A Model Checking Approach to the Parameter Estimation of Biochemical Pathways

Robin Donaldson and David Gilbert

Bioinformatics Research Centre, University of Glasgow
Glasgow G12 8QQ, Scotland, UK
{radonald,drg}@brc.dcs.gla.ac.uk

Abstract. Model checking has historically been an important tool to verify models of a wide variety of systems. Typically a model has to exhibit certain properties to be classed 'acceptable'. In this work we use model checking in a new setting; parameter estimation. We characterise the desired behaviour of a model in a temporal logic property and alter the model to make it conform to the property (determined through model checking). We have implemented a computational system called MC2(GA) which pairs a model checker with a genetic algorithm. To drive parameter estimation, the fitness of set of parameters in a model is the inverse of the distance between its actual behaviour and the desired behaviour. The model checker used is the simulation-based Monte Carlo Model Checker for Probabilistic Linear-time Temporal Logic with numerical constraints, MC2(PLTLc). Numerical constraints as well as the overall probability of the behaviour expressed in temporal logic are used to minimise the behavioural distance. We define the theory underlying our parameter estimation approach in both the stochastic and continuous worlds. We apply our approach to biochemical systems and present an illustrative example where we estimate the kinetic rate constants in a continuous model of a signalling pathway.

1 Introduction

Modelling biochemical systems is a key activity in Systems Biology [1], for example in the area of signal transduction pathways [2]. Models can be used to increase the understanding of a biochemical network in terms of the interactions between the components (the topology), or their dynamic behaviour. The representation of such systems can range from the informal, for example pathway diagrams, to the formal, which include qualitative and quantitative descriptions – the latter being stochastic or continuous and requiring kinetic information including reaction rates and concentrations/mass of components [3]. Formal models can permit both simulation of behaviour as well as the analysis of behavioural properties.

One important issue is how models are obtained, a process that usually involves fitting to some trusted data. Model fitting can involve identification of alternative topologies [4], choice of types of kinetic laws and formulae, and estimation of kinetic rate constants and initial concentration/mass values [5]. This

M. Heiner and A.M. Uhrmacher (Eds.): CMSB 2008, LNBI 5307, pp. 269–287, 2008.

is a challenging task in the biochemical field, especially due to the lack of reliable quantitative data.

A biologist or biochemist will often be unsure about exact values of biochemical species over time due to the nature of the wet-lab experimental technology, and will describe behaviour in a semi-quantitative manner. For example, "the concentration of the protein peaks within 2 to 5 minutes and then falls to less than 50% of the peak value within 60 minutes". A significant challenge is how to automatically build a model which conforms to semi-quantitative behaviour. Temporal logic is well-suited to formally represent such semi-quantiative descriptions.

In this paper we report on work to use model checking to drive the estimation of parameter values in biochemical models. We use a probabilistic temporal logic to describe desired behavioural properties and the MC2(PLTLc) model checker [6] to compute how closely the behaviour of a model conforms to the desired behaviour. We use a genetic algorithm to explore model space in order to generate a set of models which exhibit some desired behaviour. We have defined a novel extension of PLTL temporal logic [7] in order to permit a fine grained distance function suitable for use in our model exploration approach. This enables us to operate over both continuous as well as stochastic models.

Given a model with a fixed topology and ranges of parameter values to be explored, we can use a genetic algorithm to explore model space and generate values of kinetic rate constants and initial concentration/masses for which the model exhibits the desired behaviour. We illustrate this approach by considering a continuous model of the well-known MAPK signalling pathway stimulated by EGF [8], and derive values for kinetic rate constants such that the behaviour conforms to that under NGF stimulation. In doing so we confirm the results of [8] which showed that the desired results could be achieved by varying only one parameter, V28, 40-fold however in our approach we perform multi-parameteric fitting and show that the same desired behaviour can be achieved by varying a set of kinetic parameters with V28 only requiring a 16-fold increase.

Our approach contributes to the field of systems biology in terms of model construction from desired behaviour as well as to the field of synthetic biology in terms of system design and construction from desired behaviour properties [9]. This is the first step in a general approach to automatically constructing models based on a formal description of desired behaviour of a model.

This paper is organised as follows. The following section outlines the theory of our approach in both the stochastic and continuous worlds. The next section describes our computational system for parameter estimation, MC2(GA). We next present a case study where we estimate the parameters of a model of a signalling pathway. In doing so, we attempt to answer an important biochemical question concerning this pathway – what are the underlying model differences explaining the cell reactions to different signals. We conclude with a summary of our approach and propose further research ideas.

2 Theory

This section sets out the theory behind our computational system for parameter estimation. First we explain the syntax and semantics of the PLTLc temporal logic. Next, we describe how the desired behaviour of a model can be characterised using PLTLc. Then we define probabilistic domains – the relationship between the values of a free variable in a PLTLc property and the overall probability of the behaviour – and show how they can be helpful in characterising the desired behaviour. Finally, we explain how to build a distance metric of the distance between the model's behaviour and the desired behaviour.

2.1 PLTLc Syntax

Linear-time Temporal Logic (LTL) [10] is the fragment of full Computational Tree Logic (CTL*) [11] without path quantifiers, implicitly quantifying universally over all paths. LTL has been introduced in a probabilistic setting in [7], and extended by numerical constraints over real value variables in [12]. PLTLc combines both extensions, complemented by the filter construct as used in Probabilistic Computational Tree Logic (PCTL) [13] and Continuous Stochastic Logic (CSL) [14]. We start with the LTL with numerical constraints (LTLc) syntax:

$$\begin{aligned}\phi \ ::= \ & X\phi \mid G\phi \mid F\phi \mid \phi U\phi \mid \phi R\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \phi \rightarrow \phi \mid \\ & value = value \mid value \neq value \mid value > value \mid value \geq value \mid \\ & value < value \mid value \leq value \mid true \mid false\end{aligned}$$

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 range of 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:

$$\begin{aligned}value \ ::= \ & value + value \mid value - value \mid value * value \mid value/value \mid \\ & \$fVariable \mid Variable \mid function \mid Int \mid Real\end{aligned}$$

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* and *d*. 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 defined to be ϕ without any temporal operator (X, G, F, U, R), and 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.

