# Chapter 1
# Model Checking Approach to the Analysis of Biological Systems

**Nikola Beneš, Luboš Brim, Samuel Pastva and David Šafránek**

**Abstract** Formal verification techniques together with other computer science formal methods have been recently tailored for applications to biological and biomedical systems. In contrast to traditional simulation-based approaches, model checking opens an entirely novel way of viewing and analysing the dynamics of such systems. In particular, it can help in system identification and parameter synthesis, in comparison of models with respect to a priori given desired properties, in robustness analysis of systems, in relating models to experimental data, or in globally analysing the bifurcations of systems behaviour with respect to changes in parameters. In this review, we briefly describe the state-of-the-art methods and techniques employing model checking, as one of the most prominent verification techniques, to the analysis of biomedical systems. We demonstrate some of the advantages of using the model checking method by presenting a brief account of the technique itself followed by examples of the application of formal methods based on model checking to three areas related to the analysis of biomedical systems: verification of biological hypotheses, parameters synthesis, and bifurcation analysis. Finally, we discuss several case studies that show how fruitfully the methods can be utilised within the computational systems biology and biomedicine domain.

N. Beneš · L. Brim · S. Pastva · D. Šafránek (✉)
Systems Biology Laboratory at Faculty of Informatics, Masaryk University,
Botanická 68a, 602 00 Brno, Czech Republic
e-mail: safranek@fi.muni.cz

N. Beneš
e-mail: xbenes3@fi.muni.cz

L. Brim
e-mail: brim@fi.muni.cz

S. Pastva
e-mail: xpastva@fi.muni.cz

P. Liò and P. Zuliani (eds.), *Automated Reasoning for Systems Biology and Medicine*, Computational Biology 30,
https://doi.org/10.1007/978-3-030-17297-8_1

## 1.1 Introduction

Biomedical systems are complex systems of interacting parts, which may be molecules, cells, organisms, or entire species that change their properties with time in response to external and internal stimuli. Studying the *dynamic behaviour (dynamics)* of these systems is the basis for the understanding of, e.g. cellular functions or disease mechanisms.

The computational analysis of the precise dynamics of biomedical systems involves the construction of appropriate *computational models*. Building suitable sound dynamic models can be regarded as an essential step in the development of predictive models for cells or whole organisms. While the structure of the models is mostly available, some of their quantitative features are often difficult to determine. These values, which may significantly affect the system dynamics, are represented in the model as *parameters*. To gain reliable models for making predictions, its parameters, such as concentrations or reaction rates, should be specified as precisely as possible. Some of the parameter values are already defined in the literature, or they can be inferred from experimental data. However, most of them are uncertain or unknown.

The application of formal approaches and computational methods, as discovered and developed in the context of computer science and engineering [54, 83], in systems biology can contribute to the development of powerful model-based reasoning, analysis, and simulation tools for biologists. These tools have the power to bring the necessary support in preparing new experiments for testing hypotheses and eventually, for a better understanding of emergent functional properties of cells, tissues or even organisms.

In recent years, the cooperation among biologists, mathematicians and computer scientists in the area of systems biology is extending and intensifying. The reason is that both the biological and the computational systems can be regarded as systems that rely on the interaction of its components known in computer science as communicating reactive systems. Therefore, many formal methods and approaches developed in computer science for modelling and analysis of reactive computational systems may apply to the biological ones as well.

Model checking (also termed automated formal verification) is one of them. It is a very appropriate and promising formal method that can be exploited in computational systems biology with the potential of bringing significant benefits in designing biological models. The reason is that model checking, as a verification technique, can be regarded in principle as an approach to confirm or refute biological hypotheses.

The development of formal verification techniques together with the power of the underlying computer hardware has made it possible to apply these methods to very complex systems. In this review, we briefly describe three examples of the application of model checking to the formal analysis of biological systems.

## 1.2 Verification by Model Checking

*Model checking* [5, 39] is a computer science verification method that grew up from purely academic research techniques to a well-accepted modern verification method routinely used in industry. Nowadays, model checking is widely considered as a complement and enhancement to the existing validation and verification techniques such as simulation and testing.

The principle of model checking as a software verification method is to systematically check whether a model of a given computer system satisfies a property such as deadlock freedom, invariants or request–response. The main advantage of model checking is that, when applied to restricted finite-state models—Kripke structures, it is a pure 'push-button' technology. The possibility to have a fully automatic method for software verification, as opposed to other verification methods, is substantial. It requires minimal human intervention (and less experience), applies to systems with realistic properties, and produces a counterexample in case of failure.

Model checking belongs to a broader class of formal methods that are termed *formal verification* methods. Although the introduction of formal verification to computer systems development is rather costly, it pays off after all. In particular, it is not only able to catch deeper flaws in the computer systems than testing or simulation; it often results in significant reduction in the verification time as well as the development costs and time-to-market.

Model checking is a computationally demanding and memory-intensive technique in general. Its applicability to large and complex systems as seen in practice is thus limited to some extent. The major problem is the *state space explosion problem* [41] due to which large industrial models cannot be efficiently handled, unless scalable and more sophisticated methods are used.

A great deal of attention has been paid to the development of approaches to fight the state space explosion problem in the field of automated formal verification [81]. The most prominent are state space reduction [38, 51, 82], compression [65], state compaction [58], bounded model checking [22], symbolic state space representation [33] and the use of parallel computers [29].

Model checking is primarily a verification technique which is, however, often used for falsification (as a bug hunting method). The core idea behind model checking is to exhaustively explore, using clever techniques, all states of the *finite-state system* (model). This either gives the guarantee the system is correct or presents a counterexample in the opposite case. It is this exploration feature of model checking that is often used to perform other kinds of analyses of (biological) models as we see later.

Model checking, as the term suggests, needs a suitable model of the system. Model checking cannot be applied directly to real systems which is sometimes considered as its drawback. In software engineering, this is not a critical issue as there are model-based approaches to system construction and building models is a natural part of the software life cycle. On the contrary, for biological systems, the model must be acquired from the knowledge about the real, already existing, system first. We will comment on biological models and how to turn them into models suitable for model checking later in Sect. 1.3.1. In this review, we will suppose the system is modelled operationally as a nondeterministic finite-state system with a set of transitions (*a transition system*), which define the changes of state as the system evolves. A sequence of successive transitions is called a *run* or an *execution*. Furthermore, for the model checking purposes, we usually extend the model by assigning atomic logical propositions to particular states. Such a model is in computer science termed a *Kripke structure*.

The second, and equally important, ingredient for the model checking procedure is a *temporal logic formula* expressing the desired property of the system's behaviour (its dynamics). Computer science offers two main types of logical formalisms for expressing qualitative properties of systems dynamics (see, e.g. [5]): linear-time temporal logics interpreted over individual model executions (runs), and branching-time temporal logics, interpreted over trees of (nondeterministically) branching model executions. The simplest linear-time temporal logic is linear temporal logic (LTL) and the basic branching-time logic is CTL. Both LTL and CTL can be interpreted on many kinds of models. Both logics extend the classical propositional logic by temporal operators. The temporal operators in LTL are the **F** operator expressing that a property will be true in the *future* (eventually), the **G** operator expressing that a property will be true invariantly (*globally*) in all states of the execution, **X** expressing that a property will be true in the ne*X*t state of the execution, and the **U** operator expressing that one property is true *until* another property is true. The logic CTL adds to LTL 'quantifiers' **A** and **E** over runs with with a common state (computation trees). For example, the formula **AF** expresses that on *all* executions from here the property is true sometime in the future, while the formula **EG** expresses that on *some* execution from here the property is always true. The individual temporal logics differ in the selection of temporal operators and the semantic model used. There are also numerous extensions of these logics in order to increase their expressive power. In particular, a hybrid extension is very useful for describing some typical phenomena found in biology. We give several additional examples of properties taken from biology and their reformulation as formulae in Sect. 1.3.1.

## 1.3 Methods and Tools

The development of methods, techniques, and tools is not yet at a level where the formal analysis of an entire complex biological or biomedical system is possible. Nevertheless, we can already handle interesting and challenging fragments of such systems.

## *1.3.1 Model Checking Biological Systems*

The formalism used to define a model of a biological system is essential, since it not only dictates the possible behaviours that may or may not be captured but it also determines the computational means for detecting them and subsequently to perform an effective model calibration. Fisher and Henzinger [54] distinguish two kinds of models in systems biology: *operational* (also termed executable or computational) versus denotational (also termed mathematical). The *operational models* (such as Continuous time Markov chains, Petri nets, or Process algebras) are executable and mimic system *processes*. The *denotational models* (such as differential or difference equations) express mathematical relationships between quantities and how they change over time. These models are in general quantitative and tend to require a lot of computational power to simulate. In this review, we stick with computational models in the form of Kripke structures. All computational models can, at least in principle, be used instead of Kripke structures or their extensions. We prefer Kripke structures for their simplicity and generality.

