Using Probabilistic Model Checking in Systems Biology

Marta Kwiatkowska, Gethin Norman and David Parker
Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD

ABSTRACT
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 paper 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, where 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 paper we show how probabilistic model checking [2, 20, 30] and the probabilistic model checker PRISM [14, 25] 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 [11, 4, 28] and previous work which has demonstrated the applicability of probabilistic model checking to the analysis of a wide variety of complex stochastic systems [18].

This framework inherits many of the advantages of model checking, including the use of 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, see for example [17]. 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 of the paper. 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 Section 4 we discuss related work and Section 5 concludes the paper.

2. PROBABILISTIC MODEL CHECKING
Probabilistic model checking is a formal verification technique for the modelling and analysis of systems which 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, let \( \mathbb{R}_{\geq 0} \) 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 $(S, R, L)$ where:

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

The transition rate matrix $R$ 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 $R(s, s') > 0$ and, in this case, the probability of the transition being triggered within $t$ time-units equals $1 - e^{-R(s, s') t}$. Typically, in a state $s$, there is more than one state $s'$ for which $R(s, s') > 0$; 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 $E(s) = \sum_{s' \in S} R(s, s')$, called the exit rate. The probability of moving to state $s'$ is given by $R(s, s') / E(s)$.

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 $(\rho, \iota)$ where:

- $\rho: S \rightarrow \mathbb{R}_{\geq 0}$ is a state reward function;
- $\iota: (S \times S) \rightarrow \mathbb{R}_{\geq 0}$ 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 logic model checking, i.e. calculation of probabilities and rewards. 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, 23]. These allow for compact representation and efficient manipulation of large, structured models by exploring 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 to perform manual execution and debugging of probabilistic models. Other functionality provided by the user interface of the tool includes a graph-plotted 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 MAP Kinase Kinase (MAPKKK), a MAP Kinase (MAPKK) and a MAPK. The cascade is initialised through the phosphorylation of MAPKK, which then activates MAPKK, through phosphorylation at two serine residues. This then activates MAPK through phosphorylation at threonine and tyrosine residues. The initialisation of the pathway can be caused by a diverse set of stimuli including growth factors, neurotransmitters and cytokines.

Figure 1 gives an overview of the structure of the pathway and Figure 2 details the reactions that form the cascade, as taken from [15]. In the reactions presented in Figure 2, 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 by the corresponding phosphatase, while the activation and deactivation of MAPKKK is through the enzymes E1 and E2 respectively. To simplify the presentation in Figure 2 we denote MAPK, MAPKK and MAPKKK by K, KK and KKK respectively.

The kinetic rates given in Figure 2 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 Figure 2 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 $Vol \cdot N_A$ where $Vol$ is the volume and $N_A$ is Avogadro’s number. For a more detailed discussion of the relationship between kinetic and stochastic rates, see for example [33, 9].

A model described in the PRISM language comprises a set of modules, the state of each being represented by a set of finite-ranging variables. The global state of the model is determined by the union of all variables, which we denote $V$. 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:

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

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

$$ (x_1' = u_1) \& (x_2' = u_2) \& \ldots \& (x_k' = u_k)$$

where $u_1, u_2, \ldots, u_k$ are functions over $V$ and $x_1, x_2, \ldots, x_k$ are variables of the module. Intuitively, in global state $s$ of the PRISM model, the command is enabled if $s$ satisfies the predicate $\text{guard}$. If a command is enabled, a transition that updates the module’s variables according to $\text{update}$ can occur with rate $\text{rate}$. 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 ex-
press 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 Figure 3, we present the module representing quantities of species relating to MAPK and, in Figure 4, the module representing MAPK phosphatase. Note that, 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 MAPKK, 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, 25]. The complete PRISM description of the MAPK cascade is available from the case study repository on the PRISM website [25].

As can be seen in Figures 3 and 4, 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 Figure 3). 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 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 also vary these during verification.

Figures 3 and 4 also show that the modules for MAPK and MAPK phosphatase synchronise through the actions a_k_phosphate, d_k_phosphate and k_k_phosphate, which correspond to the deactivation of MAPK (as described in reactions 8 and 10 of Figure 2). The actions a_k_phosphate, d_k_phosphate and k_k_phosphate, which appear in the module for MAP (Figure 3), correspond to the activation of MAPK by MAPKK-PP (see reactions 7 and 9 of Figure 2), and there will therefore be 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_pp1, k_pp2 and k_pp3) and two MAPK phosphatases (kptase1 and kptase2) then there are six different species that can be formed: k_pp1; kptase1, k_pp2; kptase2, k_pp2; kptase1, k_pp1; kptase2, k_pp1; kptase1 and k_pp3; kptase2. 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 * kpp and kptase in the modules MAPK (Figure 3) and KPTASE (Figure 4), respectively.
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 all 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 Figure 5, we present three different reward structures for the PRISM model of the cascade. The first reward structure (“activated”) assigns a state reward equal to the percentage of MAPK that is activated. This can be used to compute the expected percentage of activated MAPK at some time instant or in the long run. The second 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 third 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 uses 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.