2.2 PLTLc Semantics

We introduce the semantics of PLTLc in an informal manner to cater to a wide audience. The formal semantics of PLTLc are described in full in [6].

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 $\pi^i = s_i, s_{i+1}, ..., s_n$. Each path in the set of paths can be evaluated to a boolean value as to whether ϕ or $\phi\{SP\}$ holds. 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 which contains a single path has a probability of either 0 or 1.

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}.

2.3 Characterising Biochemical Species' Behaviour

The behaviour of biochemical species can be described with PLTLc using four distinct descriptive approaches, with increasing specificity; qualitative, semi-qualitative, semi-quantitative and quantitative. Qualitative uses derivatives of biochemical species concentrations/mass and temporal operators to describe the general trend of the behaviour. Semi-qualitative extends qualitative with relative concentrations. Semi-quantitative extends semi-qualitative with absolute time values. Finally, quantitative extends semi-quantitative with absolute concentration values. For example, transient activation of a biochemical species called Protein can be expressed in these approaches:

qualitative: Protein rises then falls

$$\mathbf{P}_{=?}\ [\ d(\text{Protein})\ >\ 0\ U\ (\ G(\ d(\text{Protein})\ <\ 0\ )\ )\ ]$$

semi-qualitative: Protein rises then falls to less than 50% of peak concentration

$$\mathbf{P}_{=?}\ [\ (\ d(\text{Protein})\ >\ 0\ )\ U\ (\ G(\ d(\text{Protein})\ <\ 0\ )\ \wedge\ F(\ [\text{Protein}]\ <\ 0.5 * max[\text{Protein}]\ )\ )\ ]$$

semi-quantitative: Protein rises then falls to less than 50% of peak concentration at 60 minutes

$$\mathbf{P}_{=?}\ [\ (\ d(\text{Protein})\ >\ 0\ )\ U\ (\ G(\ d(\text{Protein})\ <\ 0\ )\ \wedge\ F(\ \text{time}\ =\ 60\ \wedge\ \text{Protein}\ <\ 0.5 * max(\text{Protein})\ )\ )\ ]$$

quantitative: Protein rises then falls to less than $100\mu Mol$ at 60 minutes

$$\mathbf{P}_{=?}\ [\ (\ d(\text{Protein})\ >\ 0\ )\ U\ (\ G(\ d(\text{Protein})\ <\ 0\ )\ \wedge\ F(\ \text{time}\ =\ 60\ \wedge\ \text{Protein}\ <\ 100\ )\ )\ ]$$

In our case study we find that the desired behaviour of the model is most suited to semi-quantitative PLTLc. In fact, the informal explanation of results from biochemical experiments bears a striking similarity to semi-quantitative PLTLc.

2.4 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 possible 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 I in the probabilistic domain. A count is performed on the set of domains of validity for the number of domains of validity 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.

The semi-quantitative property from the previous section can be enhanced with free variables:

semi-quantitative with free variables. Protein rises then falls to less than 50% of peak concentration at 60 minutes

$$\mathbf{P}_{=?}\,[\,(\,d(\text{Protein})\;>\;0\,)\,U\,(\,\text{Protein}\;\geq\;\$PeakConc\,\wedge\,G(\,d(\text{Protein})\;<\;0\,)\;\wedge\,F(\,\text{time}\;=\;60\;\wedge\;\text{Protein}\;<\;0.5*max(\text{Protein})\,)\,)\,]$$

where the probabilistic domain of $\$PeakConc$ associates with each value in the domain the probability of a peak of at least that value.

2.5 Distance Metrics

The distance between a model's behaviour and the desired behaviour can be calculated using a distance metric. We define a metric for the distance, with respect to some property ψ, from the behaviour of the model M to the desired behaviour M_{des}. The distance metric, written $d_\psi(M, M_{des})$, should satisfy the metric properties:

$d(x, y) > 0$ for $x \neq y$ w.r.t. ψ, $\quad d(x, y) = 0$ for $x = y$ w.r.t. ψ

$d(x, y) = d(y, x)$ for all x, y , $\quad d(x, y) \leq d(x, z) + d(y, z)$ for all x, y, z

The metric is domain and behaviour specific, and can be based on the probability of the property or the probabilistic domains.

Perhaps the simplest definition of the metric is the square difference between the model's probability of exhibiting some behavioural property ψ, $P(\psi)$ and desired probability $P_{des}(\psi)$. For example, we may want the property ψ to always hold in which case $P_{des}(\psi)$ is 1. The distance function is then written:

$d_\psi(M, M_{des}) = |P(\psi) - P_{des}(\psi)|^2$

This approach works well in the stochastic world where the model exhibits many behaviours and the probability of the property is in the range $[0 \rightarrow 1]$. However, in the continuous world there is a single behaviour and the probability

is either 0 or 1, thus the metric is too coarse grained to be used in a search algorithm in the continuous world. To be useful in the search algorithm, the distance metric should return a value which indicates whether altering the current model has caused its behaviour to be closer to the desired behaviour, therefore providing a gradient for the search algorithm to ascend.

Definitions of the distance metric over probabilistic domains of free variables can result in finer grained distance values, crucial for distance metrics in the continuous world. For a free variable $\$X$ in a property ψ, we can compare the probabilistic domain in the model $\$X$ with the desired probabilistic domain $\$X_{des}$. To do so, we use the residual sum of squares function, RSS:

$RSS(\$X, \$X_{des}, m, n) = \sum_{i=m}^{n} |\$X(i) - \$X_{des}(i)|^2$

where m to n is some sub-section of the domain being assessed. Hence, we could desire that a free variable describing the peak concentration value in transient behaviour $\$PeakConc$ of a continuous model is at least $50\mu Mol$. It is then simple to set up a desired probabilistic domain $\$PeakConc_{des}$ with probability 1 for values 0 to 50. A call to $RSS(\$Peakconc, \$PeakConc_{des}, 0, 50)$ would then return a value of how close the current model is to having its peak concentration value at least value $50\mu Mol$.

