A Monte Carlo Model Checker for Probabilistic LTL with Numerical Constraints

Robin Donaldson, David Gilbert
Bioinformatics Research Centre, University of Glasgow
Glasgow G12 8QQ, Scotland, UK
radonald@brc.dcs.gla.ac.uk, drg@brc.dcs.gla.ac.uk

Abstract

We define the syntax and semantics of a new temporal logic called probabilistic LTL with numerical constraints (PLTLc). We introduce an efficient model checker for PLTLc properties. The efficiency of the model checker is through approximation using Monte Carlo sampling of finite paths through the model's state space (simulation outputs) and parallel model checking of the paths. Our model checking method can be applied to any model producing quantitative output – continuous or stochastic, including those with complex dynamics and those with an infinite state space. Furthermore, our offline approach allows the analysis of observed (real-life) behaviour traces. We find in this paper that PLTLc properties with constraints over free variables can replace full model checking experiments, resulting in a significant gain in efficiency. This overcomes one disadvantage of model checking experiments which is that the complexity depends on system granularity and number of variables, and quickly becomes infeasible. We focus on models of biochemical networks, and specifically in this paper on intracellular signalling pathways; however our method can be applied to a wide range of biological as well as technical systems and their models. Our work contributes to the emerging field of synthetic biology by proposing a rigourous approach for the structured formal engineering of biological systems.

1. Motivation

The greatest challenge in modern bioscience is arguably the development of techniques for the engineering of living systems in a rigourous manner. This is the domain of the emerging discipline of "Synthetic Biology" [16], which can be defined as the design and construction of new biological parts, devices, and systems, as well as the re-design of existing natural biological systems for useful purposes [23]. One aspect of Synthetic Biology which distinguishes it from conventional genetic engineering is a heavy emphasis on the development of foundational technologies that make the engineering of biology easier and more reliable.

Therefore, a crucial enabling technology is that of modelling, both in system design and verification of the constructed system. An important early stage in the design process is confirming that the behaviour of the model conforms to the desired behaviour, i.e. "model checking". Qualitative and especially quantitative model checking has been proven to be useful in systems biology to validate models derived from the observed real-life behaviour, see e.g. [5], [6], [10], [15], and is likely to be of equal importance for synthetic biology. Thus, when a system has been constructed, the actual behaviour should be checked for conformance with the desired behaviour. A characteristic of biological systems is their inherently stochastic behaviour when considered at low numbers of molecules or cells, and thus it is highly desirable that models and model checking techniques can deal with continuous-time stochastic behaviour.

In this paper we present work for the development of rigourous approaches for the structured formal engineering of biological systems. Driven by the limitations imposed by the computational effort of exact probabilistic model checking, for example as encountered in applying CSL/PRISM [21], and inspired by approximate model checking [17], [24], we extend probabilistic LTL [3] with numerical constraints to define PLTLc and employ Monte Carlo approximation for PLTLc properties. Monte Carlo approximation samples a finite set of finite paths through the model's state space (simulation outputs)

and decides the probability of properties based on this set. We introduce a parallelised model checker called the Monte Carlo Model Checker for PLTLc properties MC2(PLTLc).

Our approach can be applied to both stochastic models, for example stochastic Petri nets, and deterministic models, constructed using for example continuous Petri nets or ordinary differential equations. Notably the Monte Carlo and offline approach of MC2(PLTLc) can handle state spaces beyond the current limits of exact analyses, and also for systems with complex dynamics as semi-Markov processes or generalized semi-Markov processes as well as for systems with an infinite state space. Our method can also be applied to validate the behaviour of a system which has been constructed according to a model, through model checking real-life behaviour traces. We focus on models of biochemical networks, and specifically in this paper on intracellular signalling pathways; however our method can be applied to a wide range of biological as well as technical systems and their models.

PLTL with numerical constraints is a novel extension of constraints over free variables previously applied in model checking of continuous behaviour, for example [9]. To apply numerical constraints in a probabilistic setting, a probabilistic domain is computed for each free variable in the property. We describe two examples where constraints can replace an experiment comprising many properties. Thus a single PLTLc property can replace a set of properties, resulting in a significant improvement in efficiency. This improvement is especially significant when a property contains several variables, as a single PLTLc property replaces a set of properties of size $O(L^n)$, where n is the number of variables and L the number of levels.

This paper is organised as follows. The next section defines and details the theory behind our proposed logic. Section 3 deals with the implementation of our model checker tool and discusses the choices used in our analysis. Section 4 gives examples of our analysis approach, relating to the biochemical context. We conclude in the last two sections with a comparison to related work and a summary of the main accomplishments of this work.

2. PLTLc

Syntax Linear-time Temporal Logic (LTL) [22] is the fragment of full Computational Tree Logic (CTL*) [8] without path quantifiers, implicitly quantifying universally over all paths. LTL has been introduced in a probabilistic setting in [3], and extended by numerical constraints over real value variables in [9]. PLTLc combines both extensions, complemented by the filter construct as used in Probabilistic Computational Tree Logic (PCTL) [14] and Continuous Stochastic Logic (CSL) [1]. We start with the LTL with numerical constraints (LTLc) syntax:

$\phi ::= X\phi|G\phi|F\phi|\phi U\phi|\phi R\phi|\phi\vee\phi|\phi\wedge\phi|\neg\phi|\phi\rightarrow\phi|value = value|value \neq value|value > value|value \geq value|$
$value < value \mid value \leq value \mid true \mid false$

Numerical constraints over free variables are defined in this logic through the inclusion of free variables denoted by $*fVariable* in the definition of $value$ below – the symbol $ differentiates a free variable from a regular variable. Regular variables are read-only values which form the behaviour of the model, whereas free variables are instantiated during the model checking process to the values for which the temporal logic property holds. In our current implementation free variables are defined to have integer domains initialised to $[0 \rightarrow \infty)$ and describe protein concentrations, numbers of molecules and time. Constraints over free variables, which involve equality/inequality and relational operators, restrict the domain of the free variable, such that with $\$X \in [0 \rightarrow \infty)$, $\$X > 5$ sets $\$X$ to be $[6 \rightarrow \infty)$. If there is a constraint over free variables involving real numbers, then the real numbers are cast to integers. Notice also that disjunction, conjunction, negation and implication of constraints over free variables are allowed. Finally, the values considered in this logic are integers and real numbers, and the four basic arithmetic operations over these values:

$value ::= value + value \mid value - value \mid value * value \mid value/value \mid$ *$fVariable* | *Variable* | $function$ | *Int* | *Real*

where *Int* is any integer number and *Real* is any real number. In our biochemical pathway analysis we define *Variable* to be the time dependant value of the concentration of any biochemical species in the model, either integers for molecules/levels or real numbers for concentrations, and we define a special variable called *time* to stand for the values of state time. State time values are the simulation time points such that we can, for example, express properties relative to simulation time. This is especially useful for expressing a property before or after some event, such as introducing a drug into a cell. We provide the ability to define any *function* returning a real or integer value, and in our current system we have chosen to implement the two functions, *max(variable)* and *d(variable)*. The function *max* operates over all the values of a species to return the maximum of the species' value in the simulation run, thus the peak of a species can be expressed; $Protein = max(Protein)$. We also define a function *d* which returns the derivative of the concentration of the species at each time point, thus increasing and decreasing species value can be expressed; $d(Protein) > 0$ and $d(Protein) < 0$ respectively.

