Quantitative Verification Techniques for Biological Processes

Marta Kwiatkowska, Gethin Norman, and David Parker

Oxford University Computing Laboratory,
Wolfson Building, Parks Road, Oxford, OX1 3QD
{marta.kwiatkowska,gethin.norman,david.parker}@comlab.ox.ac.uk

Summary. Probabilistic model checking is a formal verification framework for systems which exhibit stochastic behaviour. It has been successfully applied to a wide range of domains, including security and communication protocols, distributed algorithms and power management. In this chapter we demonstrate its applicability to the analysis of biological pathways and show how it can yield a better understanding of the dynamics of these systems. Through a case study of the MAP (Mitogen–Activated Protein) Kinase cascade, we explain how biological pathways can be modelled in the probabilistic model checker PRISM and how this enables the analysis of a rich selection of quantitative properties.

1 Introduction

Recent research has had considerable success adapting approaches from computer science to the analysis of biological systems and, in particular, biochemical pathways. The fundamental theory behind the majority of this work is the simulation-based techniques for discrete stochastic models originally introduced by Gillespie [9]. This models the evolution of individual molecules, whose rates of interaction are controlled by exponential distributions, and differs from the principal alternative modelling paradigm of pathways, using ordinary differential equations to model the evolution of average molecular concentrations over time. We adopt the stochastic modelling approach but, by employing formal verification techniques, compute *exact* quantitative measures as opposed to taking averages over sets of simulation runs.

In this chapter we demonstrate how probabilistic model checking [2, 20, 32] and the probabilistic model checker PRISM [14, 27] can be employed as a framework for the modelling and analysis of biological pathways. This approach is motivated by both the fact that PRISM has already been successfully applied to the study of biological pathways, see for example [11, 4, 30], and previous work which has demonstrated the applicability of probabilistic model checking to the analysis of a wide variety of complex stochastic systems, see for example [18].

This framework inherits many of the advantages of model checking, including the use of a both a formal model and specification of the system under study and the fact that the approach is exhaustive, analysing all possible behaviours of the system. We are also able to re-use existing technology, exploiting the efficient implementations and tool support developed for probabilistic model checkers such as PRISM. The intention is that probabilistic model checking should be used in conjunction with other, well-established approaches for analysing pathways based on simulation and differential equations. In combination, these techniques can offer greater insight into the complex interactions present in biological pathways.

Outline. In the next section, we give an overview of probabilistic model checking and the tool PRISM. Section 3 presents the MAPK cascade, discusses how the pathway can be modelled in the PRISM language and demonstrates how PRISM can be used to specify and analyse a wide range of quantitative properties. In Sect. 4 we discuss related work and Sect. 5 concludes the chapter.

2 Probabilistic Model Checking

Probabilistic model checking is a formal verification technique for the modelling and analysis of systems that exhibit stochastic behaviour. This technique is a variant of *model checking*, a well-established and widely-used formal method for ascertaining the correctness of real-life systems. Model checking requires two inputs:

- a description of the system, usually given in some high-level modelling formalism such as a Petri net or process algebraic expression;
- a specification of one or more desired properties of the system, normally using temporal logics such as CTL (Computation Tree Logic) or LTL (Linear-time Temporal Logic).

From these inputs, a *model checker* can construct a model of the system, typically a labelled state–transition system in which each state represents a possible configuration and each transition represents an evolution of the system from one configuration to another over time. It is then possible to automatically verify whether or not each property is satisfied, based on a systematic and exhaustive exploration of the constructed state–transition system.

In probabilistic model checking, the models are augmented with quantitative information regarding the likelihood that transitions occur and the times at which they do so. In practice, these models are typically Markov chains or Markov decision processes. To model biological pathways, the appropriate model is *continuous-time Markov chains* (CTMCs), in which transitions between states are assigned (positive, real-valued) rates. These values are interpreted as the rates of negative exponential distributions.

Formally, letting $\mathbb{R}_{\geq 0}$ denote the set of non-negative reals and AP be a fixed, finite set of atomic propositions used to label states with properties of interest, a CTMC is a tuple $(S, \mathbf{R}, L)$ where:

- S is a finite set of *states*;
- $\mathbf{R} : (S \times S) \rightarrow \mathbb{R}_{\geq 0}$ is a *transition rate matrix*;
- $L : S \rightarrow 2^{AP}$ is a *labelling* function which associates each state with a set of atomic propositions.

The transition rate matrix $\mathbf{R}$ assigns rates to each pair of states, which are used as parameters of the exponential distribution. A transition can only occur between states s and s' if $\mathbf{R}(s, s'){>}0$ and, in this case, the probability of the transition being triggered within t time-units equals $1 - \exp(-\mathbf{R}(s, s'){\times}t)$. Typically, in a state s, there is more than one state s' for which $\mathbf{R}(s, s'){>}0$; this is known as a *race condition* and the first transition to be triggered determines the next state. The time spent in state s before any such transition occurs is exponentially distributed with the rate $E(s) = \sum_{s' \in S} \mathbf{R}(s, s')$, called the *exit rate* of stat s. The probability of moving to state s' is given by $\mathbf{R}(s, s')/E(s)$.

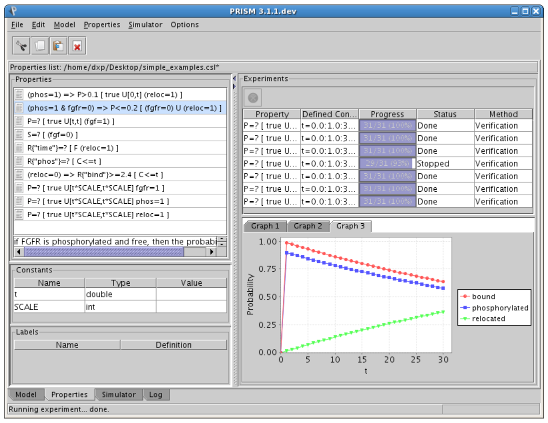
A CTMC can be augmented with *rewards*, attached to states and/or transitions of the model. Formally, a *reward structure* for a CTMC is a pair $(\boldsymbol{\rho}, \boldsymbol{\iota})$ where:

- $\boldsymbol{\rho} : S \rightarrow \mathbb{R}_{\geq 0}$ is a *state reward function*;
- $\boldsymbol{\iota} : (S \times S) \rightarrow \mathbb{R}_{\geq 0}$ is a *transition reward function.*

State rewards can represent either a quantitative measure of interest at a particular time instant (e.g. the number of phosphorylated proteins in the system) or the rate at which some measure accumulates over time (e.g. energy dissipation). Transition rewards are accumulated each time a transition occurs and can be used to compute, e.g. the number of protein bindings over a particular time period.