We can implement a distance metric using the RSS function for any number of free variables we define in our PLTLc property. In the case that we wish to optimise more than one probabilistic domain, we normalise the RSS values between 0 and 1:

$$d_\psi(M, M_{des}) = \frac{RSS(\$X, \$X_{des}, m, n)}{(n-m)} + \frac{RSS(\$Y, \$Y_{des}, u, v)}{(v-u)} + \ldots$$

3 Computational System

We implemented a computational system called the Monte Carlo Model Checker with a Genetic Algorithm, MC2(GA). The purpose of this computational system is to estimate the parameters of a model to make it exhibit desired behavioural properties. A genetic algorithm is used to move models through parameter space to minimise their distance to the desired behaviour, checked using a model checker.

A genetic algorithm [15] operates over a *population* of *individuals*, each of which is represented by their *chromosome* containing one or more *genes*. The individuals have an associated fitness based on how "good" their genes are. A selection of individuals from the current population is performed which will be used to create the next generation. Genetic operations on the chromosomes of these selected individuals (reproduction, crossover and mutation) is used to build a next generation with (hopefully) improved overall fitness.

Each model in our MC2(GA) system has a fixed structure and is represented by a chromosome, which is a set of kinetic rate constant values to be estimated (the model's genes) within predefined ranges. The chromosome could equally include initial concentrations/masses.

In the initial generation, a population of models is created by assigning to each model random values within the ranges for the kinetic rate constants. The

number of models in the population should be proportional to the size of the parameter space being explored. Each model in the population is evaluated to a fitness value related to the distance of its behaviour to the desired behaviour, hence a model with a smaller distance to the desired behaviour has a higher fitness. This is achieved by formalising the desired behaviour in temporal logic and the novel use of a model checker to calculate the distance of the model to the behaviour. Our approach is to vary models' kinetic rate constant values in order to maximise their fitness values.

After the initial generation which builds a population of models with their related fitness values, a subset of the population is selected to survive. Roulette-wheel selection is used where models in the population are chosen to survive probabilistically, with fitter models having a higher probability of survival. This is done to keep a small number of less fit models in the population such that we do not converge on a solution too early. If the computational system is exploring a high dimensional parameter space, it is important to maintain good coverage of this large space. The population for the next generation is created from the selected models by performing genetic operations on these models' chromosomes representing the kinetic rate constant values. A chromosome may be duplicated (*reproduction*), a section between two chromosomes may be swapped (*crossover*) or a section of one chromosome may be randomly altered (*mutation*) within preferred constraints. The models in the new population are evaluated to their fitness values and then go on to form the next generation.

The best and average fitness value of a model in the population should increase over successive generations of this algorithm. There are stochastic elements to the algorithm however, including random mutation and probabilistic selection of models for the next generation. Hence, it is not always the case that there will be a continual increase in best or average fitness value, though the general trend should increase. Various stopping conditions in this algorithm can be used– we choose to stop after the best fitness of a model in the population has not changed significantly after 10 generations or after a maximum of 100 generations has elapsed.

A population of models with their respective fitness values is returned upon termination of the genetic algorithm. This is quite a powerful result of parameter estimation. By the very nature of a genetic algorithm, we get a set of candidate solutions, which may be representative of more than one general solution type. Hence, with the semi-quantitative description of the desired behaviour, we can get many models, possibly grouped into distinct sub-populations, which exhibit the desired behaviour.

Although any search algorithm which uses a fitness function could be used in this approach, we have chosen a genetic algorithm because it avoids being lost in local minima, which is likely in high dimensional parameter spaces. A genetic algorithm avoids this by maintaining a population of candidate solutions and probabilistically keeping some low fitness solutions in the population between generations.

Our computational system, MC2(GA), currently operates over continuous models only. The desired behaviour of a model is expressed in the PLTLc temporal logic. Models are evaluated to a fitness value through interfaces to the continuous simulator, BioNessie Lite [16] and the Monte Carlo Model Checker for Probabilistic LTLc properties MC2(PLTLc) [17]. The model is simulated for a predefined amount of time and the simulation output is checked for the desired behaviour using MC2(PLTLc). A numerical value for the fitness of the model based on the result of model checking can be computed using the probability of the behavioural property and the probabilistic domains of free variables in the property. The fitness function using the probabilistic domains of free variables has been implemented using the theory described in Section 2.5. We employ the Java Genetic Algorithms Package (JGAP) [18] to move our population of models through parameter space in order to maximise their fitness.

4 Case Study: MAPK Pathway

We illustrate our technique to parameter estimation with a continuous model of the MAPK pathway.

4.1 Biochemical Motivation

The EGF signal transduction pathway conveys Epidermal Growth Factor signals from the cell membrane to the nucleus via the MAP Kinase cascade [2]. The model of the pathway in PC12 cells written in Systems Biology Markup Language (SBML) [19] is the subject of [8]. The same core MAPK cascade can also be stimulated by Nerve Growth Factor (NGF). The reaction of the cell to EGF stimulation is cell proliferation, however the response to NGF is cell differentiation. The active ligand-bound receptor acts as a kinase for the Shc protein. The active Shc and GS complex (ShcGS) binds with the inactive RasGDP complex which enables Son of sevenless homologue protein (SOS) to convert RasGDP to its active RasGTP form. RasGTP acts as a kinase to phosphorylate Raf, which phosphorylates MAPK/ERK Kinase (MEK), which in turn phosphorylates Extracellular signal Regulated Kinase (ERK). Feedback regulation of the pathway is through ShcGS dissociation catalysed by phosphorylated ERK. The EGF signal transduction pathway produces transient Ras, MEK and ERK activation whereas NGF stimulation produces sustained activation. The underlying differences of the models describing EGF and NGF stimulation is of key interest to biochemists.