Models in systems biology are rarely presented as Kripke structures. To be able to apply formal verification methods to a model, it has to be first converted into a Kripke structure. We omit the details about the various conversion methods here; for details see, e.g. [28].

Models can also be classified as deterministic or stochastic (or hybrid). On the one hand, *deterministic models* such as those based on ordinary differential equations (ODE) typically enable the analysis of large collections of molecules in a population. This is because they abstract from individualistic properties of each molecule, such as position or its stochastic behaviour, and take into account concentrations of each species as its variables only. On the other hand, *stochastic models* such as CTMCs (Continuous-time Markov chains) abstract from positions of molecules but maintain their individual interactions. Stochastic models overcome some of the inherent limitations of deterministic models typically at the price of higher computational complexity. We do not cover stochastic models in this review, and we refer the interested reader again to [28].

### 1.3.1.1 Temporal Properties of Biological Systems

Let us now turn our attention to properties and their formulation in temporal logics. Concerning the phenomena appearing in the dynamics of biological processes, there are several classes of properties that are typically studied on biological models. In particular, they can be organised into the following six categories: reachability properties, temporal ordering of events, variable correlations, (multi)stability properties, monotonic trends and oscillation properties.

For the demonstration purposes, suppose the states of our model incorporate information about concentration levels of some ingredients represented as system variables. *Reachability properties* express 'reachability of specified concentration levels in given model variables'. An example of a typical reachability property is the following requirement '$C$ reaches the concentration level ranging between 3.1 and 3.3 at some phase of the dynamics'. Such a property can be encoded by the formula $\mathbf{F}(3.1 \leq B \leq 3.3)$. The formula is based on the fundamental linear operator **F** (Future). It has the intuitive meaning that can be phrased in the following way 'on a given execution, there must eventually exist a state where the subformula is satisfied'. Is it worth noting that the property does not address the moment at which the event occurs (it is a qualitative property). The applicability of reachability properties is mainly seen in expressing global bounds on the reachable concentration of given substances.

For capturing the qualitative patterns of temporal behaviour observed in dynamics of given variables, the typically used properties target the phenomenon of *temporal ordering of events*. linear-time operator **U** (Until), i.e. the formula $\varphi_1 \, \mathbf{U} \, \varphi_2$, with an intuitive meaning that, for a given execution, $\varphi_2$ must eventually hold in some $i$th state of the execution and for all states from the beginning of the execution until the $i$th state, $\varphi_1$ must hold. An example of such property is the formula $(A \leq 3) \, \mathbf{U} \, [(3 < A \leq 15) \, \mathbf{U} \, (A > 15)]$ representing the following temporal pattern: species $A$ is initially kept below 3 until it reaches 15 and finally exceeds 15.

*Variable correlations* can provide important observations revealing cooperations and dependencies in biological processes, e.g. co-expression of certain genes. We can express such properties by combining several temporal ordering formulae into a single formula using the traditional propositional operators. In this way, mutual dependencies in the dynamics of inspected variables can be captured. For example, the formula $[(A \leq 3) \, \mathbf{U} \, ((3 < A \leq 15) \, \mathbf{U} \, (A > 15))] \Rightarrow [(C \geq 11) \, \mathbf{U} \, ((5 \leq C < 11) \, \mathbf{U} \, (C < 5))]$ expresses the following correlation in concentrations of species $A$ and $C$: if $A$ increases according to the temporal pattern from the previous paragraph then $C$ decreases from a level above 11 to a level below 5.

The analysis of the presence of stable concentration levels calls for using a specific kind of temporal properties. An example of an elementary *stability property* is the formula $\mathbf{G}(A \leq 3)$ stating that the concentration below 3 is stable (attractor) for species $A$. The formula $\mathbf{G}\varphi$, with the operator **G** (Globally), expresses the requirement that $\varphi$ must be invariantly true in each state of a given execution; its intuitive meaning is 'forever'. Stability properties can be combined with reachability properties and related to a specific initial condition. For example, the formula $(A \geq 0) \Rightarrow \mathbf{FG}(5 < A \leq 12)$ states that the stable concentration between 5 and 12 is reached from any non-negative initial concentration of $A$. The LTL formula $[(A \leq 5) \Rightarrow \mathbf{G}(A \leq 5)] \wedge [(A > 5) \Rightarrow \mathbf{G}(A > 5)]$ can be used to express the existence of several different stable states (multi-stability). In this case, the formula expresses the fact that there are two different stable concentration levels in the dynamics of $A$: the first one is below the level 5 and the second one is above 5. Note that this formula states the existence of the two stable attractors only, there is nothing specified with respect to reachability of both stable attractors from a particular part of the state space (the so-called basin of attraction). To formulate this kind of properties, CTL has to be employed: $\mathbf{EFAG}(A \leq 5) \wedge \mathbf{EFAG}(A \geq 5)$. The branching-time operator $\mathbf{EF}\varphi$ requires the existence of a branch where $\varphi$ is eventually satisfied, whereas $\mathbf{AG}\varphi$

requires $\varphi$ to be true in all future states. Therefore, the bi-stability formula holds in every state from which the execution can eventually branch into both attractors.

Another important dynamics property appearing in biological systems is *oscillation*, e.g. circadian rhythms. For example, the formula $(\mathbf{G}[(A \leq 3) \Rightarrow \mathbf{F}(A > 3)]) \wedge (\mathbf{G}[(A > 3) \Rightarrow \mathbf{F}(A \leq 3)])$ represents a permanent oscillation of $A$ around the concentration level 3. To express oscillation properties, we have to use linear-time operators; oscillation properties cannot be sufficiently well expressed in CTL. Finer specification of oscillations (e.g. the maximal and minimal amplitude levels) can be realised by adding additional constraints identifying the qualitative aspects of the oscillation to the formula.

To formalise biological phenomena in temporal logics, we often have to consider extensions of existing logics. The biologically relevant extensions target precise quantitative description of oscillations [9, 45] or qualitative properties combining linear-time properties with branching-time [75]. In the domain of branching-time logics, our own work brings a unique combination of two known extensions of CTL– an extension HCTL adding hybrid operators including past operators and allowing to use of state variables that can be fixed in certain parts of the formula as well as quantified [2], and an extension UCTL adding event predicates over single-step system evolutions [16]. The resulting logic called HUCTL [20] allows to efficiently express global and local properties of phase spaces of dynamical systems that cannot be expressed in LTL/CTL, e.g. the presence of a given number of mutually exclusive stable attractors. The need for hybrid branching-time logics in the domain of biological systems has been also addressed in [3].

The examples mentioned above presented an intuition behind the linear and branching-time temporal logic as a formalism used for expressing properties of biological systems. In next paragraphs, we give the full syntax and semantics of the two considered temporal logics.

#### 1.3.1.2 Linear Temporal Logic

LTL captures temporal properties of paths in discrete state-transition systems. In particular, LTL formulae are interpreted on infinite paths generated by a Kripke structure.

A Kripke structure $\mathcal{K}$ is defined as a tuple $\mathcal{K} = (S, S_0, \rightarrow, L)$ where $S$ is a set of states, $S_0 \subseteq S$ is a set of initial states (representing all initial conditions considered in a particular analysis task), $\rightarrow \subseteq S \times S$ is the transition relation, and $L : S \rightarrow 2^{AP}$ is a mapping (labelling) that assigns atomic propositions from some set $AP$ to states. The meaning of a labelling is to annotate states with attributes that are supposed to be satisfied in the respective states.

LTL formulae are defined by the following abstract syntax:

$$\varphi ::= Q \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{X}\varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

where $Q$ ranges over atomic propositions taken from a set $AP$. We use the standard abbreviations like $\mathbf{F}\varphi$ which stands for $true\mathbf{U}\varphi$ or $\mathbf{G}\varphi$ which stands for $\neg\mathbf{F}\neg\varphi$.