PLTLc enhances LTLc by the inclusion of a probability operator and filter construct, and the probabilistic interpretation of the domains for the free variables. The top-level definition of PLTLc is:

$\psi ::= \mathbf{P}_{\trianglelefteq x}[\,\phi\,] \mid \mathbf{P}_{\trianglelefteq x}[\,\phi\{SP\}\,]$

where ϕ is an LTLc expression. SP is a State Proposition containing an AP, or any boolean combination of APs using $\{\vee, \wedge, \neg, \rightarrow\}$, containing *no* free variables without a loss of expressivity. Note that the square and curly brackets are part of PLTLc. Given that $\trianglelefteq \in \{>, \geq, <, \leq\}$, $P_{\trianglelefteq x}$ is any inequality comparison of the probability of the property holding true, for example $P_{\geq 0.5}$. We also permit the expression $P_{=?}$ returning the value of the probability of the property holding true. We disallow equality testing of the probability, $P_{=x}$ because of the representation of real values and the semantics of their equality.

We define filters similar to those used in PCTL and CSL. This permits specifications to refer to the state or states that the property is checked from, rather than default to the initial state. Hence, for a property of the form $\phi\{SP\}$, ϕ is checked from the first state that SP is satisfied. In PLTLc this is equivalent to the formula $(\neg SP)U(SP \wedge \phi)$, however the filter abbreviation is introduced to increase readability and compatibility with branching-time logics.

Semantics The semantics of PLTLc is defined over a finite set of finite paths through the system's state space – in our case, stochastic or deterministic simulations, or time series data recorded in wet lab experiments.

First, let a path π be a finite sequence of states describing the behaviour of a biochemical system, $\pi = s_0, s_1, ..., s_n$ $(n < \infty)$ and π^i be the subsequence of π starting from state s_i, $i \leq n$, thus $\pi^i = s_i, s_{i+1}, ..., s_n$. The semantics of checking ϕ or $\phi\{SP\}$ as appropriate is described in the standard way in Table 1. Each path in the set of paths can be evaluated to a boolean value as to whether ϕ or $\phi\{SP\}$ holds in the path. When all paths are evaluated, the number of true values in the set over the size of the set yields the overall probability of the PLTLc property. Hence for a stochastic model, where the set of paths is typically > 1, the probability is in the range $[0 \rightarrow 1]$ and calculated through Monte Carlo approximation, whereas a continuous model has a probability of either 0 or 1.

$\pi \models X\phi$	iff $\pi^1 \models \phi$
$\pi \models G\phi$	iff for all $k \geq 0$, $\pi^k \models \phi$
$\pi \models F\phi$	iff there exists a $k \geq 0$, $\pi^k \models \phi$
$\pi \models \phi^1 U \phi^2$	iff there exists a $k \geq 0$, $\pi^k \models \phi^2$ and for all j, $0 \leq j < k$, $\pi^j \models \phi^1$
$\pi \models \phi^1 R \phi^2$	iff for all $k \geq 0$, $\pi^k \models \phi^2$ or there exists a j, such that for all k, $0 \leq k < j$, $\pi^k \models \phi^2$ and $\pi^j \models \phi^1$
$\pi \models \phi^1 \vee \phi^2$	iff $\pi \models \phi^1$ or $\pi \models \phi^2$
$\pi \models \phi^1 \wedge \phi^2$	iff $\pi \models \phi^1$ and $\pi \models \phi^2$
$\pi \models \neg\phi$	iff $\pi \not\models \phi$
$\pi \models \phi^1 \rightarrow \phi^2$	iff $\pi \models \phi^2$ or $\pi \not\models \phi^1$
$\pi \models \phi\{SP\}$	iff the smallest k such that $\pi^k \models SP$, also $\pi^k \models \phi$
$\pi \models AP$	iff $s_0 \models AP$
$s \models AP$	iff s satisfies the atomic proposition, AP.

Table 1. The standard semantics of checking ϕ or $\phi\{SP\}$ against a finite path.

Finally, the two PLTLc functions we have chosen to implement, *max(variable)* and *d(variable)* are defined as follows. *max(variable)* calculates the first state s_{max} in the finite path π for which the value of *variable* is maximal and returns this value. *d(variable)* calculates for each state s_i in the finite path π the derivative of the value of *variable* between state s_i and s_{i+1}. In the case of the final state in the finite path s_n which contains no next state, the derivative is equal to the derivative of the previous state s_{n-1}.

The choice of simulator and simulation parameters used to compute the finite path can affect the semantics of the PLTLc property and the correctness of the result. For example the Next operator refers to the next time step in the simulator, however in a fixed time-step ODE solver, adaptive time-step ODE solver and exact Gillespie simulator these refer to a fixed, varying and random amount of time in the future respectively. We demonstrate this problem in an example in Section 4.2.

Probabilistic Domains Each path in the set of paths is also evaluated to a domain of validity, $D_{\phi \; or \; \phi\{SP\}} \subset \mathbb{N}^n$ for n free variables in the PLTLc property, $\$fVar_1, \$fVar_2, ...\$fVar_n$. The domain of validity is defined such that for all valuations v of the n free variables, where $v \in D_{\phi \; or \; \phi\{SP\}}$, the property ϕ or $\phi\{SP\}$ as appropriate holds true for the path. Thus each path has an associated domain of validity, with paths resulting in a boolean value of true having a non-empty domain of validity, i.e. for these paths there must be valuations of the variables for which the property holds.

After the set of domains of validity is evaluated from the set of paths, a probabilistic domain for each of the n free variables in the PLTLc property is calculated. A probabilistic domain associates with each integer value in the domain the

probability of the property holding true for that value. If the PLTLc property evaluates to a probability p, then the maximum probability of any value in the probabilistic domains is p, such that a property with 0 probability has probabilistic domains with 0 probability for all values. The probabilistic domain of free variable $\$fVar_i$ is calculated by iterating through each integer value in the domain I. A count is performed on the set of domains of validity for the number of domains which contain at least one valuation v with $v(\$fVar_i) = I$. This number over the size of the set is the probability of the value I in the probabilistic domain of $\$fVar_i$.