The work in [8], referred to from now on as the original paper, attempted to discover the quantitative differences in initial concentrations and kinetic rate constants between models of these pathways with fixed topology. The authors varied the initial concentrations and kinetic rate constants within biochemically sensible ranges. Simulation was performed with the model using each parameter value in the range and the output was manually inspected for sustained Ras, MEK and ERK activation. A result of this work was the finding that a 40-fold increase in the kinetic rate constant of SOS dephosphorylation can change the behaviour of the model from transient activation to sustained activation.

We suggest that this analysis could be improved by constructing a formal definition of the desired behaviour in temporal logic, and using model checking of the desired behaviour to replace the manual inspection of the simulation outputs. This facilitates the automation detection of a model which exhibits the desired behaviour. We employ this in our computational system, MC2(GA), to vary many kinetic rate constants in the model in parallel to estimate a parameter set of the NGF signal transduction pathway.

4.2 Characterising the Desired Pathway Behaviour

The behaviour of sustained Ras, MEK and ERK activation arising from NGF stimulation observed in wet-lab experiment was described in rather informal statements in the original paper [8].

"The level of RasGTP rapidly reaches a maximum of up to 20% of total Ras within 2 min [then] the level of RasGTP is sustained at around 8% of total Ras."

Similar statements were made about sustained MEK and ERK activation. We have formalised these statements using semi-quantitative PLTLc such that a model could be automatically checked for these behaviours using the MC2(PLTLc) model checker. We formalised these statements in a way to account for biological error by relaxing the constraints, for example that the stable level of RasGTP is 8% to between 5% and 10%:

sustained Ras. Active Ras peaks within 2 minutes to a maximum of 20% of total Ras and is stable between 5% and 10% from at least 15 minutes

$$\mathbf{P}_{=?}\,[\,(\,d(\text{active Ras}) > 0\,) \wedge (\,d(\text{active Ras}) > 0\,)\,U\,(\,\text{time} \leq 2 \wedge \text{active Ras} \geq 0.15{*}\text{total Ras} \wedge \text{active Ras} \leq 0.2{*}\text{total Ras} \wedge d(\text{active Ras}) < 0 \wedge (\,d(\text{active Ras}) < 0 \wedge \text{time} < 15\,)\,U\,(\,G(\,(\text{active Ras}) \geq 0.05{*}\text{total Ras} \wedge \text{active Ras} \leq 0.10{*}\text{total Ras}\,)\,)\,)\,]$$

where the protein RasGTP is found in isolation and in two complexes, thus active Ras $= RasGTP{+}Ras\_Raf{+}Ras\_GAP$ and total Ras $= RasGTP{+}Ras\_Raf{+}Ras\_GAP{+}RasGDP + Ras\_ShcGS$.

sustained MEK. Active MEK peaks within 2 to 5 minutes and is stable between 40% and 50% of peak value from at least 15 minutes

$$\mathbf{P}_{=?}\,[\,(\,d(\text{MEKPP}) > 0\,) \wedge (\,d(\text{MEKPP}) > 0\,)\,U\,(\,\text{time} \geq 2 \wedge \text{time} \leq 5 \wedge d(\text{MEKPP}) < 0 \wedge (\,d(\text{MEKPP}) < 0 \wedge \text{time} < 15)\,U\,(\,G(\,\text{MEKPP} \geq 0.40{*}max(\text{MEKPP}) \wedge \text{MEKPP} \leq 0.50{*}max(\text{MEKPP})\,)\,)\,)\,]$$

sustained ERK. Active ERK peaks within 2 to 5 minutes and is stable between 85% and 100% of peak value from at least 15 minutes

$$\mathbf{P}_{=?}\,[\,(\,d(\text{ERKPP}) > 0\,) \wedge (\,d(\text{ERKPP}) > 0\,)\,U\,(\,\text{time} \geq 2 \wedge \text{time} \leq 5 \wedge d(\text{ERKPP}) < 0 \wedge (\,d(\text{ERKPP}) < 0 \wedge \text{time} < 15\,)\,U\,(\,G(\,\text{ERKPP} \geq 0.85 * max(\text{ERKPP})\,)\,)\,)\,]$$

4.3 Identification of Critical Parameters

The work reported in the original paper [8] varies parameters individually in the model and notes the effect on sustained Ras, MEK and ERK activation

by manual inspection of simulation output. We performed a similar analysis in an automated fashion, which made it easy to count the number of parameter values in a particular parameter range that gave our desired behaviour. Hence, rather than a simple yes/no answer, we were able to quantify the significance of a particular parameter regarding a particular behaviour. We used this feature to identify a set of critical parameters to vary in our MC2(GA) system. A further benefit of using an automated approach to detect desired output rather than manual inspection is that we can explore many possible behaviours, generated for example by varying one parameter within a large range.

In the absence of biochemical knowledge of acceptable ranges of kinetic rate constants, we varied each kinetic rate constant in the range $\pm$ 2 orders of magnitude from their original value. We simulated the continuous model for 60 minutes using 1,000 parameter values linearly spaced in the range. This produced a set of 1,000 simulation outputs, and we checked each one for the behaviour of sustained Ras, MEK and ERK activation expressed in PLTLc. We then computed the fraction of simulation outputs which satisfy the behavioural property over the number of simulation outputs. We call this fraction the parameter's *significance value*, such that a higher value represents parameter more likely to exhibit the desired behaviour. Each kinetic rate constant in the model was varied individually to produce their significance value.

Any kinetic rate constant with at least one non-zero significance value for sustained Ras, MEK or ERK is called a *critical parameter*. The identified critical parameters are listed in Table 1 along with their respective significance values. Although we ignored initial concentration parameters as they had little effect on sustaining activation in the original analysis, our approach can analyse initial concentrations in the same manner as kinetic rate constants.