- \((\text{kkpp}=N \land \text{kkpp}=0) \rightarrow \mathcal{P}_{\geq 0.12}[(\text{kkpp}>0) U (\text{kkpp}>0)]\) - if all MAPKKs are activated and none of the MAPKs are activated, then the probability that, while some MAPKK remain activated, some of the MAPKs become activated is at least 0.12.
- \(\mathcal{P}_{=0}[(\text{kkpp}+\text{kkpp})<l)] \) - what is the probability that the total number of MAPKs and MAPKKs activated at time instant \(t\) is less than \(l\)?
- \((\text{kkkp}>0 \land \text{kkpp}=0) \rightarrow \mathcal{P}_{0.01}[(\text{kkpp}=0) U^{(t_1-t_2)} (\text{kkpp}>0)] \) - if some MAPKKKs are activated and no MAPK are activated, then the probability that the first time a MAPK gets activated is within the time interval \([t_1,t_2]\) is greater than 0.7.
- \((k=0) \rightarrow \mathcal{P}_{0.01}[(k=0) U^{t<\infty} (k>0)] \) - if there are no inactive MAPKs, then the probability that the first time a MAPK becomes inactive is after time \(t\) is at most 0.01.
- \(\mathcal{S}_{=1}[(\text{kkpp}=l)] \) - what is the probability that in the long run there are precisely \(l\) MAPKs activated?
- \(\mathcal{R}_{\{\text{reactions}\}}=\mathcal{C}^{\leq t} \) - what is the expected number of reactions between MAPKs and MAPKKs during the first \(t\) seconds?
- \((\text{kkpp}=N) \rightarrow \mathcal{R}_{\{\text{activated}\}} \geq 75\% \mathcal{L}^{\leq t} \) - if all MAPKs are activated, then after \(t\) seconds the expected percentage of activated MAPK is at least 75%.
- \(\mathcal{R}_{\{\text{reactions}\}}=\mathcal{F} (\text{kkpp}=N) \) - what is the expected number of reactions between MAPK and MAPKK until all MAPKs are activated at the same time instant?
- \((\text{kkpp}>0) \rightarrow \mathcal{R}_{\{\text{time}\}} \leq 120\% \mathcal{F} (k=N) \) - if some MAPKs are activated, the expected time until all of the MAPKs are inactive at the same time instant is less than 120 seconds.
- \(\mathcal{R}_{\{\text{activated}\}} \geq 98\% \mathcal{S} \) - the long run percentage of activated MAPKs is at least 98%.
- \(\mathcal{R}_{\{\text{reactions}\}} \leq \mathcal{S} \) - what is the long run average number 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 in these results as some parameters of the model (e.g. initial species concentrations or reaction rates) or of the property specification (e.g. a time bound) vary. 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, Figure 6 shows results obtained with PRISM for the MAPK cascade case study. The properties we consider are the expected percentage of activated MAPK at time instant $t$, the expected number of reactions between MAPK and MAPKK up until time $t$ and the expected time until all MAPK are activated at the same time. We varied the initial quantities of MAPK, MAPKK and MAPKKK: initially we suppose there are $N$ of each of these species and that there is 1 of all remaining species in the cascade (the enzymes E1 and E2 and the phosphatases for MAPK and MAPKK). For the first two properties, we also varied the value of the time bound $t$.

The results presented in Figure 6 demonstrate that, as the size of $N$ grows, the percentage of MAPK that is activated increases and the time until all MAPK 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 Figure 2) 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 7 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 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 [28] to model codon bias, studying a range of quantitative properties of the system. Finally, [29] 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 [3] to study the effect of RKIP on the ERK signalling pathway. The stochastic $\pi$-calculus [26], an extension of the $\pi$-calculus with CTMC semantics, has been used to model many systems, see for example [37, 7]. Various tools for construction and verification of PEPA models are available and, for the stochastic $\pi$-calculus, simulators such as BioSpi [27] and SPiM (the Stochastic Pi-Machine) [24] have been developed, but no model checkers. Both formalisms can also be used in conjunction with PRISM, through language translators. The PEPA translator is part of PRISM [25] and a prototype stochastic $\pi$-calculus translator has been built based on the techniques in [22].

An alternative format for representing biological models is SBML (Systems Biology Markup Language) [32], 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 transator [31].

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
Figure 7: Language and tool connections for PRISM

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 [34], 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 [35]). Another example is the tool GRIP (Generic Representatives In PRISM) [7], which performs language-level symmetry reduction of PRISM models based on the 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 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 and MAPLE.

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].

Acknowledgment

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

6. REFERENCES