In the case that the system is described by a stochastic model, the probabilistic domains are calculated through Monte Carlo approximation – the number of occurrences of a value for a free variable in each domain of validity in the set over the size of the set. In the case of a continuous model where the size of the set is 1, the probabilistic domain contains probabilities 0 and 1 and can equally be represented by a probabilistic domain or a regular domain.

Monotone Properties Properties expressed in a logic can be categorised as monotone if they satisfy the following condition; if the property is satisfied in any path through the state space, then it is satisfied in any extension of the path [17]. A subset of PLTLc called Essentially Positive Fragment (EPF) expresses only monotone properties, consisting of PLTLc without the Globally operator and disallowing any negation of path formulas.

Our approach to model checking incorporates two approximations. We approximate the truth value of a path by operating over a finite sequence of states. We also approximate the probability of the property through sampling a finite number of paths (a subset of the model's behaviour). When assessing a monotone property against a set of paths, the probability value can only increase if those paths were extended in time. Hence ignoring the Monte Carlo (sampling) approximation, monotone properties permit an estimate of the lower bound in the time approximation made. We find that monotone properties are useful in our biochemical pathway analysis and in fact all the properties considered in our results section are monotone.

Relation to Branching-Time Logics PLTLc differs in several ways from probabilistic branching-time logics such as PCTL and CSL which operate on discrete-time and continuous-time Markov chains (DTMC, CTMC) respectively. PLTLc is a linear-time logic and operates in-turn on paths through the state space, thus it is not possible to compute probabilities at any desired state. This means we cannot have probability operators embedded within the expression. In our approach to PLTLc checking, the Markov chain is never constructed which helps for efficiency (and even feasibility for coping with infinite state spaces).

Furthermore, PCTL and CSL define a filter construct which changes the initial state from which the property is checked. PRISM's implementation of CSL's filters chooses the first state lexicographically from the set of states the filter satisfies. This is an arbitrary decision by PRISM, and the user is notified if there is more than one state satisfying the filter. PLTLc also defines filters, however in a linear-time setting, and chooses the first state chronologically (in simulation time) in each path which satisfies the filter. The filter state which is chosen to check the property from may be different between paths. Thus, with no extra effort, we calculate the probability of the property from many initial filter states weighted by the probability of their occurrence (through Monte Carlo).

Finally, PCTL and CSL can check for steady state behaviour using exact analysis. PLTLc approximates steady state behaviour analysis through model checking long paths and using the time value to check late in the path, however this cannot prove the existence of steady state behaviour.

3. Model Checking

Model Checking Algorithm We have implemented a computational system in Java called Monte Carlo Model Checker for PLTLc properties MC2(PLTLc) to evaluate PLTLc properties. This is an offline model checker, decoupled from the simulator used to generate the paths allowing any quantitative simulator to be connected or even recorded traces from wet lab experiments.

Model checking of a PLTLc property returns two results; the overall probability of the property holding true, and the probabilistic domains of any free variables, each of which describe the probability of the property holding true for any value of the free variable. For the time being, we only implement constraints between a free variables and value in MC2(PLTLc), for example $\$X > \$Y$ would be disallowed.

The *trace-based* $\exists$*-constraint-LTL formula instantiation* algorithm from [9] is applied to every path in the set of paths The PLTLc property is parsed resulting in a parse tree with leaf nodes as the Atomic Propositions and the root node as full property. This algorithm labels each state s_i in the path π with the domain of validity according to the current node in the parse tree, traversed using a depth-first traversal. Starting with the APs, if the AP does not contain any free variables, state s_i is labelled with the domain of validity $D_{AP}\ (s_i) = \mathbb{N}^n$ if the AP is true in that state, otherwise labelled with $D_{AP}\ (s_i) = \emptyset$. If the AP contains a free variable, then the state is labelled by the half-space of $\mathbb{N}$ for which the constraint is true. Next we

explain the temporal operators, for example the Finally operator $F\phi$. Starting at the last state, the state s_i is labelled with the domain of validity $D_{F\phi}(s_i) = D_{F\phi}(s_{i+1}) \cup D_{\phi}(s_i)$. Lastly, we explain the boolean connectives, for example the And operator $\phi_1 \wedge \phi_2$. The state s_i is labelled with the domain of validity $D_{\phi_1 \wedge \phi_2}(s_i) = D_{\phi_1}(s_i) \cap D_{\phi_2}(s_i)$. The full details of this algorithm and the associated strong completeness theorem are described in [9].

If a filter is present in the PLTLc property then trivially the first state s_f which satisfies the filter's State Proposition $\{SP\}$ is found. Each path satisfies the property if the first state s_0, or s_f if a filter is present, has a non-empty domain of validity for the root node of the parse tree, i.e. there are values of the free variables for which the entire PLTLc expression (root node) is true. This domain of validity is added to an overall set of domains of validity, used later to compute the probabilistic domains. After all paths in the set are evaluated, the probability operator calculates the probability based on the fraction of paths which satisfy the property over the size of the set. If the probability operator contains an inequality, then this inequality is evaluated returning a boolean value, or else the probability value is returned.

The probabilistic domain of each free variable in the property is calculated from the set of domains of validity. The probability of each integer value I in a probabilistic domain of a free variable is calculated in turn. The probability is the number of domains of validity in the set which contain a valuation of the free variable with value I over the size of the set. Probabilistic domains contain integer values from $[0 \rightarrow \infty)$ and so the stopping condition is when every subsequent value tending to infinity has a constant probability.

Model Checking of Continuous Behaviour Model checking of deterministic simulations in this paper is performed through an integration of MC2(PLTLc) in the ODE-based BioNessie simulator [4]. A feature of the integration of MC2(PLTLc) in BioNessie is property checking over parameter scans. A parameter in the model definition (kinetic rate or initial species concentration) can take a sequence of values (value range and step-size). ODE simulation is then performed with the parameter set to each value in the sequence. The MC2(PLTLc) integration can be used to check the PLTLc property against each of these simulation outputs, where the overall probability of the property is the fraction of parameter values which result in simulations which satisfy the property over of the number of parameter values. Note that even though the system is deterministic, there is a range of behaviours from the parameter scan such that the overall probability of the property holding true is within in the range $[0, 1]$. This feature can be used to efficiently find a desired behaviour of a model, assess the role of a reaction/species or assess the probabilistic domain of a free variable in a parameter range (e.g. the distribution of peak values of a species when a parameter is varied).