From the significance values it is clear that although sustained Ras and ERK activation is quite possible, sustained MEK activation is more difficult to achieve. In fact, when varying parameters individually it was only possible to achieve this using the kinetic rate constant for SOS dephosphorylation, V_28. This was the solution found in the original paper [8] and we note that it has the highest sum of the three significance values. We suspected that other parameter sets could produce our desired behaviour and thus we varied several parameters in parallel using our computational system.

4.4 Genetic Algorithm

We first implemented a fitness function for use in MC2(GA) to describe how close a model is to sustained activation. The descriptions of sustained Ras, MEK and ERK activation given earlier were not particularly helpful in the continuous setting due to the probability being simply 0 or 1. A fitness function based on a description which includes free variables allows greater expressivity using the probabilistic domains. Hence, we have rewritten these descriptions of sustained behaviours using free variables:

Table 1. The identified critical kinetic rate constant parameters in the model with their significance values with respect to sustained Ras, MEK or ERK

parameter	sustained Ras	sustained MEK	sustained ERK
V_20	0.01	0.0	0.001
V_24	0.076	0.0	0.0
V_25	0.023	0.0	0.001
V_27	0.614	0.0	0.0
V_28	0.478	0.151	0.679
k1_14	0.0	0.0	0.778
k1_16	0.0	0.0	0.001
k1_18	0.001	0.0	0.807
k2_18	0.191	0.0	0.0
Km_20	0.001	0.0	0.797
kcat_21	0.001	0.0	0.688
kcat_23	0.001	0.0	0.186
Km_23	0.121	0.0	0.0
Km_25	0.001	0.0	0.157
kcat_26	0.0	0.0	0.001
Km_26	0.0	0.0	0.005

sustained Ras with free variables. Active Ras peaks within 2 minutes to a maximum of 20% of total Ras and is stable between any value in $\$RasTail1$ and any value in $\$RasTail2$ from at least 15 minutes

$$\mathbf{P}_{=?}\,[\,(\,d(\text{active Ras}) > 0\,) \wedge (\,d(\text{active Ras}) > 0\,)\,U\,(\,\text{time} \leq 2 \wedge \text{active Ras} \geq 0.15{*}\text{total Ras} \wedge \text{active Ras} \leq 0.2{*}\text{total Ras} \wedge d(\text{active Ras}) < 0 \wedge (\,d(\text{active Ras}) < 0 \wedge \text{time} < 15\,)\,U\,(\,G(\,\text{active Ras} \geq \$RasTail1 \wedge \text{active Ras} \leq \$RasTail2\,)\,)\,)\,]$$

sustained MEK with free variables. Active MEK peaks within 2 to 5 minutes with a peak greater than concentration 20,000 and is stable between any value in $\$MekppTail1$ and any value in $\$MekppTail2$ from at least 15 minutes

$$\mathbf{P}_{=?}\,[\,(\,d(\text{MEKPP}) > 0\,) \wedge (\,d(\text{MEKPP}) > 0\,)\,U\,(\,\text{time} \geq 2 \wedge \text{time} \leq 5 \wedge \text{MEKPP} > 20000 \wedge d(\text{MEKPP}) < 0 \wedge (\,d(\text{MEKPP}) < 0 \wedge \text{time} < 15)\,U\,(\,G(\,\text{MEKPP} \geq \$MekppTail1 \wedge \text{MEKPP} \leq \$MekppTail2\,)\,)\,)\,]$$

sustained ERK with free variables. Active ERK peaks within 2 to 5 minutes with a peak greater than concentration 350,000 and is stable above any value in $\$ErkppTail$ from at least 15 minutes

$$\mathbf{P}_{=?}\,[\,(\,d(\text{ERKPP}) > 0\,) \wedge (\,d(\text{ERKPP}) > 0\,)\,U\,(\,\text{time} \geq 2 \wedge \text{time} \leq 5 \wedge \text{ERKPP} > 350000 \wedge d(\text{ERKPP}) < 0 \wedge (\,d(\text{ERKPP}) < 0 \wedge \text{time} < 15\,)\,U\,(\,G(\,\text{ERKPP} \geq \$ErkppTail\,)\,)\,)\,]$$

The property ψ in our case study was the conjunction of the sustained Ras, MEK and ERK properties expressed with free variables above. To explain the addition of free variables in these properties we consider the active ERK shown in Figure 4.4. This figure illustrates the relationship of a continuous simulation

output of ERKPP to the probabilistic domains of the free variable $\$ErkppTail$ and the desired probabilistic domain $\$ErkppTail_{des}$. The values in the probabilistic domain $\$ErkppTail$ with probability 1 are in the range of values from 0 to the tail height. Our desired behaviour of the property is that it peaks within 2 minutes to a concentration of greater than 350,000 (defined in the PLTLc property) and that the tail of the peak remains within 85% of the peak height. We observe a tail characterised by $\$ErkppTail$ and now characterise our desired tail of at least 85% of the peak height, $\$ErkppTail_{des}$. This is done by setting values from 0 to 85% of the peak height value in the probabilistic domain of $\$ErkppTail_{des}$ to probability 1. Hence if the RSS between the model's tail and the desired tail is 0, then the model's tail is within 85% of it's own peak height (our desired property) and if the model's tail falls to concentration 0, then the distance is value 85% of the peak height:

$$erkppDistance = RSS(\$ErkppTail, \$ErkppTail_{des}, 0, max(\$ErkppTail_{des}))$$