Properties of CTMCs are, like in non-probabilistic model checking, expressed in temporal logic, but are now quantitative in nature. For this, we use probabilistic temporal logics such as CSL [1, 2] and its extensions for reward-based properties [20]. For example, rather than verifying that 'the protein always eventually degrades', using CSL allows us to ask 'what is the probability that the protein eventually degrades?' or 'what is the probability that the protein degrades within t hours?'. Reward-based properties include 'what is the expected number of phosphorylations within the first t time units?' and 'what is the expected time that proteins spend bound before relocation occurs?'. For further details on probabilistic model checking of CTMCs, see for example [2, 20, 32].

PRISM [14, 27] is a probabilistic model checking tool developed at the Universities of Birmingham and Oxford. It provides support for several types of probabilistic models, including CTMCs. Models are specified in a simple, state-based language based on guarded commands. PRISM's notation



Properties list: /home/dxp/Desktop/simple_examples.csl*

Properties

```
(phos=1) => P>0.1 [ true U[0,t] (reloc=1) ]
(phos=1 & fgfr=0) => P<=0.2 [ (fgfr=0) U (reloc=1) ]
P=? [ true U[t,t] (fgf=1) ]
S=? [ (fgf=0) ]
R{"time"}=? [ F (reloc=1) ]
R{"phos"}=? [ C<=t ]
(reloc=0) => R{"bind"}>=2.4 [ C<=t ]
P=? [ true U[t*SCALE,t*SCALE] fgfr=1 ]
P=? [ true U[t*SCALE,t*SCALE] phos=1 ]
P=? [ true U[t*SCALE,t*SCALE] reloc=1 ]
```

if FGFR is phosphorylated and free, then the probabi

Constants

Name	Type	Value
t	double	
SCALE	int	

Labels

Name	Definition

Property	Defined Con...	Progress	Status	Method
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification
P=? [true U...	t=0.0:1.0:3...	29/31 (93%)	Stopped	Verification
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification
P=? [true U...	t=0.0:1.0:3...	31/31 (100%)	Done	Verification

Running experiment... done.

Fig. 1. A screenshot of the PRISM graphical user interface

for specifying properties of CTMCs incorporates the reward-based extension ([20]) of CSL. Figure 1 shows a screenshot of PRISM in action.

The underlying computation in PRISM involves a combination of:

- *graph-theoretical algorithms*, for conventional temporal logic model checking and *qualitative* probabilistic model checking;
- *numerical computation*, for *quantitative* probabilistic model checking, i.e. calculation of probabilities and reward values.

Graph-theoretical algorithms are comparable to the operation of a conventional, non-probabilistic model checker. For numerical computation, PRISM typically solves linear equation systems or performs transient analysis. Due to the size of the models that need to be handled, the tool uses iterative methods rather than direct methods. For solution of linear equation systems, it supports a range of well-known techniques including the Jacobi, Gauss-Seidel and SOR (successive over-relaxation) methods; for transient analysis of CTMCs, it employs uniformisation.

One of the most notable features of PRISM is that it uses state-of-the-art *symbolic* approaches, using data structures based on binary decision diagrams [17, 25]. These allow for compact representation and efficient manipulation of large, structured models by exploiting regularities exhibited in the high-level modelling language descriptions. The tool actually provides three distinct *engines* for numerical solution: the first is purely symbolic; the second uses sparse matrices; and the third is a hybrid, using a combination of the two. The result is a flexible implementation which can be adjusted to improve performance depending on the type of models and properties being analysed.

PRISM also incorporates a discrete-event simulation engine. This allows approximate solutions to be generated for the numerical computations that underlie the model checking process, by applying Monte Carlo methods and sampling. These techniques offer increased scalability, at the expense of numerical accuracy. Using the same underlying engine, PRISM includes a tool

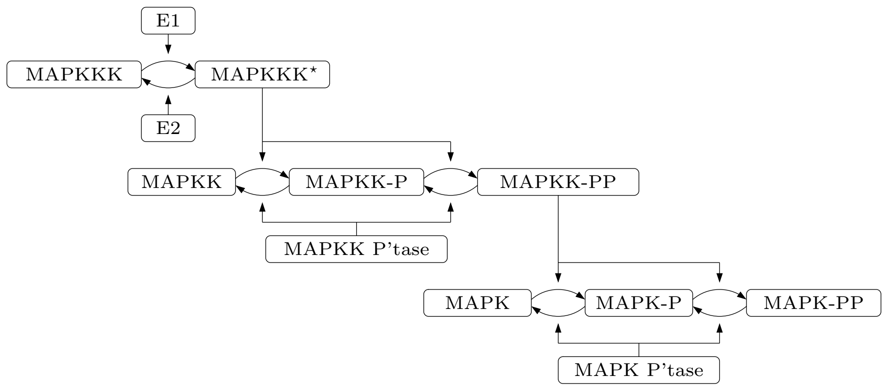


Fig. 2. MAPK cascade pathway

to perform manual execution and debugging of models. Other functionality provided by the user interface of the tool includes a graph-plotting component for visualisation of numerical results and editors for the model and property specification languages.

3 Case Study: MAPK Cascade

We demonstrate the application of probabilistic model checking to the modelling, specification and analysis of biological pathways through a case study: the MAPK cascade.

The MAP (Mitogen–Activated Protein) Kinases are involved in a pathway through which information is sent to the nucleus. It is one of the most important signalling pathways, playing a pivotal role in the molecular signalling that governs the growth, proliferation and survival of many cell types. The MAPK cascade consists of a MAPK Kinase Kinase (MAPKKK), a MAPK Kinase (MAPKK) and a MAPK. The cascade is initialised through the phosphorylation of MAPKKK, which then activates MAPKK through phosphorylation at two serine residues. This then activates MAPK through phosphorylation at theronine and tyrosine residues. The initialisation of the pathway can be caused by a diverse set of stimuli including growth factors, neurotransmitters and cytokines.

Figure 2 gives an overview of the structure of the pathway and Fig. 3 details the reactions that form the cascade, as taken from [15]. In the reactions presented in Fig. 3, it is assumed that the phosphorylation of both MAPK and MAPKK occur in two distributed steps. For example, when MAPK collides with its activator (MAPKK-PP) the first phosphorylation (MAPK-P) occurs and the activator is released. The phosphorylated MAPK must then collide again with its activator for the second phosphorylation (MAPK-PP) to occur. The deactivation of phosphorylated MAPK and MAPKK is caused