The semantics of an LTL formula $\varphi$ is interpreted on infinite paths of a Kripke structure $\mathcal{K} = (S, S_0, \rightarrow, L)$. For an infinite path $\pi = s_0 s_1 \ldots$ and some $i \in \mathbb{N}_0$, we use the notation $\pi^i$ to denote the infinite path $\pi^i = s_i s_{i+1} \ldots$ and the notation $\pi(i)$ to denote the state $s_i$.

$$\begin{array}{ll}
\pi \models_{\mathcal{K}} Q & \text{iff } Q \in L(\pi(0)) \\
\pi \models_{\mathcal{K}} \neg\varphi & \text{iff } \pi \not\models_{\mathcal{K}} \varphi \\
\pi \models_{\mathcal{K}} \varphi_1 \wedge \varphi_2 & \text{iff } \pi \models_{\mathcal{K}} \varphi_1 \text{ and } \pi \models_{\mathcal{K}} \varphi_2 \\
\pi \models_{\mathcal{K}} \mathbf{X}[\varphi] & \text{iff } \pi^1 \models_{\mathcal{K}} \varphi \\
\pi \models_{\mathcal{K}} \varphi_1 \mathbf{U} \varphi_2 & \text{iff } \exists i \in \mathbb{N}_0 \text{ such that } \pi^i \models \varphi_2 \text{ and } \forall j < i . \pi^j \models \varphi_1
\end{array}$$

We say a Kripke structure $\mathcal{K}$ satisfies $\varphi$ iff every path starting in some initial state satisfies $\varphi$. This is called the universal interpretation of LTL.

The patterns of some typical LTL formulae covering the most usual classes of properties given in Sect. 1.3.1.1 are the following:

- $\mathbf{F}[\varphi]$ expresses a reachability of a state where the condition $\varphi$ holds,
- $\mathbf{G}[\varphi]$ expresses a stabilisation with $\varphi$ being continually true,
- $[\varphi_1 \Rightarrow \mathbf{F}\varphi_2] \wedge [\varphi_2 \Rightarrow \mathbf{F}\varphi_1]$ expresses a permanent oscillation between $\varphi_1$ and $\varphi_2$.

#### 1.3.1.3 Computation Tree Logic

The key characteristics of CTL is that it captures the branching behaviour of discrete state-transition systems. More precisely, CTL formulae are interpreted on states of a Kripke structure.

CTL formulae are defined by the following abstract syntax:

$$\varphi ::= Q \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{AX}\varphi \mid \mathbf{EX}\varphi \mid \mathbf{A}(\varphi_1 \mathbf{U} \varphi_2) \mid \mathbf{E}(\varphi_1 \mathbf{U} \varphi_2)$$

where $Q$ ranges over atomic propositions taken from a set $AP$. It is worth mentioning the standard abbreviations like $\mathbf{EF}\varphi$ which stands for $\mathbf{E}(true\,\mathbf{U}\,\varphi)$ or $\mathbf{AG}\varphi$ which stands for $\neg\mathbf{EF}\,\neg\varphi$.

The semantics of a CTL formula $\varphi$ is interpreted on Kripke structures; in particular, for every state $s \in S$ and a particular form of the formula the semantics is given as follows:

$$\begin{array}{ll}
s \models_{\mathcal{K}} Q & \text{iff } Q \in L(s) \\
s \models_{\mathcal{K}} \neg\varphi & \text{iff } s \not\models_{\mathcal{K}} \varphi \\
s \models_{\mathcal{K}} \varphi_1 \wedge \varphi_2 & \text{iff } s \models_{\mathcal{K}} \varphi_1 \text{ and } s \models_{\mathcal{K}} \varphi_2 \\
s \models_{\mathcal{K}} \mathbf{AX}[\varphi] & \text{iff for all } \pi \text{ in } \mathcal{K} \text{ such that } \pi(0) = s \text{ it holds that } \pi(1) \models_{\mathcal{K}} \varphi \\
s \models_{\mathcal{K}} \mathbf{EX}[\varphi] & \text{iff there exists } \pi \text{ in } \mathcal{K} \text{ such that } \pi(0) = s \text{ and } \pi(1) \models_{\mathcal{K}} \varphi \\
s \models_{\mathcal{K}} \mathbf{A}(\varphi_1 \,\mathbf{U}\, \varphi_2) & \text{iff for every } \pi \text{ in } \mathcal{K} \text{ such that } \pi(0) = s \text{ there exists} \\
 & \quad i \in \mathbb{N}_0 \text{ such that } \pi(i) \models \varphi_2 \text{ and } \forall j < i . \pi(j) \models \varphi_1
\end{array}$$

$s \models_{\mathcal{K}} \mathbf{E}(\varphi_1 \,\mathbf{U}\, \varphi_2)$ iff there exists $\pi$ in $\mathcal{K}$ such that $\pi(0) = s$ and
$\exists i \in \mathbb{N}_0$ such that $\pi(i) \models \varphi_2$ and $\forall j < i.\pi(j) \models \varphi_1$

We say a Kripke structure $\mathcal{K}$ satisfies $\varphi$ iff for all $s \in S_0$, $s \models_{\mathcal{K}} \varphi$.

The following examples show typical CTL formulae used for biological systems as discussed above:

- $\mathbf{EF}[\varphi]$ expresses a reachability of a state where the condition $\varphi$ holds,
- $\mathbf{AG}[\varphi]$ expresses a stabilisation with $\varphi$ ($\varphi$ is continually true),
- $\mathbf{EF}[\mathbf{AG}[\varphi_1]] \wedge \mathbf{EF}[\mathbf{AG}[\varphi_2]]$ expresses a bistable switch (two different stable situations $\varphi_1, \varphi_2$ can be reached).

#### 1.3.1.4 Model Checking and Monitoring Methods

Model checking based techniques for the analysis of biological and biomedical systems can be roughly split into *monitoring (simulation) techniques* and *exhaustive (verification) techniques*. Both classes of techniques make a crucial tool for automatised analysis of engineered systems. The exhaustive techniques check whether all executions—state-event sequences—generated by a given system $\mathcal{S}$, satisfy the inspected property described as a formula $\varphi$ in a suitable logic. To generate all executions, the whole state space must be (at least in principle) stored and evaluated. For systems which cannot be handled by exhaustive techniques, either due to the presence of continuous and/or unbounded values or simply due to the state space explosion problem, the monitoring techniques are the only feasible validation method. Unlike model checking, the monitoring techniques check an individual execution. The key rationale behind the efficiency of monitoring is that for large and complex systems, the simulation is generally easier and faster than building a concise representation of a whole system required for the exhaustive model checking. However, since a single simulation generates one trajectory out of all the possible ones of a system, usually the average values among several simulations must be considered for achieving the necessary level of confidence in the results obtained.

The accuracy of monitoring techniques can be improved by employing the (Bayesian) statistical model checking. The method addresses general stochastic systems in terms of (Bayesian) statistical inference [66, 69]. First, it samples the behaviours (simulations) of a model. Second, it verifies their validity with respect to a temporal formula (i.e. performs the membership test). Finally, it applies a statistical estimation technique to compute an approximate value for the probability that the formula is satisfied.

It is worth noting that the monitoring procedure can be also applied to experimentally measured time-series data. Monitoring then enables methods of automatised inference of logical specification of system's dynamics from experimental data [79].

Exhaustive model checking, statistical model checking, and monitoring techniques have been applied to the analysis of models of biological systems. They provide researchers with means to make predictions and test their hypotheses for systems of different kinds 'in silico'. For continuous-time models, the exhaustive techniques cannot be used directly due to the infinite number of possible states. Therefore, more sophisticated monitoring techniques for many temporal logics have been developed to analyse complex nonlinear systems. A comprehensive survey is available in [74]. These approaches have been further extended for application in systems biology. The Breach tool [47] makes a good example. The tool provides a coherent set of simulation-based techniques for the analysis and parameter identification/synthesis of deterministic continuous-time models of complex biological systems studied in biology. The main features of the tool facilitate the computation and the property investigation of large sets of trajectories. Additionally, it also provides information about the sensitivity with respect to parameter perturbations. A successful application of this tool in systems biology has been demonstrated in [49] where a model of the acute inflammatory response to bacterial infection is analysed.

Another, more or less similar, extension to the monitoring techniques has been proposed in [53]. The authors present a generalisation of a constraint-solving trace-based model checking algorithm [34] for the quantifier-free fragment of the first-order extension of linear temporal logic QFLTL($\mathbb{R}$) with numerical constraints over the real numbers. Given an ODE model and a temporal property to be verified within a finite time horizon, the computation of a finite simulation trace by numerical integration provides a linear Kripke structure—a structure where each state has a single successor. The generalisation provided by QFLTL($\mathbb{R}$) gives the ability to compute those instantiations of a formula that are true in a finite trace. This is realised by giving the complete domain of the real-valued variables occurring in the formula for which it is true. The BIOCHAM [52] tool implements this approach.