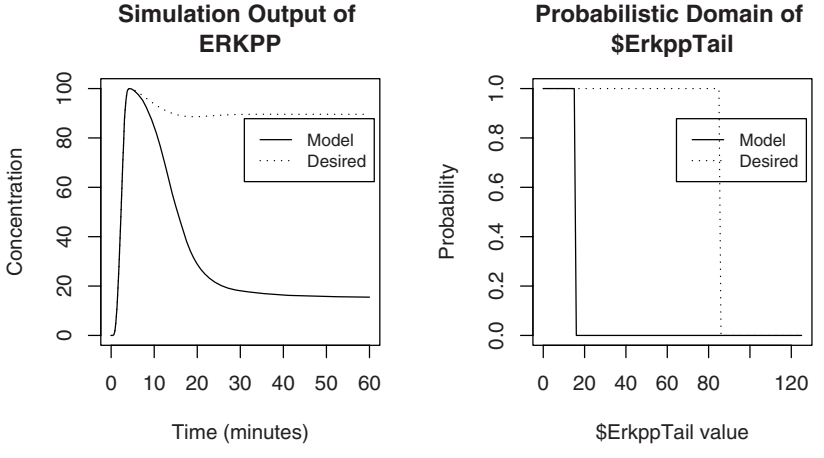

Fig. 1. Continuous simulation output of active ERK (left) with the probabilistic domain of free variables $\$ErkppTail$ (middle) and the desired probabilistic domain $\$ErkppTail_{des}$ (right). The probabilistic domain of $\$ErkppTail$ contains probability 1 for all values from 0 up to the tail height and the desired probabilistic domain $\$ErkppTail_{des}$ contains probability 1 for all values from 0 up to 85% of the peak height.

Next, recalling that sustained active MEK is defined to be stable between 40% and 50% of the peak, we set the desired probabilistic domain of $\$MekppTail1$ to have probability 1 for values 0 to 40% of the peak value and the desired probabilistic domain of $\$MekppTail2$ to have probability 1 for values 50% to 100% of the peak value. The distance of the active MEK behaviour was then:

$$mekppDistance = RSS(\$MekppTail1, \$MekppTail1_{des}, 0, max(\$MekppTail1_{des})) + RSS(\$MekppTail2, \$MekppTail2_{des}, min(\$\$MekppTail2_{des}), max(\$\$MekppTail2_{des}))$$

Finally, recalling that sustained active Ras is defined to be stable between 5% and 10% of total Ras, we set the desired probabilistic domain of $\$RasTail1$ to

have probability 1 for values 0 to 5% of total Ras and the desired probabilistic domain of $\$RasTail2$ to have probability 1 for values 10% to 100% of total Ras. The distance of the active Ras behaviour was then:

$rasDistance = RSS(\$RasTail1, \$RasTail1_{des}, 0, max(\$RasTail1_{des}))+$
$RSS(\$rasTail2, \$rasTail2_{des}, min(\$rasTail2_{des}), max(\$rasTail2_{des}))$

The overall metric describing the distance between a model and the desired behaviour can now be defined by averaging the sum of the normalised individual distances, $erkppDistance$, $mekppDistance$ and $rasDistance$, and ranges from 0 (identical) to 1:

$$d_\psi(M, M_{des}) = (\frac{erkppDistance}{85\%ERKPeak} + \frac{mekppDistance}{90\%MEKPeak} + \frac{rasDistance}{95\%TotalRas})/3$$

Note that if the model does not satisfy the PLTLc property – i.e. it does not peak or does not peak within the concentration or time constraints – then all values in the probabilistic domains of the free variables are set to probability 0, thus the distance is 1.

The definition of a model's fitness has opposite semantics from a distance metric – i.e. a high fitness value represents a good model. Hence the fitness function is:

$$fitness(M, M_{des}) = 1 - d_\psi(M, M_{des})$$

Finally, we added quantitative constraints to the properties to exclude the behaviour being observed with insignificantly small concentration values by requiring that both active ERK and active MEK be greater than approximately half of their peak value in the EGF signalling pathway model. The concentration of RasGTP is kept to a significant level through its definition stating that it must be between 15% and 20% of total Ras concentration.

As a proof of concept, we used MC2(GA) with a population of 2,000 models to estimate the value of the kinetic rate constant V_28 only using the range of ± 2 orders of magnitude from the original value as defined in the previous section. We hoped to reproduce the result from the original paper [8] that a 40-fold increase of V_28 (to value 3,000 $molecule^{-1}minute^{-1}$) produces sustained activation of Ras, MEK and ERK. The fitness of the best model immediately converged to value 1 and the genetic algorithm stopped 10 generations after the convergence, which can be seen in Figure 4. The V_28 value against respective fitness of the models in the final population can be seen in Figure 2. There is a wide range of V_28 values which produce the desired model behaviour – the V_28 value proposed in the original paper of 3000 $molecule^{-1}minute^{-1}$ falls within the range of models with a fitness value 1, which suggests that our genetic algorithm and behavioural properties are correct.

We then applied MC2(GA) to find novel parameter sets which exhibit the desired behaviour. We estimated the values of the 16 critical parameters identified in the previous section, listed in Table 1. We also applied MC2(GA) to the critical parameters without V_28, to assess whether V_28 is crucial to achieving sustained activation. The result of the convergence of these runs are presented in Figure 4. It can be seen from this figure that if the critical parameters are estimated with V_28, then the convergence is quicker and the best model returned was fitter. The best model returned when estimating the critical parameters had

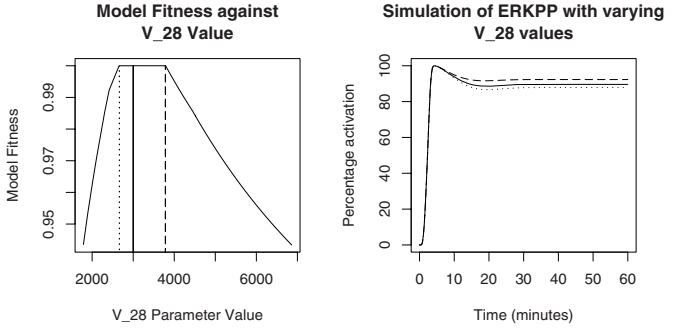

Fig. 2. The population of models from the genetic algorithm (left) and simulation output of three selected models (right). The model proposed in the original paper (solid) has fitness value 1. Also, the model with lowest (dotted) and highest (dashed) value of V_28 whilst maintaining fitness value 1 is shown.

fitness value 1, whereas with V_28 removed the best model returned had a fitness value approximately 0.93.