1. MAPKKK is activated through enzyme E1	
KKK + E1 $\rightarrow$ KKK:E1	$a_1 = 1\,\text{nM}^{-1}\,\text{s}^{-1}$
KKK + E1 $\leftarrow$ KKK:E1	$d_1 = 150\,\text{s}^{-1}$
KKK:E1 $\rightarrow$ KKK* + E1	$k_1 = 150\,\text{s}^{-1}$
2. MAPKKK is deactivated through enzyme E2	
KKK* + E2 $\rightarrow$ KKK*:E2	$a_2 = 1\,\text{nM}^{-1}\,\text{s}^{-1}$
KKK* + E2 $\leftarrow$ KKK*:E2	$d_2 = 150\,\text{s}^{-1}$
KKK*:E2 $\rightarrow$ KKK + E2	$k_2 = 150\,\text{s}^{-1}$
3. MAPKK is activated by MAPKKK*	
KK + KKK* $\rightarrow$ KK:KKK*	$a_3 = 1\,\text{nM}^{-1}\,\text{s}^{-1}$
KK + KKK* $\leftarrow$ KK:KKK*	$d_3 = 150\,\text{s}^{-1}$
KK:KKK* $\rightarrow$ KK-P + KKK*	$k_3 = 150\,\text{s}^{-1}$
4. MAPKK-P is deactivated by MAPKK phosphatase	
KK-P + KK-Ptase $\rightarrow$ KK-P:KK-Ptase	$a_4 = 1\,\text{nM}^{-1}\,\text{s}^{-1}$
KK-P + KK-Ptase $\leftarrow$ KK-P:KK-Ptase	$d_4 = 150\,\text{s}^{-1}$
KK-P:KK-Ptase $\rightarrow$ KK + KK-Ptase	$k_4 = 150\,\text{s}^{-1}$
5. MAPKK-P is activated by MAPKKK*	
KK-P + KKK* $\rightarrow$ KK-P:KKK*	$a_5 = 1\,\text{nM}^{-1}\,\text{s}^{-1}$
KK-P + KKK* $\leftarrow$ KK-P:KKK*	$d_5 = 150\,\text{s}^{-1}$
KK-P:KKK* $\rightarrow$ KK-PP + KKK*	$k_5 = 150\,\text{s}^{-1}$
6. MAPKK-PP is deactivated by MAPKK phosphatase	
KK-PP + KK-Ptase $\rightarrow$ KK-PP:KK-Ptase	$a_6 = 1\,\text{nM}^{-1}\,\text{s}^{-1}$
KK-PP + KK-Ptase $\leftarrow$ KK-PP:KK-Ptase	$d_6 = 150\,\text{s}^{-1}$
KK-PP:KK-Ptase $\rightarrow$ KK-P + KK-Ptase	$k_6 = 150\,\text{s}^{-1}$
7. MAPK is activated by MAPKK-PP	
K + KK-PP $\rightarrow$ K:KK-PP	$a_7 = 1\,\text{nM}^{-1}\,\text{s}^{-1}$
K + KK-PP $\leftarrow$ K:KK-PP	$d_7 = 150\,\text{s}^{-1}$
K:KK-PP $\rightarrow$ K-P + KK-PP	$k_7 = 150\,\text{s}^{-1}$
8. MAPK-P is deactivated by MAPK phosphatase	
K-P + K-Ptase $\rightarrow$ K-P:K-Ptase	$a_8 = 1\,\text{nM}^{-1}\,\text{s}^{-1}$
K-P + K-Ptase $\leftarrow$ K-P:K-Ptase	$d_8 = 150\,\text{s}^{-1}$
K-P:K-Ptase $\rightarrow$ K + K-Ptase	$k_8 = 150\,\text{s}^{-1}$
9. MAPK-P is activated by MAPKK-PP	
K-P + KK-PP $\rightarrow$ K-P:KK-PP	$a_9 = 1\,\text{nM}^{-1}\,\text{s}^{-1}$
K-P + KK-PP $\leftarrow$ K-P:KK-PP	$d_9 = 150\,\text{s}^{-1}$
K-P:KK-PP $\rightarrow$ K-PP + KK-PP	$k_9 = 150\,\text{s}^{-1}$
10. MAPK-P is deactivated by MAPK phosphatase	
K-PP + K-Ptase $\rightarrow$ K-PP:K-Ptase	$a_{10} = 1\,\text{nM}^{-1}\,\text{s}^{-1}$
K-PP + K-Ptase $\leftarrow$ K-PP:K-Ptase	$d_{10} = 150\,\text{s}^{-1}$
K-PP:K-Ptase $\rightarrow$ K-P + K-Ptase	$k_{10} = 150\,\text{s}^{-1}$

Fig. 3. MAPK cascade reactions

by the corresponding phosphatase, while the activation and deactivation of MAPKKK is through the enzymes E1 and E2 respectively. To simplify the presentation in Fig. 3 we denote MAPK, MAPKK and MAPKKK by K, KK and KKK respectively.

The kinetic rates given in Fig. 3 are based on the data presented in [15] where it is assumed that the K_m values ($K_m = (d_m + k_m)/a_m$) for phosphorylation and dephosphorylation of MAPK, MAPKK and MAPKKK all equal 300 nM.

3.1 Specifying the Model

We now outline how to construct a *discrete stochastic* model of the MAPK cascade reactions from Fig. 3 in the modelling language of the PRISM tool.

The applicability of probabilistic model checking and PRISM follows from the fact that the underlying model can be shown to be a CTMC, in which the stochastic rates associated with each transition can be derived from the kinetic rates of the reactions. In the case of unary reactions, the stochastic rate equals the kinetic rate. On the other hand, for binary reactions, if the kinetic rate is given in terms of molar concentrations, then the stochastic rate can be obtained by dividing by $\mathit{Vol} \times \mathcal{N}_A$ where Vol is the volume and $\mathcal{N}_A$ is Avogadro's number. For a more detailed discussion of the relationship between kinetic and stochastic rates, see for example [36, 9].

A model described in the PRISM language comprises a set of *modules*, the state of each being represented by a valuation over a set of finite-ranging *variables*. The global state of the model is determined by a valuation over the union of all variables (denoted V). The atomic propositions of the model are given by predicates over the variables V and the labelling function assigns to each state the predicates that it satisfies.

The behaviour of a module, i.e. the changes in state which it can undergo, is specified by a number of *guarded commands* of the form:

$$[\mathit{act}]\ \mathit{guard}\ \rightarrow\ \mathit{rate}\ :\ \mathit{update};$$

where *act* is an (optional) action label, *guard* is a predicate over the variables V, *rate* is a (non-negative) real-valued expression and *update* is of the form:

$$(x_1'{=}u_1)\ \&\ (x_2'{=}u_2)\ \&\ \ldots\ \&\ (x_n'{=}u_n)$$

where $u_1, u_2, \ldots, u_k$ are functions over V and $x_1, x_2, \ldots, x_n$ are variables of the module. Intuitively, in global state s (i.e. a valuation over the variables V) of the PRISM model, the command is enabled if s satisfies the predicate *guard*. If a command is enabled, a transition that updates the module's variables according to *update* (i.e. for $1 \leq i \leq n$ the variable x_i is updated to the value $u_i(s)$) can occur with rate *rate*. When multiple commands with the same update are enabled, the corresponding transitions are combined into a single transition whose rate is the sum of the individual rates.