The main drawback of statistical model checking is handling rare events; in particular, the temporal formulae that characterise behaviour with a tiny probability of occurrence. To estimate the probability of such formulae, the number of required simulations for ensuring a good estimate becomes impractical. In [40], the importance sampling is addressed. More specifically, a variance-reduction technique is presented for the Monte Carlo method, and the cross-entropy method, a general Monte Carlo approach to combinatorial and continuous multi-extremal optimisation. The bounded linear temporal logic is employed here; it is a variant of LTL that enhances temporal operators with time bounds.

Besides the utilisation of Bayesian inference [66], there have been recently developed techniques employing moment-closures [1, 4, 27] (a robust implementation of some of these methods is available in the tool U-check [26]). Additionally, methods based on a correct deterministic approximation of the transient distribution can also be employed [25].

Several model checking tools have been developed implementing exact and approximate techniques, e.g. PRISM [68], MARCIE [92]. Examples of applications to analysis of biological systems are, e.g. [63] where the authors apply PRISM to analyse the complex FGF (Fibroblast Growth Factor) signalling pathway and [91] where the authors analyse stochastic Petri nets using efficient state space representation based on interval decision diagrams. The prototype tool SABRE [44] implements techniques for exact CSL model checking that allow to reduce the state space explosion problem for some classes of biological systems.

In the case of real-time models, model checking requires to transform the uncountable continuous-time model into an equivalent finite discrete structure (the so-called zone automaton). The main tools targeting real-time models are UPPAAL [17] and KRONOS [96]. They have also been used for the analysis of biological models. In the case of UPPAAL, there are applications to gene regulatory networks [60, 93] and signalling pathways [89]. The latter case study has lead to a novel tool ANIMO adapting the technology to the specific format of biological models [90]. KRONOS was applied to real-time abstractions of continuous-time deterministic models [73] and to gene regulatory networks [14].

At the end of this section, we give a brief overview of applications of model checking techniques to analysis of biological systems. Model checking is highly relevant for Boolean models of genetic regulatory networks [21, 36] and signalling networks [50, 88], provided that symbolic verification techniques can be employed. The tool BIOCHAM [34] allows verification of qualitative CTL properties of asynchronous qualitative models with Boolean semantics using the standard symbolic model checker NuSMV [37].

Explicit model checking is employed in [67] where the authors propose a new methodology for parameter synthesis of discrete gene networks based on coloured LTL model checking [6].

In [64], the authors present a methodology adapting Petri nets to qualitative and quantitative analysis of biological systems including examples demonstrated on a mitogen-activated kinase cascade. The authors study structural properties (reflecting the modelling approach), properties specified in temporal logic and general properties studied on Petri nets (boundedness, liveness, reversibility, invariants).

The tool BioDiVinE [7] provides techniques for finite discrete abstraction of the continuous state space and that way allows to analyse biological models specified in terms of a set of chemical reactions. First, dynamics of chemical reactions is represented by means of multi-affine differential equations. Second, the multi-affine system is further discretised to a finite-state-transition system in order to employ the standard LTL model checking techniques including parameter synthesis with respect to LTL properties.

### *1.3.2 Parameter Synthesis for Dynamical Systems*

The construction of models describing the dynamics of a biochemical process is one of the most critical tasks to be done when we want to improve the understanding of existing or not yet discovered behavioural phenomena and physiological phenotypes occurring in biology. The model-based prediction and analysis are traditional approaches that form the cornerstones of systems biology. With computational models, we can also count on new and very efficient computer-aided formal analysis techniques, that have not only the potential to speed up the analysis itself, but they can even provide new and unexpected results. In many cases, the structure of a dynamical model capturing the hypothesis about some biological process is already available at the qualitative level represented by known entities and their mutual interactions. However, most of the quantitative aspects related to the systems dynamics, such as initial concentrations or reaction rates, are either unknown or cannot be readily determined. These quantitative attributes of interactions are usually reflected in the model as *parameters*. To design a reliable model, its parameters must be specified as precisely as possible. Typically, a small fraction of the parameter values can be determined from the literature or experimental data while leaving many parameter values uncertain or unknown entirely. The reason is that many parameters are hard to measure *in vitro*/*in vivo*.

The *algorithmic synthesis* of unknown parameter values (also referred to as *parameter discovery*, *parameter estimation*, *parameter identification* or the *inverse problem*) remains thus one of the most challenging problems in computational systems biology. As an alternative to the traditional approaches solving this problem (e.g. [55, 56, 59, 85]), there have recently appeared completely new techniques grounded in formal verification [6, 11, 71]. These methods and techniques typically focus on synthesising subsets of parameter space instead of finding unique parameter values. The parameter synthesis procedure computes, for a given *parametrised model* and a desired *temporal property*, the maximal set of parametrisations satisfying the property. The overall setting of the property-driven parameter synthesis is depicted in Fig. 1.1. The hypotheses obtained from literature as well as time-series experiments data conducted in wet-labs can be considered as temporal properties restricting the admissible set of model parameter values.

The main advantage of using a temporal logic specification for parameter synthesis is in the ability to focus on certain precisely specified qualitative aspects of observed behaviour [78]. A good example is a temporal ordering of events qualita-

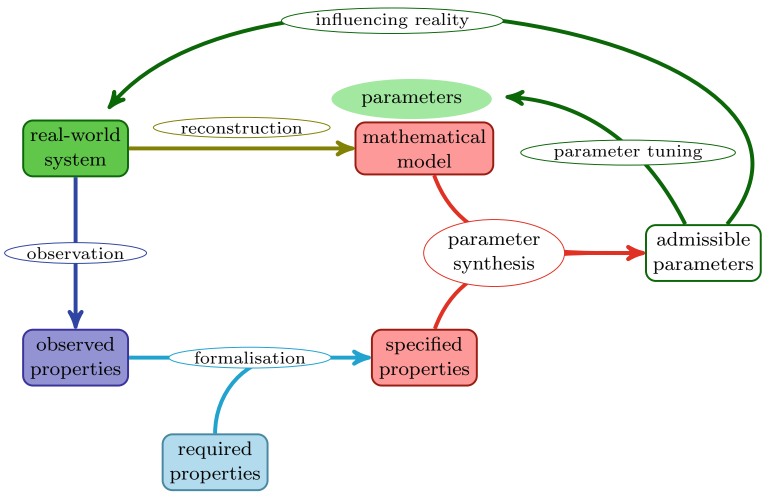


**Fig. 1.1** A general scheme of parameter synthesis methods based on system properties formalised in a temporal logic

tively characterising order of important moments in the systems dynamics. Temporal properties also have the power to express *global properties* independent of the specific set of initial conditions. The tool thus can, for a given model and a property, compute the maximal set of parameter values and initial conditions for which the model entirely fulfils the property. Such an approach is complementary to traditional methods based on monitoring a numerical simulation [15, 46] or local sensitivity analysis [48]. To express required biologically relevant temporal hypotheses, both the branching-time and the linear-time operators are needed [13].

It is worth noting that there are two levels of complexity that significantly affect the tractability of parameter synthesis for biological models. First, the procedure needs to be exhaustive in terms of considering all possible settings of parameters—points in the parameter space. The size of the parameter space increases exponentially with the number of unknown parameters. However, to achieve practically usable results, the number of parameters to be considered has to be small. A model with too many parameters is hard to falsify—it can fit almost any data. Second, during every model checking step—analysis of the model with a particular parameter setting—the dynamics of the model has to be explored. More precisely, the state space of the model, which increases exponentially with the number of state variables, has to be examined for every considered parametrisation.

Several techniques have been developed for parameter synthesis of continuous-time and discrete-time dynamical systems. In the case of linear-time temporal logic, the dominating approach is based on monitoring simulated trajectories of the system [8, 31, 48, 84, 86]. These techniques use numerical solvers working on systems with fixed parameters or with small parameter spaces (perturbations). An advantage of these techniques is that the function defining the systems dynamics is considered as a black box provided that there is almost no limitation on the form of the system parametrisation. The main disadvantage is the need to sample the parameter space and initial states while losing robust guarantees on the results. This problem can be overcome by replacing numerical solvers with Satisfiability Modulo Theories (SMT) solvers that cope well with nonlinear functions and real domains up to required precision [57]. However, such symbolic techniques are limited to reachability analysis [72]. In particular, their extension to general temporal logic specifications is a non-trivial task yet to be explored. In [43], the authors show that exploration (sampling) of the parameter space can be improved with sensitivity analysis.