Model Checking of Stochastic Behaviour MC2(PLTLc) can operate on output from any stochastic simulator, either with discrete values such as Gillespie's algorithm [12] or continuous values such as Stochastic Differential Equations (SDEs). In this paper we use an efficient implementation of Gillespie's algorithm called Gillespie2 [11]. We have modified this implementation to become an exact Gillespie simulator and the output has been restricted to only the values of species and molecular events of interest to the property being checked.

However, to handle model checking of a set of large simulation output files, such as a set of Gillespie simulations with many molecules, we carry out simulation and analysis over a cluster of computers, performing model checking of single simulation runs on each node in the cluster.

Concentration Levels Calder *et al.* [5] and later Gilbert *et al.* [10] assess properties in terms of discrete concentration levels as shown in Figure 1. The range of the continuous ODE concentrations is split evenly into N equivalence classes, defining the N+1 levels 0, 1, ... N. In the following characterisation of the granularity we always give the highest level number, hence 4 levels defines levels at 0, 1, ... 4. We have taken a similar approach, however we additionally define a distinction between molecules and levels. Molecules refer to the granularity of the system (the number of tokens in the simulation) and levels refer to the granularity of model checking (the concentration values at which to check the property). Essentially the number of levels relates to the granularity of the analysis, with a higher number of levels checking at smaller increments in the molecular range. Of course, we cannot assess a property at a higher number of levels than number of molecules in simulation. In this paper we have used the same number of levels as the number of molecules with a maximum of 500 levels, producing sufficiently smooth and detailed graphs of the results. It is advisable for the sake of efficiency that the ratio of levels to molecules is reduced for higher system granularities.

4. Results

4.1. Two Reaction Model

As an initial illustrative example and to calibrate our model checker, we have constructed a simple reaction model comprising two mono-molecular reactions of the form:

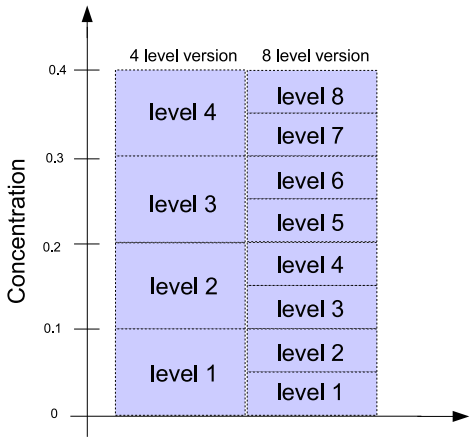

Figure 1. The abstraction from a concentration range of 0 mMol to 0.4 mMol to 4 and 8 concentration levels.

$A \xrightarrow{0.01} B$
$C \xrightarrow{0.1} D$

where the initial concentration of the reactants, A and C are 0.1 $mMol$ and there are initially no products B and D. The values 0.01 and 0.1 above the arrows denote the rates of the reactions, in $mMol \cdot s^{-1}$. In the deterministic world, the rate values are the values of the ODE rates. In (stochastic) Gillespie simulation the rate of a reaction over the sum of all rates in the system is the probability of that molecular event occurring, hence reaction $A \rightarrow D$ occurs with probability $0.01/0.11$. The time interval until this molecular event occurs is sampled from an exponential distribution with lambda equal to the sum of all rates of the enabled reactions.

We assess the probability of the reactant A equalling for the first time the product D at some value represented by the placeholder X, written in PLTLc as:

$P_{=?}[(A = X)\{A = D\}]$

'Experiments' replace the placeholder X with a range of values to produce a set of properties where X is instantiated to each value in the range. Each property in the experiment assesses the probability that $A = D$ for the first time at a different value, such that there is the same number of properties as values in the range. The number of values in the range is equivalent to the number of levels, which we define previously as the detail of the analysis.

We first assess how close the approximate results of MC2(PLTLc) are to exact results provided by analytical approaches, for example PRISM. The deterministic model involving concentrations in $mMol$ is converted to a stochastic model with 10 molecules. The property is assessed in an experiment with X replaced by values $0, 1, ..., 10$. A comparison of PRISM's exact results and MC2(PLTLc)'s approximate results (with a simulation time of 100s, sufficient to capture $A = D$) is shown in Figure 2. From this figure it is clear that considering the given inaccuracies in determining the rate constants, 10,000 simulation runs matches exact results and 1,000 simulation runs provides sufficient approximation. The mean squared errors between the exact results and approximated results for 100, 1,000 and 10,000 simulation runs are 0.0134, 0.0036 and 0.0010 respectively.

Continuing with sufficient approximation using 1,000 simulation runs, the property is assessed with models up to 10,000 molecules. To exploit PLTLc fully, the placeholder X is replaced with a free variable $\$X$:

$P_{=?}[(A = \$X)\{A = D\}]$

The constraint $A = \$X$ now sets the free variable $\$X$ to the value at which $A = D$ for the first time. Considered over the set of 1,000 simulation outputs, the probabilistic domain of $\$X$, evaluated through a single property, contains the same information as an experiment containing, in the worst case in our example, 10,000 properties to be checked. The property has probability 1, stating that in all simulations $A = D$ at some point, and the probabilistic domain of $\$X$ has the values at which $A = D$ for the first time with the associated probability. We observe that as the number of molecules in the model increases, the most probable value predicted in the probabilistic domain of $\$X$ tends towards the intersection of A and D in ODE simulation at around 8.35 $mMol$. This is shown in Figure 3.

Table 2 shows the model checking time in MC2(PLTLc) (approximate) with the constraint and experiment approaches and

the model checking time in PRISM (exact) with the experiment approach. The gain in efficiency when using constraints is clear, and the exact approach quickly becomes infeasible. For systems containing more than one placeholder, free variables result in an even greater improvement of efficiency. The efficiency gain for n variables and L levels is $O(L^n)$. Furthermore, the experiment approach requires *a priori* knowledge of the value range of the variables allowed in the model.

Molecules	Approximate Constraint Approach	Approximate Experiment Approach	Exact Experiment Approach
10	33 seconds	35 seconds	2 seconds
100	37 seconds	54 seconds	20 minutes
1,000	1 minute	8 minutes	N/A
10,000	8.5 minutes	13 hours	N/A

Table 2. A comparison of the model checking times of the approximate constraint, approximate experiment and exact experiment approaches with the two reaction model. In the experiment approach, the same number of levels as molecules is used. Times listed for MC2(PLTLc) are the combination of 1,000 simulations at 100s and model checking, and all times listed are for a current standard workstation. The size of the state space that the exact approach must construct is $(\#molecules + 1)^2$. N/A indicates that the time taken to perform model checking of a single query exceeded 24 hours.