To model interactions where the state of several modules changes simultaneously, we use *synchronisation*, through the action labels that can be included in the guarded commands. The rate of the combined transition is defined as the product of the rates for each command. As we will see below, the rate of the combined transition is often fully specified in one module and rates omitted from the other modules (this yields the correct rate since PRISM assigns a rate of 1 to any command for which none is specified).

When building a PRISM model of a biological pathway, it is possible to construct an *individual-based* model which provides a detailed model of the evolution of individual molecular components. However, taking this approach comes at a cost: it will inevitably suffer from the well known state-space explosion problem where, as the complexity of the system increases, the state space of the underlying model grows exponentially.

An alternative is to employ a *population-based* approach where the number of each type of molecule or species is modelled, rather than the state of each individual component. Such an approach leads to a much smaller state-space (see for example [11]) while still including sufficient detail to express the properties of interest. For these reasons, it is this approach that we use here.

For the PRISM language, a population-based model can be expressed naturally by using the variables of modules as counters, i.e. there is a variable for each of the possible species in the system which keeps count of the number of that species that are currently present.

In Fig. 4, we present the module representing quantities of the species relating to MAPK and, in Fig. 5, the module representing MAPK phosphatase. The whole cascade could have been specified in one single large PRISM module. However, there is a natural separation of the different elements in the cascade (those relating to MAPKKK, MAPKK, MAPK, MAPKK phosphatase, MAPK phosphatase, E1 and E2) and defining the system using individual modules based on this separation makes the description simpler, easier to understand and less prone to modelling errors. This fact can be seen in other PRISM language models of biological pathways, see for example [11, 4, 27]. The complete PRISM description of the MAPK cascade is available from the case study repository on the PRISM website [27].

As can be seen in Figs. 4 and 5, we have specified that there are initially N inactive MAPKs (the initial value of the variable k is N) and M MAPK phosphatases (the initial value of *kptase* is M). The actual values of N and M have been left undefined since, as will be seen later, this allows these parameters to be varied during model checking.

The values for stochastic reaction rates of the system are defined as constants (see the top of Fig. 4). Notice that the stochastic rates of the binary reactions (i.e. those specified by the constants *a7*, *a8*, *a9* and *a10*) are obtained from the kinetic rates by dividing by the initial number of MAPKs (i.e. N). This is because (recall the discussion of computing stochastic reaction rates earlier in this section) we make the assumption that the volume of the system is proportional to the initial number of MAPKs. It would also have been possible to leave some of the constants for the stochastic rates unspecified and then vary these during verification.

Figures 4 and 5 also show that the modules for MAPK and MAPK phosphatase synchronise through the actions *a_k_ptase*, *d_k_ptase* and *k_k_ptase*, which correspond to the deactivation of MAPK (as described in reactions 8 and 10 of Fig. 3). The actions *a_k_kk*, *d_k_kk* and *k_k_kk*, which appear in the module for MAPK (Fig. 4), correspond to the activation of MAPK by MAPKK-PP (see reactions 7 and 9 of Fig. 3), and there are corresponding commands in the module for MAPKK.

When using a population-based approach, we must ensure that the rates of the CTMC take into account the different possible interactions that can occur. For example, if there are three activated MAPKs (k_pp_1, k_pp_2 and k_pp_3)

```
const int N; // initial amount of MAPK

// stochastic reaction rates
const double a7=1/N; const double d7=150; const double k7=150;
const double a8=1/N; const double d8=150; const double k8=150;
const double a9=1/N; const double d9=150; const double k9=150;
const double a10=1/N; const double d10=150; const double k10=150;

module MAPK

    k : [0..N] init N; // quantity of MAPK
    k_kkpp : [0..N] init 0; // quantity of MAPK:MAPKK-PP
    kp : [0..N] init 0; // quantity of MAPK-P
    kp_kkpp : [0..N] init 0; // quantity of MAPK-P:MAPKK-PP
    kp_ptase : [0..N] init 0; // quantity of MAPK-P:MAPK phosphatase
    kpp : [0..N] init 0; // quantity of MAPK-PP
    kpp_ptase : [0..N] init 0; // quantity of MAPK-PP:MAPK phosphatase

    // reaction 7 (MAPK is activated by MAPKK-PP)
    [a_k_kk] k>0 & k_kkpp<N
                → a7 ∗ k : (k_kkpp′=k_kkpp + 1) & (k′=k − 1);
    [d_k_kk] k<N & k_kkpp>0
                → d7 ∗ k_kkpp : (k_kkpp′=k_kkpp − 1) & (k′=k + 1);
    [k_k_kk] k_kkpp>0 & kp<N
                → k7 ∗ k_kkpp : (k_kkpp′=k_kkpp − 1) & (kp′=kp + 1);
    // reaction 8 (MAPK-P is deactivated by MAPK phosphatase)
    [a_k_ptase] kp>0 & kp_ptase<N
                → a8 ∗ kp : (kp_ptase′=kp_ptase + 1) & (kp′=kp − 1);
    [d_k_ptase] kp<N & kp_ptase>0
                → d8 ∗ kp_ptase : (kp_ptase′=kp_ptase − 1) & (kp′=kp + 1);
    [k_k_ptase] kp_ptase>0 & k<N
                → k8 ∗ kp_ptase : (kp_ptase′=kp_ptase − 1) & (k′=k + 1);
    // reaction 9 (MAPK-P is activated by MAPKK-PP)
    [a_k_kk] kp>0 & kp_kkpp<N
                → a9 ∗ kp : (kp_kkpp′=kp_kkpp + 1) & (kp′=kp − 1);
    [d_k_kk] kp<N & kp_kkpp>0
                → d9 ∗ kp_kkpp : (kp_kkpp′=kp_kkpp − 1) & (kp′=kp + 1);
    [k_k_kk] kp_kkpp>0 & kpp<N
                → k9 ∗ kp_kkpp : (kp_kkpp′=kp_kkpp − 1) & (kpp′=kpp + 1);
    // reaction 10 (MAPK-PP is deactivated by MAPK phosphatase)
    [a_k_ptase] kpp>0 & kpp_ptase<N
                → a10 ∗ kpp : (kpp_ptase′=kpp_ptase + 1) & (kpp′=kpp − 1);
    [d_k_ptase] kpp<N & kpp_ptase>0
                → d10 ∗ kpp_ptase : (kpp_ptase′=kpp_ptase − 1) & (kpp′=kpp + 1);
    [k_k_ptase] kpp_ptase>0 & kp<N
                → k10 ∗ kpp_ptase : (kpp_ptase′=kpp_ptase − 1) & (kp′=kp + 1);

endmodule
```