Figure 3 shows the output of one of the best model returned when estimating the critical parameters with and without V_28. Both behaviours showed good similarity (visually and in terms of fitness value) to the behaviour of the NGF signalling pathway outlined in the original paper. We also found that we can achieve a model with fitness value 1 through a 16-fold increase of V_28, compared with the original paper's 40-fold increase, if we also vary the other critical parameters.

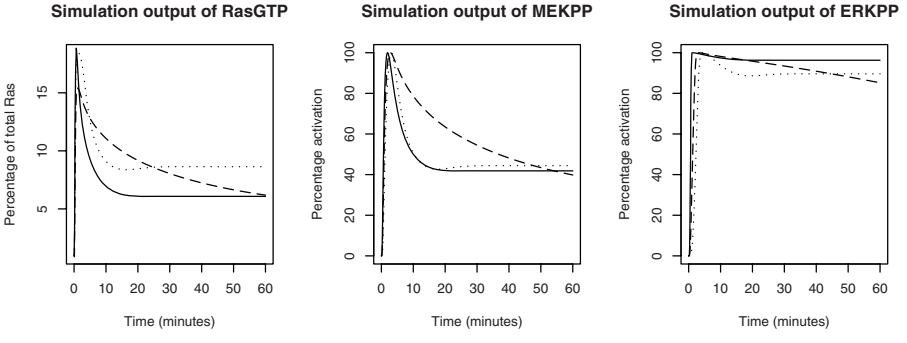

Fig. 3. The original model of the NGF signalling pathway (dotted) compared with the best model returned when varying the critical parameters (solid) and when varying the critical parameters without V_28 (dashed). The best model returned when varying the critical parameters only required a 16-fold increase in V_28 to achieve fitness value 1.

Finally we tested how MC2(GA) copes with a high dimensional parameter space by estimating the values of all 65 kinetic rate constants in the model. Again, these parameters were varied in the range ± 2 orders of magnitude from

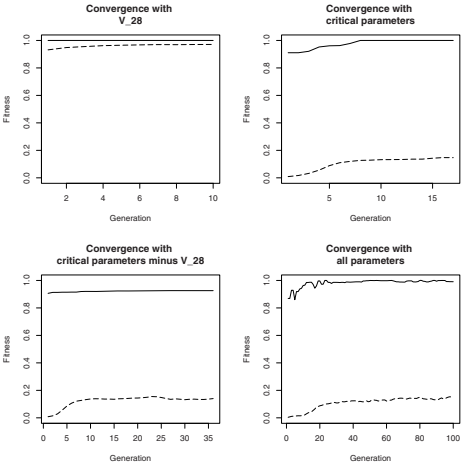

Fig. 4. From top-left to bottom right, the convergence of varying; V_28 only, the critical parameters, the critical parameters without V_28 and all parameters. The fitness of the best model in the population (solid) is shown as well as the average fitness of the models (dashed).

their original values. As shown in Figure 4 there is no convergence after the maximum of 100 generations. However, the best model returned had fitness value approximately 0.99. This is a strong result that even with such a high dimensional parameter space, MC2(GA) still found a viable solution.

To give an idea of the computational expense of our system, a single generation of the genetic algorithm took around ten minutes. Each generation contains on the order of 2000 simulation and model checking operations, thus the evaluation of each model's fitness value took around 300ms. Overall, with a population of M models and N generations, the number of calls to both the model checker and the simulator is $O(N * M)$. In our case study which contained 2,000 models and a maximum of 100 generations, we had around $2 \cdot 10^5$ calls to the model checker.

5 Related Work

Manual parameter estimation of a model can be performed, especially using insight into the real-life system. The authors in [8] manually altered a model of the MAPK pathway to change the behaviour from EGF stimulation to NGF stimulation. They focus their estimation to parameters which have been identified through biochemical experiments as possible targets to explain this difference. This work was accomplished using computational tools, such as a continuous simulator, however it relied heavily on manual inspection of simulation outputs. As such, the degree to which the model could be varied was limited to the amount of manual inspection possible. Hence, it was infeasible for the authors to vary the parameters within a large range or vary parameters in parallel.

Automated parameter estimation approaches employ a function which returns a value of how close model's behaviour is to some desired behaviour. A search algorithm can be used to alter the model to minimise the behavioural distance. Representing the desired behaviour using target data derived from experiments on the real-life system being modelled is an obvious choice. Approaches to parameter estimation using target data are studied and reviewed in [5]. Many functions can compute the difference of a model's output to the target data, such as Maximum Likelihood, Bayes and Weighted Least Squares.

However, the target data of models of biochemical systems results from wet-lab experiments. The data produced from such experiments is typically noisy and sparse with relative concentrations [20]. Any function calculating the difference of the output of a biochemical system model to the target data should account for this. [20] used weights on time-series data derived from wet-lab experiments to account for noise. Furthermore, a Bayesian approach to estimating the difference in model output and target data is the subject of [4].

Although the literature contains approaches to parameter estimation approaches using target data, we found little evidence of implementing parameter estimation with a target behaviour expressed in temporal logic. The closest we have found is [21] which specifies the expected behaviour of a continuous model of a biochemical pathway in the LTL temporal logic. The parameters in the model are varied in some range until a satisfying parameter is found and returned. However, this approach works only for the continuous world and the authors note the computational expense of parameter scans of multiple variables in parallel. They express desire for a "multi-valued measure of satisfaction" of a temporal property rather than the boolean result of LTL checking. This would facilitate the use of LTL as the target function in a search algorithm, allowing many parameters to be varied in parallel.