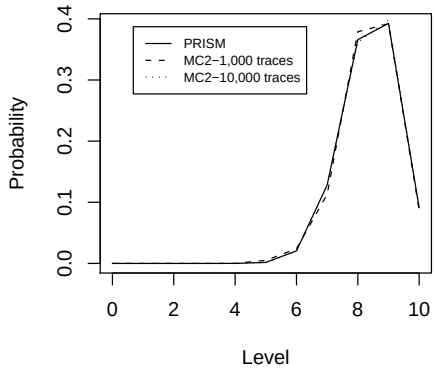

Figure 2. A comparison of exact PRISM results and approximate MC2(PLTLc) results with 1,000 and 10,000 simulation runs. This is the probability of the property as per Figure 3. From this graph it is clear that 10,000 simulation runs approximates the exact results nearly perfectly, while 1,000 simulation runs delivers sufficient accuracy.

4.2. Mitogen Activated Protein Kinase (MAPK) Signalling Pathway

We perform model checking of a model of the mitogen-activated protein kinase (MAPK) cascade published in [20]. This is the core of the ubiquitous ERK/MAPK pathway that can, for example, convey cell division and differentiation signals from the cell membrane to the nucleus. The description of the pathway starts at the RasGTP complex which acts as a kinase to phosphorylate Raf, which phosphorylates MAPK/ERK Kinase (MEK), which in turn phosphorylates Extracellular signal Regulated Kinase (ERK). The response of the cell to the input signal is dependent on the activity of activated ERK (ERKPP).

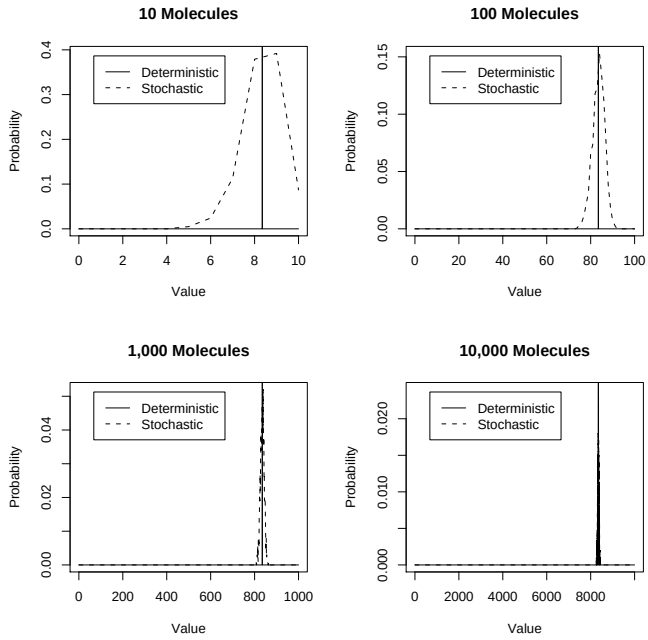

Figure 3. The probabilistic domain of variable X compared to the deterministic result. From top-left to bottom-right; 10, 100, 1,000, 10,000 molecules. The most probable value for [A] = [D] tends towards the deterministic answer as the number of molecules increases. Note that in these graphs the deterministic behaviour is cut off and actually rises to a probability of 1.

The Petri net in Figure 4 describes the typical modular structure for such a signalling cascade. The Petri net can be read as continuous or stochastic and in the following illustrative examples we recast the continuous and stochastic properties from Gilbert *et al.* [10] to PLTLc and perform model checking using MC2(PLTLc).

Properties C1, C2 & C3 We have performed the continuous queries C1, C2 and C3 (See Table 3) using the BioNessie simulator up to simulation time 400s and our MC2(PLTLc) model checker. As the set of simulations contains only one simulation which is the average behaviour of the model (deterministic simulation), the resulting probability value of either 1 or 0 is converted to a boolean value true or false through the probability operator $P_{\geq 1}$. The original analysis in [10] was performed through Biocham and its inbuilt LTL model checker. A comparison of our results to the original results are summarised in Table 3.

	PLTLc Query	BioNessie & MC2(PLTLc)	Biocham
C1	$P_{\geq 1}[((MEKPP < 0.001) \wedge (ERKPP < 0.0002))U(Raf\_P > 0.06)]$	**true**	**true**
C2	$P_{\geq 1}[((Raf\_P > 0.06) \wedge (ERKPP < 0.0002)) \rightarrow ((Raf\_P > 0.06) \wedge (ERKPP < 0.0002))U(MEKPP > 0.004)]$	**false**	**true**
C3	$P_{\geq 1}[((Raf\_P > 0.06) \wedge (MEKPP > 0.004)) \rightarrow ((Raf\_P > 0.06) \wedge (MEKPP > 0.004))U(ERKPP > 0.0005)]$	**true**	**true**

Table 3. The results for the replication of C1, C2 and C3 queries [10] in PLTLc, showing a discrepancy in C2 with the original (Biocham) results.

The difference in the results is due to the different ODE solvers used in BioNessie and Biocham. We found that due to the adaptive time steps used in Biocham's ODE solver, no state information is outputted for an important time period which is

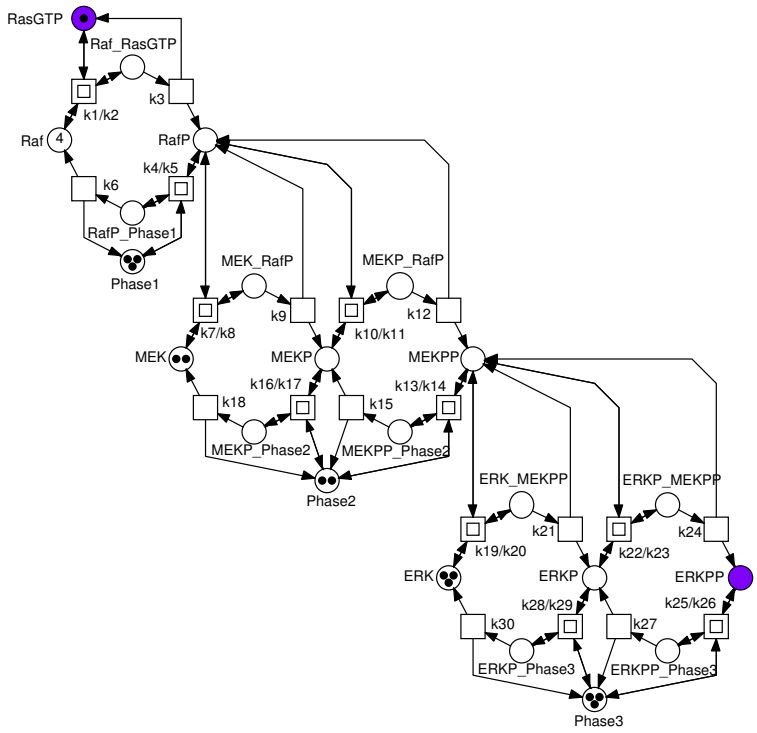

Figure 4. The Petri net for the ERK/MAPK pathway model [20] with initial marking for the 4 level version.

a counter-example to the C2 query, shown in Figure 5. The fixed time step and sufficient granularity of time points used in BioNessie provided state information which was a counter-example to this query, thus resulting in a false value. This is an example of where the simulator choice affects the model checking result.