Fig. 4. PRISM module representing quantities of species relating to MAPK

and two MAPK phosphatases ($kptase_1$ and $kptase_2$) then there are six different species that can be formed: $k\_pp_1\colon kptase_1$, $k\_pp_1\colon kptase_2$, $k\_pp_2\colon kptase_1$, $k\_pp_2\colon kptase_2$, $k\_pp_3\colon kptase_1$ and $k\_pp_3\colon kptase_2$. The reaction rate is thus proportional to both the number of activated MAPKs and the number of MAPK phosphatases. This is straightforward to achieve in the PRISM modelling language since PRISM multiplies rates when modules synchronise: in this case, we set the rates to $a10 \times kpp$ and $kptase$ in the modules *MAPK* (Fig. 4) and *KPTASE* (Fig. 5), respectively.

```
const int M; // initial amount of MAPK phosphatase

module KPTASE

    kptase : [0..M] init M; // amount of MAPK phosphatase

    // reactions 8 and 10 (MAPK/MAPK-P is deactivated by MAPK phosphatase)
    [a_k_ptase] kptase>0  →  kptase : (kptase′=kptase − 1);
    [d_k_ptase] kptase<M  →  1 : (kptase′=kptase + 1);
    [k_k_ptase] kptase<M  →  1 : (kptase′=kptase + 1);

endmodule
```

Fig. 5. PRISM module representing quantity of MAPK phosphatase

```
rewards "activated"

    true : kpp;

endrewards
```

```
rewards "percentage"

    true : 100∗(kpp/N);

endrewards
```

```
rewards "reactions"
    [a_k_kk] true : 1;
    [d_k_kk] true : 1;
    [k_k_kk] true : 1;
endrewards
```

```
rewards "time"

    true : 1;

endrewards
```

Fig. 6. Reward structures for the cascade

3.2 Specifying Rewards

Rewards are PRISM's mechanism for describing additional quantitative measures of probabilistic models. In this section we explain how to specify reward structures for the PRISM model of the MAPK cascade presented in the previous section. Reward structures in PRISM are described using the construct:

`rewards` "*reward_name*" ... `endrewards`

comprising one or more state-reward items of the form:

guard : *reward*;

and/or transition-reward items of the form:

[*act*] *guard* : *reward*;

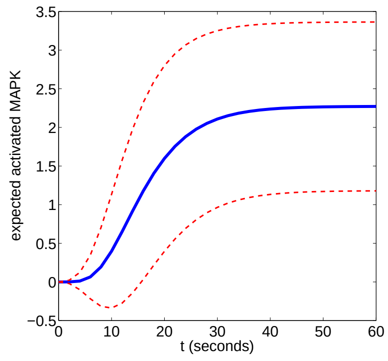
where *guard* is a predicate (over the variables V of the model), *act* is an action label appearing in the commands of the model and *reward* is a real-valued expression (which can contain variables and constants from the model). A state-reward item assigns a state reward of *reward* to all states satisfying *guard* and a transition-reward item assigns a transition reward of *reward* to all *act*-labelled transitions from states satisfying *guard*. Multiple rewards (from different reward items) for a single state or transition are summed and states or transitions with no assigned reward are assumed to have reward 0.

In Fig. 6, we present four different reward structures for the PRISM model of the cascade. The first reward structure ("*activated*") assigns a state reward equal to the amount of MAPK that is activated while the second reward structure ("*percentage*") assigns a state reward equal to the percentage of MAPK that is activated. These can be used to compute the expected amount/percentage of activated MAPK at some time instant or in the long run. The third reward structure "*reactions*" assigns a reward of 1 to all transitions which correspond to a reaction between MAPK and MAPKK. This can be used to compute the expected number of such reactions within a particular period of time or on average (in the long run). The final reward structure ("*time*") simply assigns a state reward of 1 to all states in the model which can be used, for example, to analyse the total expected time before an event/reaction occurs or a certain configuration is reached.

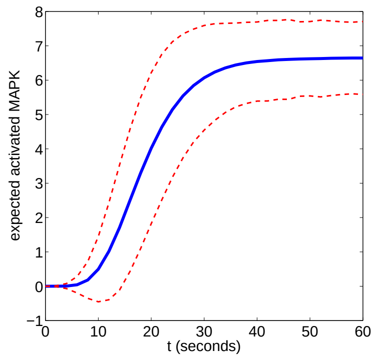
3.3 Specifying Properties

The temporal logic CSL, originally introduced by Aziz et al. [1] and since extended by Baier et al. [2], is based on the temporal logics CTL [5] and PCTL [10]. It provides a powerful means of specifying a variety of performance measures on CTMCs. PRISM use an extended version [20] which also allows for the specification of reward properties. We now give a number of examples of such specifications relating to the PRISM model and reward structures for the MAPK cascade presented in the previous sections. Recall that, in a PRISM model, atomic propositions are given by predicates over the variables of the model.

- $(kkpp{=}N \wedge kpp{=}0) \rightarrow \mathcal{P}_{\geq 0.12}[\ (kkpp{>}0)\ \mathcal{U}\ (kpp{>}0)\ ]$ - if all MAPKKs are activated and none of the MAPKs are activated, then the probability that, while some MAPKKs remain activated, a MAPK becomes activated is at least 0.12.
- $\mathcal{P}_{=?}[\ \texttt{true}\ \mathcal{U}^{[t,t]}\ ((kpp{+}kkpp){=}l)\ ]$ - what is the probability that the total number of MAPKs and MAPKKs activated at time instant t equals l?
- $(kkkp{>}0 \wedge kpp{=}0) \rightarrow \mathcal{P}_{\leq 0.7}[\ (kpp{=}0)\ \mathcal{U}^{[t_1,t_2]}\ (kpp{>}0)\ ]$ - if some MAPKKKs are activated and no MAPKs are activated, then the probability that the first time a MAPK gets activated is within the time interval $[t_1, t_2]$ is at most 0.7.
- $(k{=}0) \rightarrow \mathcal{P}_{\leq 0.01}[\ (k{=}0)\ \mathcal{U}^{[t,\infty)}\ (k{>}0)\ ]$ - if there are no inactive MAPKs, then the probability that some MAPK is deactivated for the first time after time t is at most 0.01.
- $\mathcal{S}_{=?}[\ (kpp{=}l)\ ]$ - what is the probability that in the long run there are precisely l MAPKs activated?
- $\mathcal{R}_{\{\text{"}reactions\text{"}\}=?}[\ \mathcal{C}^{\leq t}\ ]$ - what is the expected number of reactions between MAPKs and MAPKKs during the first t seconds?
- $(kpp{=}N) \rightarrow \mathcal{R}_{\{\text{"}activated\text{"}\}\geq N/2}[\ \mathcal{I}^{=t}\ ]$ - if all MAPKs are activated, then after t seconds the expected number of activated MAPK is at least half of the total number of MAPK.
- $\mathcal{R}_{\{\text{"}reactions\text{"}\}=?}[\ \mathcal{F}\ (kpp{=}N)\ ]$ - what is the expected number of reactions between MAPK and MAPKK before all MAPKs are activated at the same time instant?