In this chapter, we focus on robust property-driven parameter synthesis techniques that instead of sampling the parameter values synthesise an exact (symbolic) representation of the maximal set of parametrisations satisfying a given temporal logic property. These techniques are based on model checking performed directly on a qualitative finite quotient or hybridisation of systems dynamics (e.g. [6, 12, 15, 23, 32]). Several approaches encode parameter sets symbolically in terms of polytopes [15, 61]. Another solution is to encode parametrisation sets utilising predicate formulae with nonlinear arithmetic over real numbers and use SMT to reason on them. In this chapter, we focus on this one. The reason is that the SMT-based solution is sufficiently general in terms of expressiveness of the supported parametrised

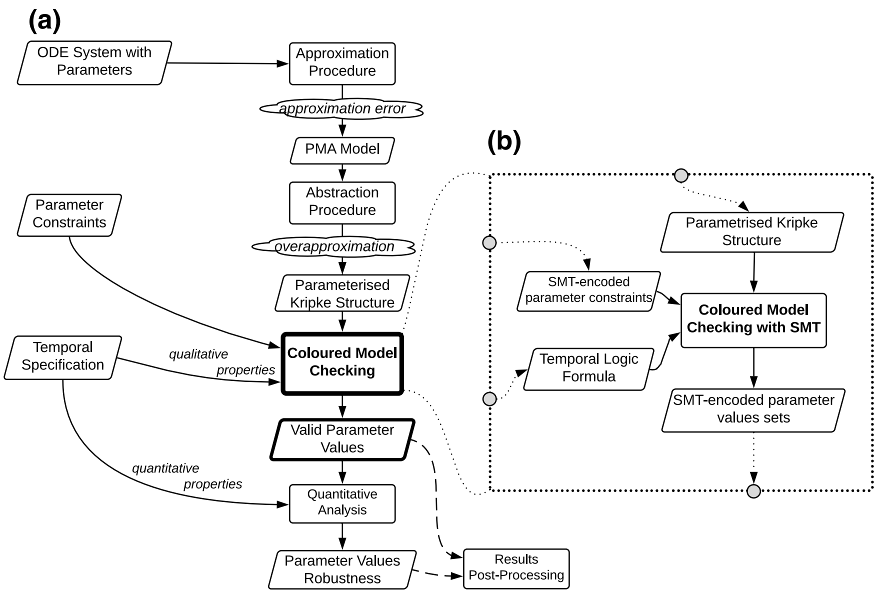


**Fig. 1.2** Parameter synthesis workflow based on coloured model checking technology (**a**). Detailed scheme of the coloured model checking procedure (**b**)

models and can be even improved in future with the progress of SMT state-of-the-art techniques.

To target the needs for efficient and scalable algorithmic base for parameter synthesis, we have developed the *coloured model checking* technique that extends the enumerative model checking framework to parametrised dynamical systems [19]. The intuitive idea is to symbolically execute the system runs over the unknown parameters while model checking the behaviour with respect to a given property. During the execution, a set of parametrisations enabling the satisfying behaviour is synthesised by constructing an appropriate SMT formula encoding the set.

The overall procedure of parameter synthesis based on model checking consists of several tasks. The input of the method is a set of *behaviour constraints* specifying requirements on the systems dynamics, a set of *parameter constraints* collecting all *a priori* known restrictions, dependencies and correlations of individual parameter values. The particular workflow depends on the kind of temporal logic employed and the kind of models considered. In Fig. 1.2a, there is depicted the general scenario with ODE models and CTL. In such a case, the parametrised ODE model has to be discretised into a parametrised Kripke structure. The important step is thus the appropriate abstraction procedure. Some classes of nonlinear ODE models can be approximated by a piecewise multi-affine ODE model (PMA) or a piece-wise affine ODE model (PAA). To that end, techniques based on linear optimisation are employed [61]. Consequently, a suitable abstraction technique is applied to obtain an over-approximation of the original system dynamics [11, 18]. The abstracted model is translated into the form of parametrised Kripke structure [42] that makes the input to the coloured model checking procedure for CTL [32]. The coloured model checking procedure can efficiently utilise satisfiability modulo theories for encoding compactly the (possibly infinite) sets of parameter values enabling a given transition in the Kripke structure (Fig. 1.2b).

The output of the procedure is for every state of the system a complex SMT formula encoding the set of parameter values satisfying the given temporal property in that particular state. Coloured model checking gives thus entirely *global* results that cover all states of the discretised system. The output can be further post-processed. First, optimisation tools based on SMT such as Symba [70] can be employed to find parameter values optimal with respect to a given objective function. Second, sampling or statistical model checking can provide a detailed exploration of the valid parameter space including quantitative measures such as satisfaction or robustness degree [86]. All the steps starting from approximation, abstraction and finally the parameter synthesis are fully automatised. The only input required from the user in addition to the models and constraints are the appropriate settings of the approximation/abstraction steps.

The approximation and abstraction steps can be realised automatically by techniques introduced in [11, 61]. In particular, each nonlinear function (i.e. Hill kinetics, Michaelis–Menten kinetics, or their variants) appearing in the right-hand side of the model equations is replaced with a sum of piecewise affine or piecewise multi-affine functions optimally fitting the original kinetic function. As a result of the abstraction, a parametrised direction transition system (PDTS) is obtained. This PDTS exactly over-approximates the PAA/PMA model. The details on how the models can be approximated and abstracted by means of PDTSs are described in [30]. The main principle of the abstraction is shown in Fig. 1.3. Additionally, the transitions are naturally labelled by an up- (resp. down-) arrow expressing the change in a particular model variable.

### *1.3.3 Digital Bifurcation Analysis*

The goal of the classical bifurcation theory is to study qualitative changes to the properties of a parameter-dependent system as its parameters are varied. Even a tiny change in one or more of parameter values can namely have a significant effect on the entire system dynamics. It is clear that getting a deeper insight into these qualitative structure changes of flow fields is of great importance for better understanding the general system behaviour.

The main questions to be answered when studying these behaviours are: what is the range of the parameter values over which a particular behaviour exists, for which values the system changes its behaviour qualitatively, what are the different behavioural patterns and what are the mechanisms of transition between them.

To analyse a system behaviour concerning a given domain of parameter values, traditional dynamical systems theory provides the apparatus called *bifurcation anal-*

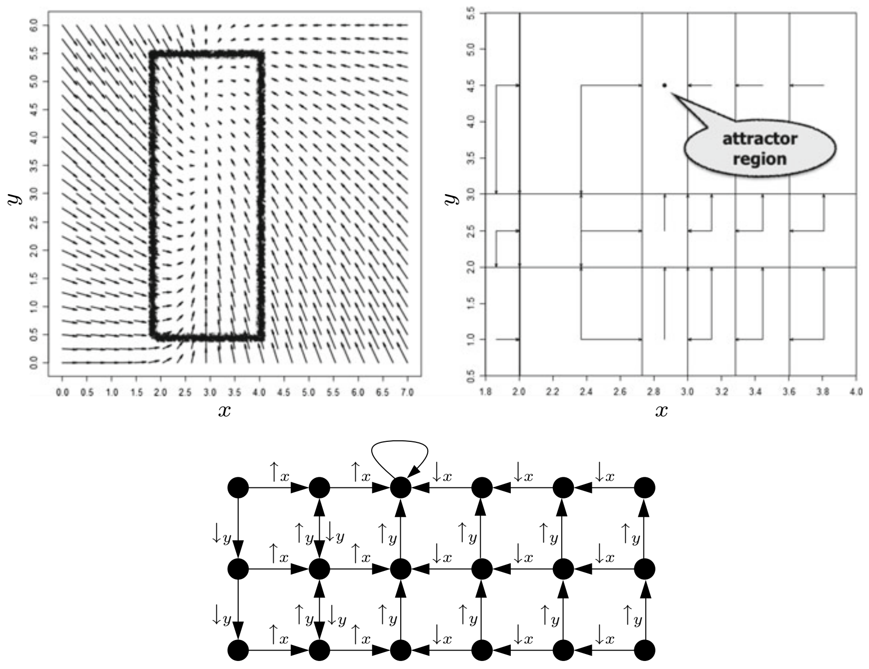


**Fig. 1.3** (Top left) the vector field of a PMA model. Ttop right) the discretisation of the emphasised part of the vector field highlighting the attractor (stable equilibrium) of the system's dynamics. The partitioning determining the rectangles was computed by the algorithm in [61]. (Bottom) the state-transition system (Kripke structure) corresponding to the discretisation. The arrows in the rectangles display the directions of changes of the particular value of the model variable

*ysis* [35] that allows to identify the so-called *bifurcation topologies* that organise typical asymptotic solutions (*attractors*) such as multiple steady states, limit cycles, etc. Its goal is primarily to compute the bifurcation diagrams (parameter space maps) that split the parameter space into areas for which the parameters do not affect the system's behaviour qualitatively, e.g. the structure and quality of attractors remain unchanged. Bifurcations occur at particular parameter values (so-called *bifurcation points*) that lie on the borders of these areas.

