C.W. Gardiner. *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*. Springer, 2004.

[GH94] S. Gilmore and J. Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In *Proceedings of the Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, number 794 in Lecture Notes in Computer Science, pages 353–368, Vienna, May 1994. Springer-Verlag.

[GHR01] S. Gilmore, J. Hillston, and M. Ribaudo. An efficient algorithm for aggregating PEPA models. *IEEE Transactions on Software Engineering*, 27(5):449–464, May 2001.

[Gil77] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340 –2361, 1977.

[Gil91] Daniel T. Gillespie. *Markov Processes: An Introduction for Physical Scientists*. Academic Press, 1991.

[GP98] P.J.E. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic petri nets. *Proceedings of National Academy of Science, USA*, 95(12):7650–6755, June 1998.

[Hil96] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[HKN$^+$06] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. In C. Priami, editor, *Proceedings of 4th International Workshop on Computational Methods in Systems Biology*, volume 4210 of *Lecture Notes in Bioinformatics*, pages 32–47, Trento, Italy, 18-19th October 2006. Springer-Verlag.

[KNP02] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In T. Field, P. Harrison, J. Bradley, and U. Harder, editors, *Proc. 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS'02)*, volume 2324 of *LNCS*, pages 200–204. Springer, 2002.

[KNP$^+$06] Marta Kwiatkowska, Gethin Norman, David Parker, Oksana Tymchyshyn, John Heath, and Eamonn Gaffney. Simulation and verification for computational modelling of signalling pathways. In *Proceedings of the 2006 Winter Simulation Conference*, 2006. To appear.

[KS60] J.G. Kemeny and J.L. Snell. *Finite Markov Chains.* Van Nostrand, 1960.

[LS91] K. Larsen and A. Skou. Bisimulation through Probabilistic Testing. *Information and Computation*, 94(1):1–28, September 1991.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[Nor97] J.R. Norris. *Markov Chains.* Cambridge University Press, 1997.

[PNK04] D. Parker, G. Norman, and M. Kwiatkowska. PRISM 2.1 Users' Guide. The University of Birmingham, September 2004.

[PRSS01] C. Priami, A. Regev, W. Silverman, and E. Shapiro. Application of a stochastic name passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80:25–31, 2001.

[Reg02] A. Regev. *Computational Systems Biology: a Calculus for Biomolecular Knowledge.* PhD thesis, Tel Aviv University, 2002.

[RSS01] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using π-calculus process algebra. In *Pacific Symposium on Biocomputing 2001 (PSB 2001)*, pages 459–470, 2001.

[SEJGM02] B. Schoeberl, C. Eichler-Jonsson, E.D. Gilles, and G. Muller. Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors. *Nature Biotechnology*, 20:370–375, 2002.

[Ste94] W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1994.

[Voi00] E. O. Voit. *Computational Analysis of Biochemical Systems.* Cambridge University Press, 2000.