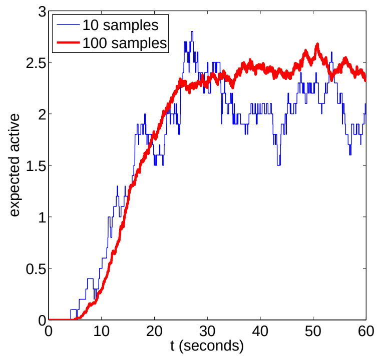


(a) $N = 4$

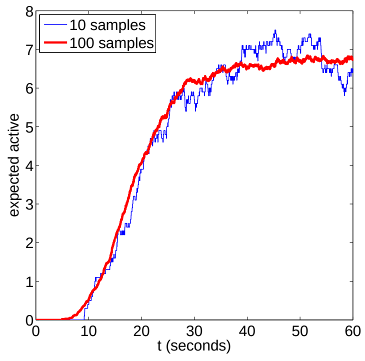


(b) $N = 8$

Fig. 7. Expected activated MAPK at time t ($\mathcal{R}_{\{\text{“}activated\text{”}\}=?}[\ \mathcal{I}^{=t}\ ]$)



(a) $N = 4$



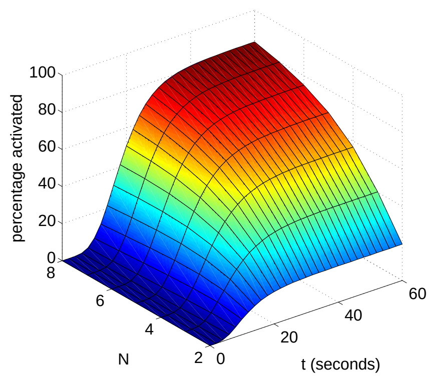
(b) $N = 8$

Fig. 8. Simulation results for amount of activated MAPK at time t

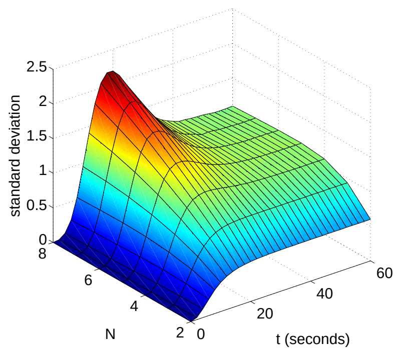
- $(kpp{>}0) \rightarrow \mathcal{R}_{\{\text{“}time\text{”}\}\leq 120}[\ \mathcal{F}\ (k{=}N)\ ]$ - if some MAPKs are activated, the expected time until all of the MAPKs become deactivated at the same time instant at most 120 seconds.
- $\mathcal{R}_{\{\text{“}percentage\text{”}\}\geq 98}[\ \mathcal{S}\ ]$ - in the long run, at least 98% of MAPK is activated.
- $\mathcal{R}_{\{\text{“}reactions\text{”}\}=?}[\ \mathcal{S}\ ]$ - what is the long-run average rate of reactions between MAPK and MAPKK?

3.4 Results and Analysis

When analysing quantitative properties such as those listed above, it is often beneficial to study trends resulting from variations in parameters either from the model (e.g. initial species concentrations or reaction rates) or from the



(a) Percentage



(b) Standard devation

Fig. 9. Expected activated MAPK at time t and corresponding standard deviation values

property specification (e.g. a time bound). Performing analysis in this way is more likely to provide insight into the dynamics of the model or to identify interesting or anomalous behaviour.

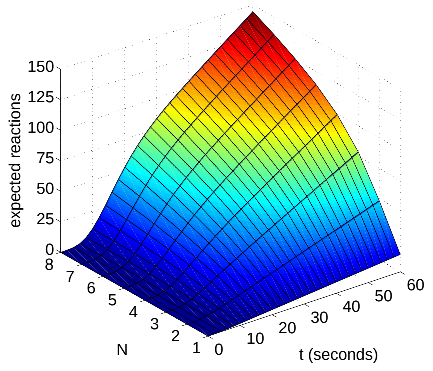
To illustrate this, Fig. 7 shows results obtained with PRISM for the MAPK cascade case study when considering the expected amount of activated MAPK at time instant t, as t varies. The initial quantities of MAPK, MAPKK and MAPKKK (denoted N) are 4 for Fig. 7(a) and 8 for Fig. 7(b). The initial quantity of all remaining species in the cascade (the enzymes E1 and E2 and the phosphatases for MAPK and MAPKK) is 1. The plots in Fig. 7 also show the standard deviation of the random variable for the amount of activated MAPK at time t, drawn as a pair of dotted lines. Since, the standard deviation of a random variable X equals the square root of its variance which equals $\mathbf{E}(X^2) - \mathbf{E}(X)^2$, the standard deviation (and variance) is calculated by additionally computing the expected value at time t for the reward structure:

```
rewards "activated_squared"
    true : kpp * kpp;
endrewards
```

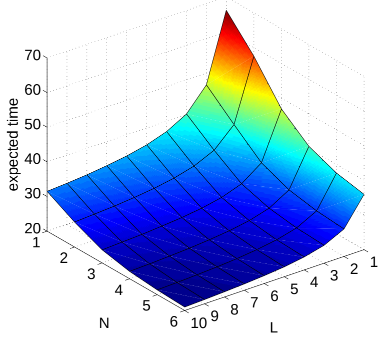
i.e. the square of the reward structure "*activated*" given in Fig. 6.

For the purposes of comparison, we also show results for the expected amount of activated MAPK computed using PRISM's discrete-event simulation engine. These results are presented in Fig. 8 (for the same initial configurations as those used in Fig. 7). These are generated using very small numbers of simulation runs (10 and 100). Smoother approximations for the plots from Fig. 7 can be obtained with higher numbers of runs.

Since it is also easy to change the initial amount N of MAPK, MAPKK and MAPKKK in our model, we also show how the expected amount of activated MAPK over time varies for different values of N. Figure 9(a) shows the



(a) MAPK–MAPKK reactions



(b) Time until MAPK activated

Fig. 10. Expected MAPK–MAPK reactions by t and time until all MAPK activated

expected percentage of of activated MAPK at time t for values of N from 2 up to 8, and Fig. 9(b) the standard deviation for the amount of MAPK over the same parameters.

Using the other reward structures from Fig. 6, we also presents results for the expected number of reactions between MAPK and MAPKK up until time t (Fig. 10(a)) and the expected time until all MAPKs are activated at the same time (Fig. 10(b)). In both cases, we vary the initial amount N of MAPK, MAPKK and MAPKKK and, in Fig. 10(b), we also vary the initial quantity (denoted L) of the enzyme E1.

The results demonstrate that, as N grows, the percentage of MAPK that is eventually activated increases and the time until all MAPKs are activated decreases. They also show the (expected) dynamics that raising species quantities increases the number of reactions that occur between them. We also observe that, as N increases, the behaviour of the PRISM model demonstrates the same behaviour as that presented in [15] (computed through ODEs and the reactions given in Fig. 3) where, in response to an external stimulus (E1), the cascade acts as a switch for the activation of MAPK.