A common disadvantage of these traditional methods for bifurcation analysis is that they require mathematical skills, especially they need sophisticated results from numerical linear algebra and, as a result, their application is complicated and limited. Furthermore, another, in some sense even more severe, problem of existing methods (analytical [80] and numerical [76]) is their limitation in scalability. This is given by a relatively small number of parameters that can be handled and the impossibility to sufficiently automatise the analysis. In spite of these problems, bifurcation analysis is still a unique technique for the analysis of systems with parameters that gives a global understanding of the relationship between significant phases of systems dynamics and parameters.

Inspired by the classical bifurcation theory for dynamical systems, we have developed a novel approach to bifurcation analysis that is based on model checking and called the *digital bifurcation analysis*. The digital bifurcation analysis works on a discrete finite *abstraction* of the original continuous model. The method is, unlike mathematical methods, fully automatic and does not need any mathematical skills to be utilised. Another significant advantage is that the method is scalable to state spaces with tens of variables and high co-dimension of bifurcation (tens, possibly dependent, parameters), overcoming thus significantly the limits of the traditional mathematical methods. Last but not least the approach is advantageous in performing the global bifurcation analysis which is generally hard to compute by classical methods. The expressiveness regarding particular types of bifurcations, however, relies on the precision of the original systems phase-space discretisation/abstraction and on the logic chosen to describe various kinds of bifurcation.

The bifurcation analysis builds on so-called phase plane portraits of a dynamical system which we can understand as a division of the state space according to properties of trajectories. The phase portraits are typically presented in a graphical form, and they visualise how the solution of the system behaves in the long-term run. A phase portrait can reveal substantial information about the behaviour of a dynamical system. The individual parts of a phase portrait are called *portrait patterns*. A phase portrait can, for example, disclose the number and types of asymptotic equilibria like cycles or fixed points. Since it is practically impossible to graphically visualise all individual runs, only several key runs are depicted in the pictures to present phase portraits schematically.

Digital bifurcation analysis supposes the system is *abstracted* into a discrete structure (Kripke structure) in which transitions represent state changes and are in addition assigned with symbols representing directions of flow. A *run* in a Kripke structure is a sequence of transitions that change directions in individual states. In such a way, continuous trajectories in the original continuous system are abstracted into discrete runs in the Kripke structure preserving the direction of flow in the original system. Note that the size of the vectors in the original vector field is abstracted away. The phase portraits are represented in the Kripke structure as discrete counterparts of the original phase portraits and are called the *discrete phase portraits*. The patterns of the discrete phase portraits can be characterised by temporal logic formulae that take into account changes in directions.

It is worth noting that here we focus on *discrete* and even finite space systems. In continuous-time dynamical systems, a phase portrait plots both the position and momentum variables as a function of time to set up the ‘field’ that gives structure to the phase portrait. However, as already stated above, in our discrete system abstraction, we generally do not have the same kind of momentum. In our approach we simplify the velocity vectors that are put together to make a phase portrait of the continuous-time system by ‘vectors’ all having the same size. Bifurcation analysis is thus interpreted relatively with respect to the particular discrete abstraction procedure.

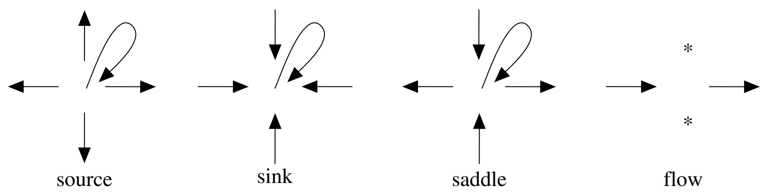


**Fig. 1.4** Examples of single-state portrait patterns. The symbol '$*$' stands for the following situations: incoming only, outgoing only, both or none. The patterns representing a saddle and a flow can be rotated along a particular axis, resulting in additional examples

The continuous phase portraits are formed from phase patterns that can have various shapes. The typical shapes that can occur in a phase portrait of a piecewise-affine system in a plane are the following:

- *sink*: a point into which all nearby trajectories flow;
- *source*: a point away from which all nearby trajectories flow;
- *stable spiral*: a point to which trajectories converge in a spiral;
- *unstable spiral*: a point near which trajectories diverge out in a spiral;
- *centre*: an infinite number of orbits; and
- *saddle*: a point near which two trajectories flow in, two flow out and the rest come close but then move away again.

The phase patterns closely relate to the notion of *stability*. We call a state of a dynamical system stable if the system returns to that state after a small disturbance, or perturbation occurs. Otherwise, the state is called unstable. The interpretation on a phase portrait is the following: a state is stable if all nearby trajectories point towards it, and unstable otherwise. The sink and limit cycle are stable, but the source and saddle are unstable. All mentioned shapes characterise systems stability (resp. instability) around an *equilibrium*.

When the individual phase patterns are projected into the discrete abstraction of the given system, we get a discrete abstraction of patterns called the *discrete portrait patterns*. Some of the patterns can be characterised as single-state patterns (Fig. 1.4) while other patterns require several states to be included. For some single-state patterns, we can guarantee (by the abstraction procedures) that the abstracted pattern represents exactly the original phase portrait.

We can roughly classify all possible long-term runs in a Kripke structure as fixed points, cycles, and 'others'. To further classify individual elements of a phase portrait, in particular, other kinds of asymptotic behaviours of the system covered by 'others', a notion of a non-trivial strongly connected component (SCC) can be used. A set of states is called an *invariant set* if it has the property that placing a system into an arbitrary state of the set guarantees that the system cannot escape from the set and we speak about *strong invariancy* (the SCC is final in such case), or it *may* stay in the set forever (*weak invariancy*). A fixed point thus makes a special case of an invariant set.

The types of portrait patterns are characterised by properties of their constituting runs. Properties of runs can be formally represented as formulae of suitable temporal logic. We have already seen how to use the computational tree logic (CTL) to express various properties of runs. To describe adequately various kinds of complex patterns, we need a more expressive logic than CTL. We therefore employ a *hybrid extension* of CTL (with directions) which we now briefly introduce.

The main idea behind the hybrid extension is in the addition of special variables allowing to refer to states (*nominals*). The *down-arrow binder* $\downarrow s$ sets the state variable $s$ to the current state of evaluation. The formula $\downarrow s.\mathbf{AX}s$, for example, characterises the set of states which have themselves as their only next state (steady states). Another hybrid operator is the *at operator* ($@n$). Intuitively, $@n$ means 'at the state named by $n$'.

To describe a richer class of portrait patterns, it is also handy to consider some other extensions of CTL. One of them is the temporal logic *past* CTL (PCTL) which incorporates past operators facilitating the expression of 'time going backwards'. For example, the past formula $\hat{\mathbf{E}}\mathbf{F}Q$ represents the fact that once in the past $Q$ was true. Another extension, known as UCTL [16], adds the possibility to express 'directions' to the temporal operators. For example the formula $\mathbf{EX}_N\,true$ expresses the existence of a transition from the current state to the 'north'. For all additional operators and a full definition of HUCTL$_\mathrm{P}$, the hybrid CTL logic augmented with past operators and directions, we refer to [19].

We now show how to use the extended logic to describe patterns not definable in the standard version of CTL. We distinguish several classes of formulae. The following formulae are typical examples.

*Single-State patterns*

- sink (stable steady state): $\downarrow s.\mathbf{AX}s$
- source (only self-loops, no other incoming): $\downarrow s.\hat{\mathbf{A}}\mathbf{X}s$
- presence of self-loop (unstable steady state): $\downarrow s.\mathbf{EX}s$
- two-dimensional saddle: $\mathbf{AX}_{N\vee S}\,true \wedge \mathbf{EX}_N\,true \wedge \mathbf{EX}_S\,true \wedge \hat{\mathbf{A}}\mathbf{X}_{E\vee W}\,true \wedge \hat{\mathbf{E}}\mathbf{X}_E\,true \wedge \hat{\mathbf{E}}\mathbf{X}_W\,true$ (north–south outgoing, west–east incoming)

Another kind of general multi-state patterns are invariant sets like periodic runs or limit cycles. Here are some examples.