Property S1 The first stochastic property S1 is defined as: "What is the probability of the concentration of RafP increasing, when starting in the first encountered state where the level is already at L?" This is written in CSL and PLTLc as:

$P_{=?}[(RafP = L)U(RafP > L)\{RafP = L\}]$

This query was checked using MC2(PLTLc) with 1,000 Gillespie simulation runs up to simulation time 300s at 4 and 8 molecules. In this case we have had to use MC2(PLTLc) in an experiment approach for the placeholder L. As the filter relativises the property to the placeholder L, each property in the experiment has a different filter state – a free variable cannot perform this relativising. The original results using PRISM at 8 concentration levels contrasted with our reproduced results are shown in Figure 7. Importantly, the time taken to perform both the simulation and model checking was significantly less in our method. Furthermore, unlike in the original paper, the time efficiency of our approach means that we do not have to impose time bounds on temporal operators as done in the original paper.

We extend the analysis of the stochastic properties S1 and S2 in [10] to higher numbers of molecules. First this acts as a platform to demonstrate the computational efficiency of our approach. However more than this, such system granularities are required to accurately check some behaviours. For example, in order to check increases in MEKPP and ERKPP in a similar manner to S1 and S2 we would need 50 and 200 molecules respectively. Furthermore, to check these properties, we would need two and three placeholders respectively.

We have extended this analysis up to 4,000 molecules shown in Figure 6 and observe that when increasing the number of molecules, the behaviour of the pathway tends towards the deterministic behaviour. The deterministic behaviour states that the protein RafP will always increase (property probability 1) until it reaches its maximum concentration value of around 0.1182 $mMol$. With increasing molecules, the maximum possible number of molecules in the stochastic behaviour of RafP tends towards the deterministic maximum (vertical line). The stochastic behaviour is seen to tend towards a probability of 0.5 between its maximum and minimum value, due to the stochastic nature where there is always a possibility of the protein

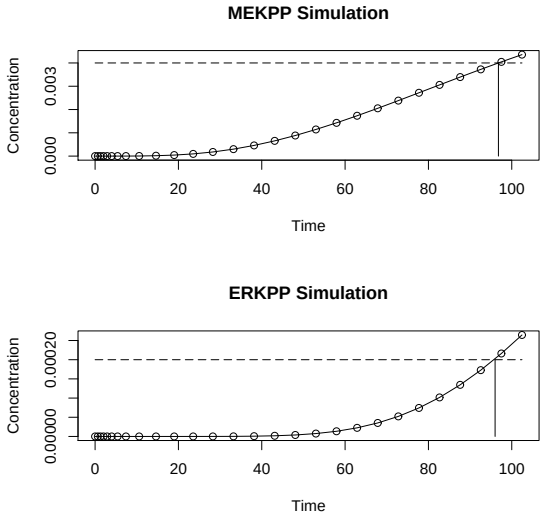

Figure 5. The output of Biocham simulation showing that it does not output a state in the time period where ERKPP (bottom) $>$ 0.0002 before MEKPP (top) $>$ 0.004, which is a counter example to C2.

decreasing or increasing.

Property S2 The second stochastic property S2 is defined as: "What is the probability that, given the initial concentrations of RafP, MEKPP and ERKPP being zero, the concentration of RafP rises above some level L while the concentrations of MEKPP and ERKPP remain at zero, i.e. RafP is the first species to react?" This is written in CSL and PLTLc as:

$P_{=?}[((MEKPP = 0) \wedge (ERKPP = 0))U(RafP > L)\{(MEKPP = 0) \wedge (ERKPP = 0) \wedge (RafP = 0)\}]$

To perform this analysis, we use the same simulation time (300s) and number of runs (1,000) as per S1. In this case we can use a constraint approach where we replace the placeholder L with a free variable $\$L$:

$P_{=?}[((MEKPP = 0) \wedge (ERKPP = 0))U(RafP > \$L)\{(MEKPP = 0) \wedge (ERKPP = 0) \wedge (RafP = 0)\}]$

For a single simulations, this sets the free variable \$L to the natural numbers which RafP is greater than while MEKPP and ERKPP are 0. Over the 1,000 simulations, the probabilistic domain of $\$L$ contains for each value the probability that RafP is greater than it while MEKPP and ERKPP are 0. The reproduction of the original results at 8 molecules is shown in Figure 7. Similar to S1, we have extended model checking up to 4,000 molecules shown in Figure 8 and note that the stochastic behaviour again begins to approximate the deterministic behaviour. In the deterministic behaviour, only at the initial state of the system are RafP, MEKPP and ERKPP all zero, hence a probability of 1 at this state and probability of 0 elsewhere. With increasing molecules, the stochastic behaviour becomes less curved and more step-like, tending towards the vertical line in the deterministic behaviour.

From Figure 7 it can be seen that there is a large difference between the original results and the reproduced results using PLTLc. This is due to the discussed difference in filter implementation, where we assess many initial states and CSL in PRISM assesses only one initial state.

We observe a similar gain in efficiency for property S2 when using the constraint approach as reported for the two reaction model in Section 4.1.