6 Conclusion

We have shown how we have used probabilistic temporal logic descriptions of biochemical pathway behaviours as the basis for a model checking approach to the parameter estimation of biochemical pathways. This is the first step in a general methodology for behaviour driven model construction.

A key aim of our approach is to be able to operate in both the stochastic and continuous worlds. The PLTLc temporal logic and its simulation-based model checker operate in both these worlds. It is especially important that the model checker is simulation-based as the use of current analytical model checker would be computationally infeasible due to the well-known state space explosion. This is further exacerbated when used within a search algorithm which will typically have many calls to the model checker – $2 \cdot 10^5$ in our case study. Furthermore, a novel aspect of the numerical constraints in PLTLc is that they can be applied in both the stochastic and continuous worlds; this is crucial to the calculation of the distance of a model's behaviour to the desired behaviour.

We have demonstrated our approach through a case study of a continuous model of the well-known MAPK signalling pathway. This model has previously been manually explored in [8]. They identify a single parameter which when modified by a 40-fold increase produced the desired behaviour. Having first characterising the desired behaviour in PLTLc, we then automatically identified a set of critical parameters. We then used our MC2(GA) system to estimate the values of the critical parameters and discovered novel kinetic rate constant parameter sets which produce the desired behaviour. These include parameter sets which do not require varying the parameter identified in [8]. Finally we showed that the computational system can operate with high dimensional parameter spaces by estimating the values of all 65 kinetic rate constants in the model.

The case study presented in this paper is of a continuous model of a biochemical pathway. However, the theory underlying this approach has been described for both the stochastic and continuous worlds. We are now working on applying this analysis to a stochastic model. Furthermore, we are currently able to estimate the kinetic rate constant and initial concentration/mass values in the model, however we could define theory to vary the model topology (adding, removing or altering reactions). We then could answer questions such as what are the topologies which give rise to particular behaviours of interest. Finally, PLTLc is rather unfriendly to a biologist who is not well versed in temporal logic. We are now developing a user-friendly interface for biologists to describe behaviours using PLTLc.

The computational system, MC2(GA), together with the case study results are available at: `www.brc.dcs.gla.ac.uk/software/mc2/mc2ga_bf`.

Acknowledgements

This research was supported by the SIMAP project which is funded by the European Commission framework 6 STREP programme.

References

1. Wolkenhauer, O.: Systems Biology: the Reincarnation of Systems Theory Applied in Biology? Briefings in Bioinformatics 2(3), 258–270 (2001)
2. Kolch, W., Calder, M., Gilbert, D.: When kinases meet mathematics: the systems biology of MAPK signalling. FEBS Lett. 579, 1891–1895 (2005)
3. Gilbert, D., Heiner, M., Lehrack, S.: A unifying framework for modelling and analysing biochemical pathways using Petri nets. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 200–216. Springer, Heidelberg (2007)
4. Vyshemirsky, V., Girolami, M.: Bayesian Ranking of Biochemical System Models. Bioinformatics 24(6), 833–839 (2008)
5. Maria, G.: A Review of Algorithms and Trends in Kinetic Model Identification for Chemical and Biochemical Systems. Chem. Biochem. Eng. Q. 18(3), 195–222 (2004)
6. Donaldson, R., Gilbert, D.: A Monte Carlo Model Checker for Probabilistic LTL with Numerical Constraints. Technical report, University of Glasgow, Department of Computing Science (2008)

7. Baier, C.: On Algorithmic Verification Methods for Probabilistic Systems. Habilitation thesis, University of Mannheim (1998)
8. Brightman, F., Fell, D.: Differential feedback regulation of the mapk cascade underlies the quantitative differences in egf and ngf signalling in pc12 cells. FEBS Lett. 482, 169–174 (2000)
9. Gilbert, D., Heiner, M., Rosser, S., Fulton, R., Gu, X., Trybilo, M.: A Case Study in Model-driven Synthetic Biology. In: Biologically Inspired Cooperative Computing: BICC 2008. IFIP, vol. 268, pp. 163–175. Springer, Heidelberg (2008)
10. Pnueli, A.: The Temporal Semantics of Concurrent Programs. Theor. Comput. Sci. 13, 45–60 (1981)
11. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (1999) (third printing, 2001)
12. Fages, F., Rizk, A.: On the Analysis of Numerical Data Time Series in Temporal Logic. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 48–63. Springer, Heidelberg (2007)
13. Hansson, H., Jonsson, B.: A Logic for Reasoning about Time and Reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
14. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.K.: Verifying Continuous-Time Markov Chains. In: Alur, R., Henzinger, T.A. (eds.) CAV 1996. LNCS, vol. 1102, pp. 269–276. Springer, Heidelberg (1996)
15. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
16. BioNessie: A Biochemical Pathway Simulation and Analysis Tool. University of Glasgow, `http://www.bionessie.org`
17. MC2(PLTLc) Website: MC2(PLTLc) - PLTLc model checker. University of Glasgow (2008), `http://www.brc.dcs.gla.ac.uk/software/mc2/`
18. Meffert, K., et al.: JGAP - Java Genetic Algorithms and Genetic Programming Package (2008), `http://jgap.sf.net/`
19. Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., et al.: The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. J. Bioinformatics 19, 524–531 (2003)
20. Fujarewicz, K., Kimmel, M., Lipniacki, T., Świerniak, A.: Parameter estimation for models of cell signaling pathways based on semi-quantitative data. In: BioMed 2006: Proceedings of the 24th IASTED international conference on Biomedical engineering, Anaheim, CA, USA, pp. 306–310. ACTA Press (2006)
21. Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine Learning Biochemical Networks from Temporal Logic Properties. In: Priami, C., Plotkin, G. (eds.) Transactions on Computational Systems Biology VI. LNCS (LNBI), vol. 4220, pp. 68–94. Springer, Heidelberg (2006)