*Invariant sets (multi-state patterns)*

- state in a non-trivial SCC (i.e. on a cycle): $\downarrow s.\mathbf{EXEF}s$
- state in a final SCC (generalised sink): $\downarrow s.\mathbf{AGEF}s$
- state in an initial SCC (generalised source): $\downarrow s.\hat{\mathbf{A}}\mathbf{G}\hat{\mathbf{E}}\mathbf{F}s$
- non-north flow in the whole system: $\forall s.@s.\mathbf{AX}_{\neg N}\,true$

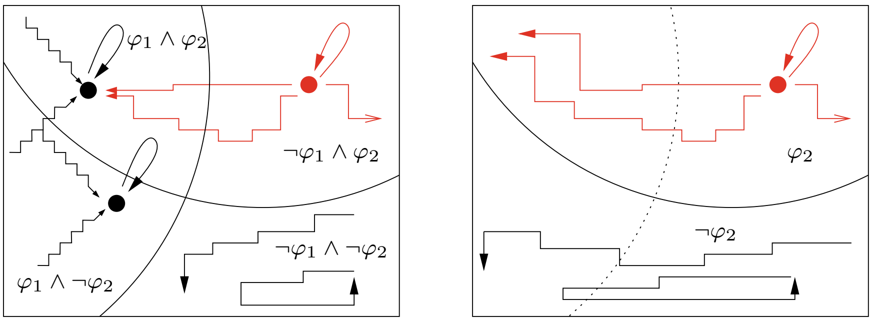
Relations among patterns (elements) of phase portraits can also be captured with HUCTL$_\mathrm{P}$ formulae as the following examples demonstrate.

*Relations among patterns*

- the system contains at least two sinks: $\exists s.\exists t.(@s.\neg t \wedge \mathbf{AX}s) \wedge (@t.\mathbf{AX}t)$
- the system contains at least two terminal SCCs: $\exists s.\exists t.(@s.\mathbf{AG}\neg t \wedge \mathbf{AGEF}s) \wedge (@t.\mathbf{AGEF}t)$ (similarly for initial SCCs)
- formula that is true in states that have two outgoing executions to two different sinks: $\exists s.\exists t.(@s.\neg t \wedge \mathbf{AX}s) \wedge (@t.\mathbf{AX}t) \wedge \mathbf{EF}s \wedge \mathbf{EF}t$ (intersection of basins of attraction of two different sinks)
- formula that is true in states that satisfy $\varphi_1$ and can reach a state satisfying $\varphi_2$ without ever going north: $\varphi_1 \wedge \exists s.(@s.\varphi_2) \wedge \mathbf{E}_{\neg\mathbf{N}}\mathbf{F}s$

We now turn our attention to the *digital bifurcation analysis* itself. When conducting the bifurcation analysis of a given system, we are interested in the question of how a phase portrait changes when parameter values vary. We therefore suppose we are given a *parametrised* $n$-dimensional KS where the parameters are taken from a finite set $\mathcal{P}$. For the purpose of the digital bifurcation analysis, we assume $\mathcal{P}$ to be a partially ordered set.

Digital bifurcation analysis allows to characterise *qualitative (structural)* changes in discrete phase portraits. Each of the changed situation can be captured by specifying the corresponding patterns of the phase portrait with a *finite set* of HUCTL$_{\mathrm{P}}$ formulae and observing when (regarding changes in parameters) these patterns appear or disappear (the formulae change their truth-value). The temporal logic specification of a set of phase portraits is called *phase portrait specification*. The set defines in an obvious way a division of the state space according to the validity of the individual formulae (the structure of the phase portrait) as exemplified in Fig. 1.5. From the practical view point, the phase portrait specification is supposed to describe various patterns appearing in the phase portrait and their mutual relationship. As an example consider two formulae, one expressing the reachability of a sink state ($\varphi_1 \overset{\mathrm{df}}{=} \mathbf{EF}(\downarrow s.\mathbf{AX}s)$) and the other one expressing the backward reachability of a source state ($\varphi_2 \overset{\mathrm{df}}{=} \hat{\mathbf{E}}\mathbf{F}(\downarrow s.\hat{\mathbf{A}}\mathbf{X}s)$). The state space is in general divided into four parts as is shown in Fig. 1.5 (left).



$\varphi_1 \wedge \varphi_2$
$\neg\varphi_1 \wedge \varphi_2$
$\neg\varphi_1 \wedge \neg\varphi_2$
$\varphi_1 \wedge \neg\varphi_2$
$\varphi_2$
$\neg\varphi_2$

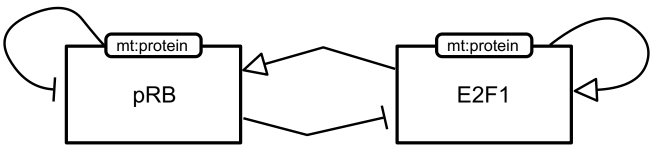
**Fig. 1.5** Characterisation by formulae and change of phase portraits

A change in any of the parameter values may affect the corresponding transition relation and may thus result in a change of the truth-value of the respective formula in the portrait specification. If such a change results in non-satisfiability of one of the formulae (or its negation) in the specification, we consider this as a *structural change* in the phase portrait—a *bifurcation*. For the example above, if the parameter value changes in such a way that the formula $\varphi_1$ does not hold in any state, the four-part 'structure' collapses into two parts, as shown in Fig. 1.5 (right). The set of all parameter values for which the structure of the phase portrait is kept unchanged is called a *stratum*. A boundary point of a stratum is called a *bifurcation point*. We obtain the so-called *parametric portrait* of the system by taking all strata in the parameter space with respect to a given phase portrait specification. The parametric portrait together with its characteristic phase portraits makes a *bifurcation diagram*. The set of phase portraits that are characteristic for a given parametric portrait is called the *phase portrait pattern*.

## 1.4 Case Studies

### 1.4.1 Regulation of $G_1/S$ Cell Cycle Transition

We now consider the well-known ODE model of a two-gene regulatory network [94] to indicate the applicability of our framework. The network describes the interaction of the tumour suppressor protein $pRB$ and the central transcription factor $E2F1$, see Fig. 1.6 (top). This interaction is the essential mechanism that governs the transition from the $G_1$ phase to the $S$ phase in the cell cycle of mammalian cells. The $G_1$ phase makes an important decision point. If the concentration levels of $E2F1$ are high, the $G_1/S$ transition mechanism is activated. If the concentration levels of $E2F1$ are low, the transition to $S$ phase is refused and the cell thus avoids the DNA replication.



$$\frac{d[pRB]}{dt} = k_1 \frac{[E2F1]}{K_{m1}+[E2F1]} \frac{J_{11}}{J_{11}+[pRB]} - \phi_{pRB}[pRB]$$
$$\frac{d[E2F1]}{dt} = k_p + k_2 \frac{a^2+[E2F1]^2}{K_{m2}^2+[E2F1]^2} \frac{J_{12}}{J_{12}+[pRB]} - \phi_{E2F1}[E2F1]$$

$$a = 0.04,\ k_1 = 1,\ k_2 = 1.6,\ k_p = 0.05,\ \phi_{E2F1} = 0.1$$
$$J_{11} = 0.5,\ J_{12} = 5,\ K_{m1} = 0.5,\ K_{m2} = 4$$

**Fig. 1.6** The $G_1/S$ transition regulatory network represented in SBGN and its ODE model taken from [94]

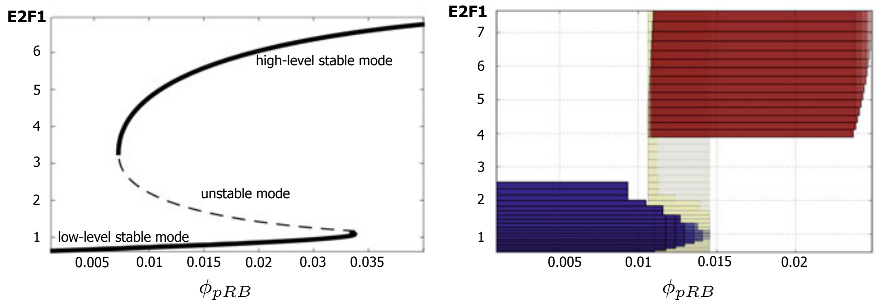


**Fig. 1.7** (left) Equilibrium diagram reproducing the results achieved in [94]. (right) Visualisation of the parameter synthesis results. The high and low stable regions are represented by the red and blue coloured areas, respectively. The yellow areas denote the states in which the *bistable switch* formula $\varphi$ is satisfied

This mechanism is an example of a pattern called the *bistable switch*. This irreversible process eventually reaches one of two different stable equilibria. In this case, we would like to investigate the existence of such two stable states on $E2F1$. The activity of $pRB$ is modulated by the phosphorylation/dephosphorylation turn-over controlled by growth factor signals transferred to cyclin-dependent kinases each acting on a specific subset of $pRB$ phosporylation sites [77]. This is captured in the ODE model using the parameter $\phi_{pRB}$ describing the degradation rate of $pRB$.