4 Related Work

In this section, we briefly review some other applications of probabilistic verification techniques to systems biology. We also describe the connections that exist between these approaches and the PRISM tool. Figure 11 illustrates the ways in which PRISM can interact with other tools and specification formalisms.

PRISM has been applied to a variety of biological case studies. In [11], it is used to study a model of the FGF (Fibroblast Growth Factor) signalling

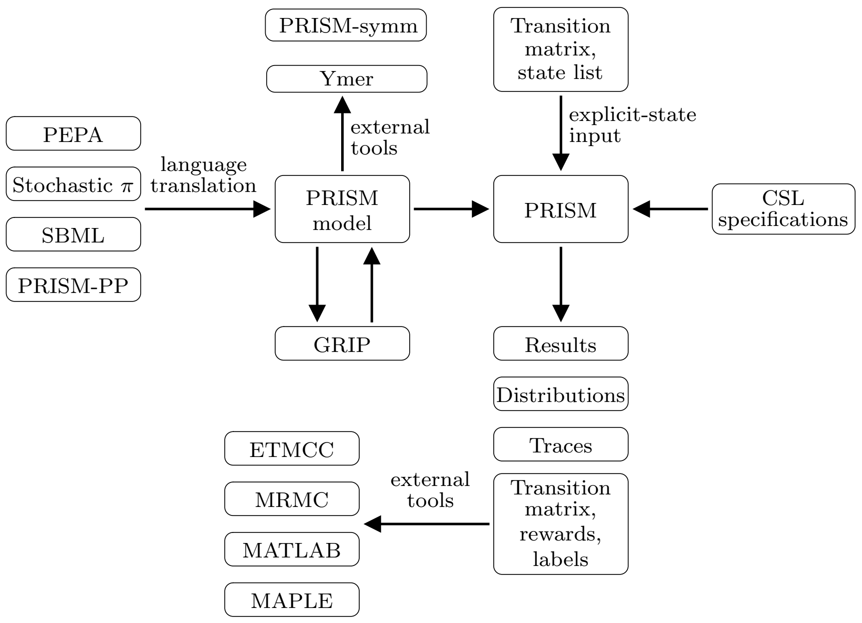


Fig. 11. Language and tool connections for PRISM

pathway. The model corresponds to a single instance of the pathway, i.e. there is at most one of each molecule or species, which has the advantage that the resulting state space is relatively small. However, the model is still highly complex due to the large number of different interactions that can occur in the pathway and is sufficiently rich to explain the roles of each component and how they interact. In [4], PRISM is used to model the RKIP-inhibited ERK pathway where concentrations of each protein are modelled as discrete abstract quantities. Through comparisons with simulations for a traditional differential equation model, the authors show that accurate results can be obtained with relatively small sets of discrete values. PRISM is used in [30] to model codon bias, studying a range of quantitative properties of the system. Finally, [31] uses PRISM, in combination with several other tools, to analyse gene expression modelled using P-Systems.

Another formalism that has proved popular for modelling biological systems is *stochastic process algebra*. For example, PEPA [13] is used in [3] to study the effect of RKIP on the ERK signalling pathway. The stochastic π-calculus [28], an extension of the π-calculus with CTMC semantics, has been used to model many systems, see for example [29, 21]. Various tools for construction and verification of PEPA models are available and, for the stochastic π-calculus, simulators such as BioSpi [29] and SPiM (the Stochastic Pi-Machine) [26] have been developed, but no model checkers. Both formalisms can also be used in conjunction with PRISM, through language translators.

```
@model:2.3.1=MAPK "MAPK"
@compartments
 cell=1
@species
 cell:e1=1 "Enzyme E1"
 cell:kkk=3 "MAPKKK"
 cell:kkk_e1=0 "MAPKKK:E1"
 cell:kkkp=0 "MAPKKK*"
 ...
@parameters
 a1=0.3333333333333333
 d1=150
 k1=150
 ...
@reactions
@r=r1a "MAPKKK is activated through enzyme E1 - 1a"
 kkk+e1 -> kkk_e1
 a1*kkk*e1
@r=r1d "MAPKKK is activated through enzyme E1 - 1d"
 kkk_e1 -> kkk+e1
 d1*kkk_e1
@r=r1k "MAPKKK is activated through enzyme E1 - 1k"
 kkk_e1-> kkkp+e1
 k1*kkk_e1
...
```

Fig. 12. Fragment of SBML-shorthand code for the MAPK cascade of Fig. 3

The PEPA translator is part of PRISM [27] and a prototype stochastic π-calculus translator has been built based on the techniques in [24].

An alternative format for representing biological models is SBML (Systems Biology Markup Language) [34], a computer-readable language based on XML. This is intended to facilitate exchanging models between different systems biology software tools. Biochemical reaction networks are described by specifying the set of species in the system, the reactions they can undergo, and the kinetic laws and parameters which govern these reactions. Again, support for PRISM is provided through a language translator [33]. For illustration, Fig. 12 shows a fragment of the "SBML-shorthand" [35] code which describes the set of MAPK reactions used throughout this chapter. This simple textual language can be automatically translated [35] into SBML. When the SBML model produced is then converted into PRISM code [33], the resulting CTMC is identical to the one used in this chapter.

Further mechanisms are also available for input of models into PRISM. The tool includes a simple pre-processing language (PRISM-PP) which can be used to automatically generate model and property specifications that contain a lot of repetition. Markov chains can also be imported directly (through an explicit list of their states, transitions and rates) allowing models to be generated in other tools and then analysed in PRISM.

Conversely, it is also possible to use external tools to analyse PRISM models. One example is the statistical based model-checker Ymer [37], which performs approximate CSL model checking of CTMCs expressed as PRISM models, using discrete-event simulation and sequential acceptance sampling (for a detailed comparison of the merits of this approach and the probabilistic model checking techniques used by PRISM, see [38]). Another example is the tool GRIP (Generic Representatives In PRISM) [7], which performs language-level symmetry reduction of PRISM models based on the generic representatives approach of [8]. Further support for symmetry reduction is provided by PRISM-symm [19], a prototype extension of PRISM which uses an efficient symbolic (MTBDD-based) implementation.

Finally, models that have been specified in the PRISM modelling language can be constructed in PRISM, and then exported to an explicit representation of the Markov chain for analysis in other tools. In particular, this output can be customised for the probabilistic model checkers MRMC (Markov Reward Model Checker) [16] and ETMCC (the Erlangen-Twente Markov Chain Checker) [12] which can both be used for verifying CTMCs against CSL specifications. MRMC also supports rewards-based property specifications through the logic CSRL [6]. Models, in addition to other PRISM outputs such as numerical results or simulation traces, can be imported into more general-purpose tools such as MATLAB [23] and MAPLE [22].