4.3. Oscillation Example

We have also performed model checking on a parameter scan of the continuous oscillating Kholodenko model of the MAPK pathway [19]. This model of the MAPK pathway is smaller than the Levchenko model, comprising 8 species and 10 reactions. We vary the strength of the negative feedback (inhibition) in the model, said to account for the oscillatory behaviour, and note the effect on oscillations. The pattern used to check for oscillatory behaviour is written in PLTL as:

$P_{\geq 1}[F(d(Protein) > 0 \wedge F(d(Protein) < 0 \wedge ...)))]$

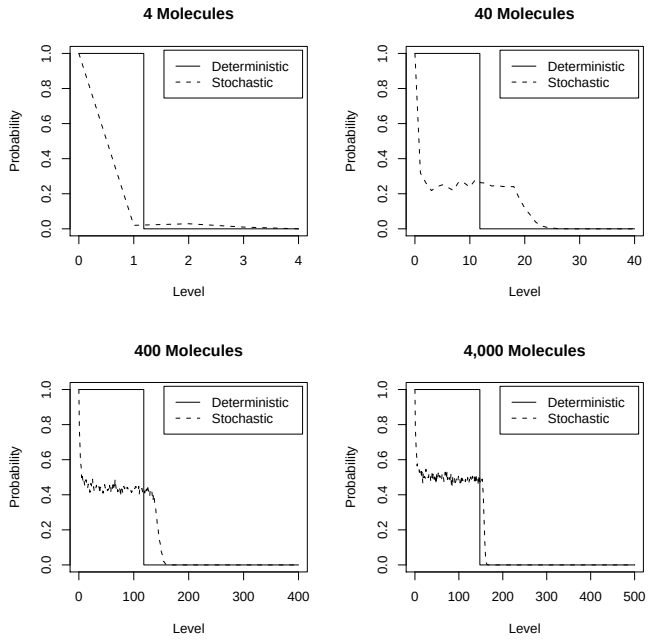

Figure 6. Model checking for property S1 using a larger number of molecules. From top-left to bottom-right; 4, 40, 400 and 4,000. This shows a progression towards the deterministic behaviour as the number of molecules increases.

This pattern can be used to check for a varying number of peaks in oscillatory behaviour. Note that such use requires knowledge of the expected start orientation of oscillations – oscillations start upwards or downwards. The results for checking for various number of peaks in a simulation time of 5,000s while varying the strength of negative feedback Ki between 1 and 30 $nMols^{-1}$ in steps of 1 are summarised in Figure 9.

There are differences between our approach to oscillation detection and the approaches of CTL and CSL. We can only ever check for oscillating behaviour in the simulated time period whereas exact approaches can guarantee oscillation forever. Furthermore, we base detection through a numerical approach using the derivative pattern. This means we can only check for oscillations when the derivatives are trustworthy, however in the stochastic world the derivatives would be the derivatives of the noise rather than the signal. Various measures were considered to overcome this; Fourier transforms, curve fitting, noise reduction. However, oscillation detection in the stochastic world remains a challenge in this approach.

5. Related Work

Two approaches can be distinguished in probabilistic model checking; exact and approximative. Exact methods provide - if they are applicable - a higher accuracy, however usually with much higher costs than approximative methods. Exact methods construct the (discrete or continuous) Markov chain, so they require finite state spaces, i.e. bounded models, and to apply them to complex dynamics is still a challenge. Exact methods are for instance used in PRISM [21] for PCTL/CSL model checking, and in LiQuor [7] for automata-based LTL model checking.

Approximative model checking of CSL using discrete event simulation of probabilistic models has been proposed in [25] and implemented in the tool Ymer [24]. Unlike our offline model checking approach, they follow an inline (on-the-fly) approach by generating the simulation runs as long as needed to decide time-bounded properties and as many runs as necessary to pass an acceptance sampling test. The required time bound on the temporal operators brings about the termination of the generation of sample executions. In our approach, the time bound is specified as parameter to the stand-alone stochastic simulation algorithm.

While the inline approach allows the adjustment of the length and number of simulation runs according to the given property to be checked, it prohibits the reuse of established stand-alone simulators, including deterministic ones. Moreover,

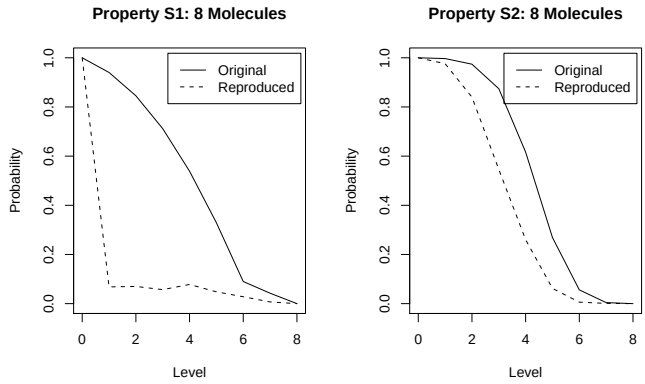

Figure 7. The original results from PRISM [10] at 8 levels contrasted with the reproduced results using MC2(PLTLc) for properties S1 and S2.

our decoupled approach permits the validation of a synthetic biological system by checking time series data recorded in biochemical experiments against the behaviour of the model used for the design of the system.

Ymer supports nested CSL queries, whereby the total number of required samples grows rapidly with the level of nesting. To solve this problem, a combination of exact and statistical techniques is proposed. On the contrary, we introduce a dedicated linear-time logic which seems to be more suited to the analysis of sets of independent simulation runs, i.e. in a non-branching time scenario. So, PLTLc could be considered as a linear-time counterpart to CSL, and can easily be used to formalize the visual evaluation of diagrams as generated by deterministic/stochastic simulation runs or by recording experimental time series.

Approximative model checking of PLTL and PCTL by distributed path sampling is also applied in the tool APMC [18]. APMC takes PRISM's modelling language as its input language, so it requires bounded models and *a priori* knowledge of the boundedness degree.

MC2 [13] performs approximative LTL model checking by random walks on the Büchi automaton, so it requires finite state spaces to follow the automata-based LTL model checking approach.

None of these probabilistic model checkers supports the constraint approach, which has been proven to be extremely powerful to replace the much more expensive experiment approach.

Finally, approximative LTL model checkers are also available in Simpathica [2] and Biocham [9], however both support only the analysis of deterministic simulation runs.

6. Summary

We have defined a Probabilistic Linear-time Temporal Logic with numerical constraints (PLTLc) and developed a Monte Carlo Model Checker for PLTLc properties MC2(PLTLc). With increasing system granularities and variables in the property, model checking experiments quickly become intractable. We define probabilistic domains for free variables in a property which can replace model checking experiments (comprising of a set of properties) with a single property. We have illustrated the increase in efficiency with our approach through the analysis of a biochemical system, specifically a model of an intracellular signalling pathway. MC2(PLTLc) utilises a general technique for model checking of dynamic systems. It accepts any quantitative time-series output from a simulation or actual system behaviour. Our work contributes to the emerging field of synthetic biology by proposing a rigourous approach for the structured formal engineering of biological systems. This method can be applied to perform checking over the behaviours of models or indeed over behaviours of a biological system, permitting verification of a system which is being constructed according to the model.

The MC2(PLTLc) tool, together with the sample models in this paper, simulation outputs, PLTLc queries and analysis results are available at:

`www.brc.dcs.gla.ac.uk/software/mc2`.