5 Conclusions

We have illustrated how probabilistic model checking and, in particular, the probabilistic model checker PRISM can be employed as a framework for the analysis of biological pathways. One of the key strengths of this approach is that it allows for the computation of exact quantitative measures relating to the evolution of the system over time. Since, as we have demonstrated, it is possible to specify and verify a wide variety of such measures, a detailed, quantitative analysis of the interactions between the components of a pathway is possible.

The principal challenge remaining for the application of probabilistic model checking to biological systems, as in so many other domains, is the scalability of the techniques to ever larger systems and models. There is hope that some of the techniques that have already been developed in the field of formal verification, such as symmetry reduction, bisimulation minimisation and abstraction, will prove beneficial in this area. For further details on such approaches and pointers to related work, see for example [11].

Acknowledgement. This work was supported in part by EPSRC grants EP/D07956X and EP/D076625.

References

1. A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model checking continuous time Markov chains. *ACM Trans. Computational Logic*, 1(1):162–170, 2000.
2. C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. Software Engineering*, 29(6):524–541, 2003.
3. M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. *Trans. Computational Systems Biology*, 7:1–23, 2006.
4. M. Calder, V. Vyshemirsky, D. Gilbert, and R. Orton. Analysis of signalling pathways using continuous time Markov chains. *Trans. Computational Systems Biology*, 4:44–67, 2006.
5. E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logics. *ACM Trans. Programming Languages and Systems*, 8(2):244–263, 1986.
6. L. Cloth, J.-P. Katoen, M. Khattri, and R. Pulungan. Model checking Markov reward models with impulse rewards. In *Proc. Int. Conf. Dependable Systems and Networks (DSN'05)*, pages 722–731. IEEE Computer Society Press, 2005.
7. A. Donaldson, A. Miller, and D. Parker. GRIP: Generic representatives in PRISM. In *Proc. 4th Int. Conf. Quantitative Evaluation of Systems (QEST'07)*, pages 115–116. IEEE Computer Society, 2007.
8. E. Emerson and T. Wahl. On combining symmetry reduction and symbolic representation for efficientc model checking. In D. Geist and E. Tronci, editors, *Proc. 12th Conf. Correct Hardware Design and Verification Methods (CHARME 2003)*, volume 2860 of *Lecture Notes in Computer Science*, pages 216–230. Springer, 2003.
9. D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.
10. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
11. J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 319:239–257, 2008.
12. H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A tool for model checking Markov chains. *Software Tools for Technology Transfer*, 4(2):153–172, 2003.
13. J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.
14. A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, editors, *Proc. 12th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, volume 3920 of *Lecture Notes in Computer Science*, pages 441–444. Springer, 2006.
15. C. Huang and J. Ferrell. Ultrasensitivity in the mitogen-activated protein kinase cascade. *Proc. Natl. Acad. Sci.*, 93:10078–10083, 1996.
16. J.-P. Katoen, M. Khattri, and I. Zapreev. A Markov reward model checker. In *Proc. 2nd Int. Conf. Quantitative Evaluation of Systems (QEST'05)*, pages 243–244. IEEE Computer Society Press, 2005.

17. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *Software Tools for Technology Transfer*, 6(2):128–142, 2004.
18. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic model checking in practice: Case studies with PRISM. *ACM SIGMETRICS Performance Evaluation Review*, 32(4):16–21, 2005.
19. M. Kwiatkowska, G. Norman, and D. Parker. Symmetry reduction for probabilistic model checking. In T. Ball and R. Jones, editors, *Proc. 18th Int. Conf. Computer Aided Verification (CAV'06)*, volume 4114 of *Lecture Notes in Computer Science*, pages 234–248. Springer, 2006.
20. M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In M. Bernardo and J. Hillston, editors, *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, volume 4486 of *Lecture Notes in Computer Science*, pages 220–270. Springer, 2007.
21. P. Lecca and C. Priami. Cell cycle control in eukaryotes: A BioSpi model. In *Proc. Workshop Concurrent Models in Molecular Biology (BioConcur'03)*, Electronic Notes in Theoretical Computer Science, 2003.
22. MAPLE. `www.maplesoft.com/products/maple/`.
23. MATLAB. `http://www.mathworks.com/products/matlab/`.
24. G. Norman, C. Palamidessi, D. Parker, and P. Wu. Model checking the probabilistic π-calculus. In *Proc. 4th Int. Conf. Quantitative Evaluation of Systems (QEST'07)*, pages 169–178. IEEE Computer Society, 2007.
25. D. Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham, 2002.
26. A. Phillips and L. Cardelli. Efficient, correct simulation of biological processes in the stochastic π-calculus. In M. Calder and S. Gilmore, editors, *Proc. 5th Int. Conf. Computational Methods in Systems Biology (CMSB'07)*, volume 4695 of *Lecture Notes in Bioinformatics*, pages 184–199. Springer Verlag, 2007.
27. PRISM web site. `http://www.prismmodelchecker.org/`.
28. C. Priami. Stochastic π-calculus. *The Computer Journal*, 38(7):578–589, 1995.
29. 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.
30. T. Pronk, E. de Vink, D. Bosnacki, and T. Breit. Stochastic modeling of Codon bias with PRISM. In I. Linden and C. Talcott, editors, *Proc. 3rd Int. Workshop Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems (MTCoord 2007)*. Computer Science Department, University of Cyprus, Nicosia, 2007.
31. F. Romero-Campero, M. Gheorghe, L. Bianco, D. Pescini, M. Prez-Jimnez, and R. Ceterchi. Towards probabilistic model checking on P Systems using PRISM. In H. Hoogeboom, G. Paun, G. Rozenberg, and A. Salomaa, editors, *Proc. 7th Int. Workshop Membrane Computing (WMC06)*, volume 4361 of *Lecture Notes in Computer Science*, pages 477–495. Springer, 2006.
32. J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, volume 23 of *CRM Monograph Series*. AMS, 2004.
33. SBML-to-PRISM translator. `http://www.prismmodelchecker.org/sbml/`.
34. SBML web site. `http://sbml.org/`.

35. SBML-shorthand. `http://www.staff.ncl.ac.uk/d.j.wilkinson/software/sbml-sh/`.
36. O. Wolkenhauer, M. Ullah, W. Kolch, and K.-H. Cho. Modeling and simulation of intracellular dynamics: choosing an appropriate framework. *IEEE Trans. Nanobioscience*, 3:200–207, 2004.
37. H. Younes. Ymer: A statistical model checker. In K. Etessami and S. Rajamani, editors, *Proc. 17th Int. Conf. Computer Aided Verification (CAV'05)*, volume 3576 of *Lecture Notes in Computer Science*, pages 429–433. Springer, 2005.
38. H. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *Software Tools for Technology Transfer*, 8(3):216–228, 2006.