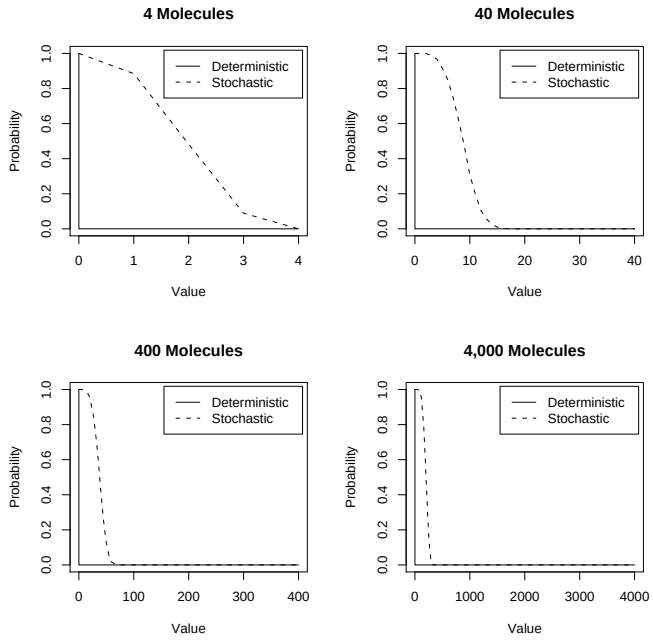

Figure 8. The probabilistic domain of L through model checking property S2 in PLTLc using a larger number of molecules. From top-left to bottom-right; 4, 40, 400 and 4,000. This shows a progression towards the deterministic behaviour as the number of molecules increases.

References

[1] A. Aziz, K. Sanwal, V. Singhal, and R. K. Brayton. Verifying Continuous-Time Markov Chains. In Rajeev Alur and Thomas A. Henzinger, editors, *Eighth International Conference on Computer Aided Verification CAV*, volume 1102, pages 269–276, New Brunswick, NJ, USA, / 1996. Springer Verlag.

[2] M. Antoniotti, A. Policriti, N. Ugel, and B. Mishra. Model Building and Model Checking for Biochemical Processes. *Cell Biochemistry and Biophysics*, 38:271–286, 2003.

[3] C. Baier. *On Algorithmic Verification Methods for Probabilistic Systems*. Habilitation thesis, University of Mannheim, 1998.

[4] BioNessie. A Biochemical Pathway Simulation and Analysis Tool. University of Glasgow, www.bionessie.org.

[5] M. Calder, V. Vyshemirsky, D. Gilbert, and R. Orton. Analysis of Signalling Pathways using Continuous Time Markov Chains. In *T. Comp. Sys. Biology VI*, pages 44–67. LNCS 4220, Springer, 2006.

[6] L. Calzone, N. Chabrier-Rivier, F. Fages, and S. Soliman. Machine Learning Biochemical Networks from Temporal Logic Properties. In *T. Comp. Sys. Biology VI*, pages 68–94. LNCS 4220, Springer, 2006.

[7] F. Ciesinski and C. Baier. LiQuor: A tool for Qualitative and Quantitative Linear Time Analysis of Reactive Systems. In *QEST*, pages 131–132, 2006.

[8] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press 1999, third printing, 2001.

[9] F. Fages and A. Rizk. On the Analysis of Numerical Data Time Series in Temporal Logic. In *Proc. CMSB 2007*, pages 48–63. LNCS/LNBI 4695, Springer, 2007.

[10] D. Gilbert, M. Heiner, and S. Lehrack. A Unifying Framework for Modelling and Analysing Biochemical Pathways Using Petri Nets. In *Proc. CMSB 2007*, pages 200–216. LNCS/LNBI 4695, Springer, 2007.

[11] C. S. Gillespie, D. J. Wilkinson, C. J. Proctor, D. P. Shanley, R. J. Boys, and T. B. L. Kirkwood. Tools for the SBML community. *Bioinformatics*, 22(5):628–629, 2006.

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

[13] R. Grosu and S. A. Smolka. Monte Carlo Model Checking. In *TACAS*, pages 271–286, 2005.

[14] H. Hansson and B. Jonsson. A Logic for Reasoning about Time and Reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[15] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic Model Checking of Complex Biological Pathways. In *Proc. CMSB 2006*, pages 32–47. LNCS/LNBI 4210, Springer, 2006.

[16] M. Heinemann and S. Panke. Synthetic Biology - Putting Engineering into Biology. *Bioinformatics*, 22(22):2790–2799, 2006.

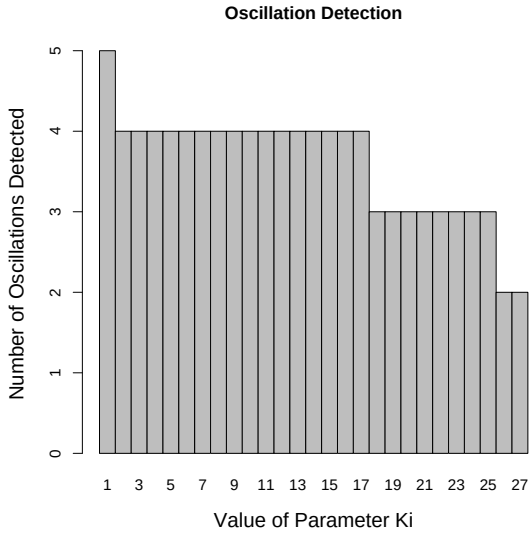

Figure 9. The number of oscillations detected in 5,000 seconds (Y-axis) for a varying value of the parameter Ki (X-axis), responsible for the inhibition in the Kholodenko model.

[17] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate Probabilistic Model Checking. In *Proc. 5th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'04): 307–329*, volume 2937 of *LNCS*. Springer, 2004.

[18] T. Hérault, R. Lassaigne, and S. Peyronnet. APMC 3.0: Approximate Verification of Discrete and Continuous Time Markov Chains. In *QEST*, pages 129–130, 2006.

[19] B. N. Kholodenko. Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. *Eur. J. Biochem. 2000, 267(6): 1583–1588*, 2000.

[20] A. Levchenko, J. Bruck, and P. Sternberg. Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. *Proc Natl Acad Sci USA*, 97(11):5818–5823, 2000.

[21] D. Parker, G. Norman, and M. Kwiatkowska. *PRISM 3.0.beta1 Users' Guide*, 2006.

[22] A. Pnueli. The Temporal Semantics of Concurrent Programs. *Theor. Comput. Sci.*, 13:45–60, 1981.

[23] SyntheticBiology.org. www.syntheticbiology.org.

[24] H. L. S. Younes, M. Z. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. Statistical Probabilistic Model Checking. *STTT*, 8(3):216–228, 2006.

[25] H. L. S. Younes and R. G. Simmons. Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling. *LNCS*, 2404:223–235, 2002.