In [94], the authors provide a numerical bifurcation analysis that investigates the $E2F1$ equilibria depending on $\phi_{pRB}$. They have constructed an equilibrium point curve for $E2F1$ and discovered two saddle-node bifurcation points by non-trivial elaboration with numerical analysis methods and utilising previous knowledge of the equilibria. This result is illustrated in Fig. 1.7 (left). For $\phi_{pRB}$ less than 0.007, the system converges to a single stable equilibrium with low $E2F1$ concentration. For $\phi_{pRB}$ higher than 0.027, the system dually converges to a single stable equilibrium with high $E2F1$ concentration. If $\phi_{pRB}$ is between these two bifurcation points the system exhibits the bistable switch behaviour. This means that there can always be found an unstable equilibrium whose neighbourhood intersects the basins of attraction of both the stable equilibria.

To employ our framework in this case, we first create the piece-wise multi-affine approximation (PMA) of the nonlinear ODE model [61]. Each nonlinear function appearing in the ODEs is approximated with an optimal sequence of piece-wise affine ramp functions. In this case study, we have set the granularity to 70 affine segments per each function. We then employ the technique of rectangular abstraction [11] to obtain a finite discrete transition system over-approximating the PMA.

## CTL Parameter Synthesis

The bistable switch situation we are interested in can be described by the CTL formula $\varphi \equiv \mathbf{EFAG}\texttt{high} \wedge \mathbf{EFAG}\texttt{low}$. Informally, this formula describes a branching that can end in two different stable states with different concentration levels. The atomic propositions `low` and `high`, which express the location of the $E2F1$ stable equilibria, are chosen based on the results reported in [94]. We thus define `high` $\equiv (E2F1 > 4 \wedge E2F1 < 7.5)$ and `low` $\equiv (E2F1 > 0.5 \wedge E2F1 < 2.5)$. These definitions cover the expected regions of the two stable attractors as well as a part of their attractor basins. We use the formula $\varphi$ together with the initial parameter space $\phi_{pRB} \in [0.001, 0.025]$ as the input to the coloured CTL model checking algorithm.

The results obtained using the algorithm are illustrated in Fig. 1.7 (right) and can be compared against the equilibrium curve of [94] in Fig. 1.7 (left). The blue area represents the states satisfying the formula **AG**`low`. In such states, the low concentration of $E2F1$ is *guaranteed* to stabilise for the corresponding values of $\phi_{pRB}$ in the PMA. This is due to the fact that the abstraction we employ is an over-approximation [42] in the following sense: For every trajectory in the PMA, there has to exist a corresponding run in the parametrised Kripke structure. As an example, the model checking result implies that if the value of the parameter is fixed at 0.005 then no run in the parametrised Kripke structure exits the concentration bounds $0.5 \leq E2F1 \leq 2.5$; hence, there is no such trajectory in the PMA. On the other hand, although no red region is identified at $\phi_{pRB} = 0.005$ we cannot be sure that the corresponding property holds also in the PMA since a spurious run might have been introduced by the abstraction. For a universally quantified CTL formula, the abstraction makes the parameter space synthesised by model checking under-approximated [11]. As $\phi_{pRB}$ gets closer to the bistable region, the guarantee of low stabilisation becomes limited to a smaller subset of the `low` region until it disappears at $\phi_{pRB} > 0.0145$. A similar observations can be made about the red area representing the states satisfying **AG**`high`. Note that in this case the effect of the parameter value under-approximation is much less significant, which can be seen from the comparison with the equilibrium point curve. For $\phi_{pRB} \in [0.012, 0.0145]$, the system has two stable equilibria, i.e. the set of states satisfying **AG**`low` and the set of states satisfying **AG**`high` are both guaranteed to be non-empty.

The yellow area represents states satisfying the formula $\varphi$ as explained above. Since the satisfaction of an **EF**-formula might be caused by a spurious behaviour introduced by the abstraction, this result does not come with any guarantees but can be rather seen as an estimate of the parameter values and initial conditions under which the system exhibits the bistable switch behaviour and both stable regions might be reached. The fill opacity of the coloured areas of the diagram represents the number of states with the fixed $E2F1$ and $\phi_{pRB}$ values and the varying amounts of $pRB$. This can be seen as a projection of the third axis (value of $pRB$) onto the 2D plane of the diagram. The grey area represents the fact there are values of $pRB$ from which the red or the blue region is not reachable. This information is again guaranteed.

### Bifurcation Analysis

We now focus on the bifurcation analysis with respect to the parameter $\phi_{pRB}$. As explained above, the biological switch is known to be bistable, i.e. two different stable states can exist in the systems dynamics; furthermore, this bi-stability is known to be sensitive to change in the parameter $\phi_{pRB}$. To explore this sensitivity, we express the portrait specification using the following formulae:

- $\varphi_1 := \exists s.\exists t.(@s.\mathbf{AGEF}s) \wedge (@t.\neg\mathbf{EF}s \wedge \mathbf{AGEF}t) \wedge \mathbf{E}_{\neg\uparrow\mathbf{E2F1}}\mathbf{F}s \wedge \mathbf{E}_{\neg\downarrow\mathbf{E2F1}}\mathbf{F}t$
  *The system contains at least two terminal SCCs (generalised stable states); furthermore, the formula holds in states that have a non-$E2F1$-increasing execution (i.e. an execution on which $E2F1$ decreases or stay the same in each step) leading to one of the terminal SCCs and similarly a non-E2F1-decreasing execution to the other terminal SCC.*
- $\varphi_2 := \neg\varphi_1 \wedge \downarrow s.\mathbf{AGEF}s \wedge E2F1 < 4$
  *There is exactly one terminal SCC and it is located in the region where the value of E2F1 is **below*** 4. *This formula holds in all states included in this terminal SCC.*
- $\varphi_3 := \neg\varphi_1 \wedge \downarrow s.\mathbf{AGEF}s \wedge E2F1 > 4$
  *There is exactly one terminal SCC and it is located in the region where the value of E2F1 value is **above*** 4.

We do not have to compute the entire parametric portrait here as we have previous knowledge about the system dynamics as well as the relation among the three formulae. We thus rather focus directly on the mutual bifurcations among the portraits characterised by the individual formulae. The results we obtain using our prototype implementation are as follows: $\varphi_1$ holds for $\phi_{pRB} \in [0.011, 0.0136]$, $\varphi_2$ for $\phi_{pRB} \in [0.002, 0, 011]$ and $\varphi_3$ for $\phi_{pRB} \in [0.0136, 0.5]$. We can thus draw the conclusion that the values 0.011 and 0.0136 represent the bifurcation points of the parametrised system. For $\phi_{pRB} = 0.011$ the portrait changes between $\varphi_2$ and $\varphi_1$ and for $\phi_{pRB} = 0.0136$ it changes between $\varphi_1$ and $\varphi_3$. Figure 1.8 shows the vector fields and the corresponding abstractions for three chosen values of $\phi_{pRB}$, each belonging to one of the computed intervals. The blue rectangles depict the states that satisfy the particular formulae.

We have also explored a variant of the formula $\varphi_1$ in which we also require the states satisfying the formula to not be sources (and thus require them to actually be saddle states): $\varphi_1' := \varphi_1 \wedge \neg(\downarrow s.\hat{\mathbf{A}}\mathbf{X}s)$. The results obtained using our prototype show that this formula also holds for $\phi_{pRB} \in [0.011, 0.0136]$.

Note that the results we obtain are affected by the precision of the approximation and abstraction of the original continuous model. Nevertheless, the intervals of $\phi_{pRB}$ computed using our method comply with the numerical bifurcation analysis presented in [94].

## 1.4.2 Signalling Pathways

In this section, we demonstrate the use of parameter synthesis to differentiate models that display or do not display a particular studied dynamical behaviour expressed in the form of a temporal property. The details on the modelling and analysis provided here have been published in [62].

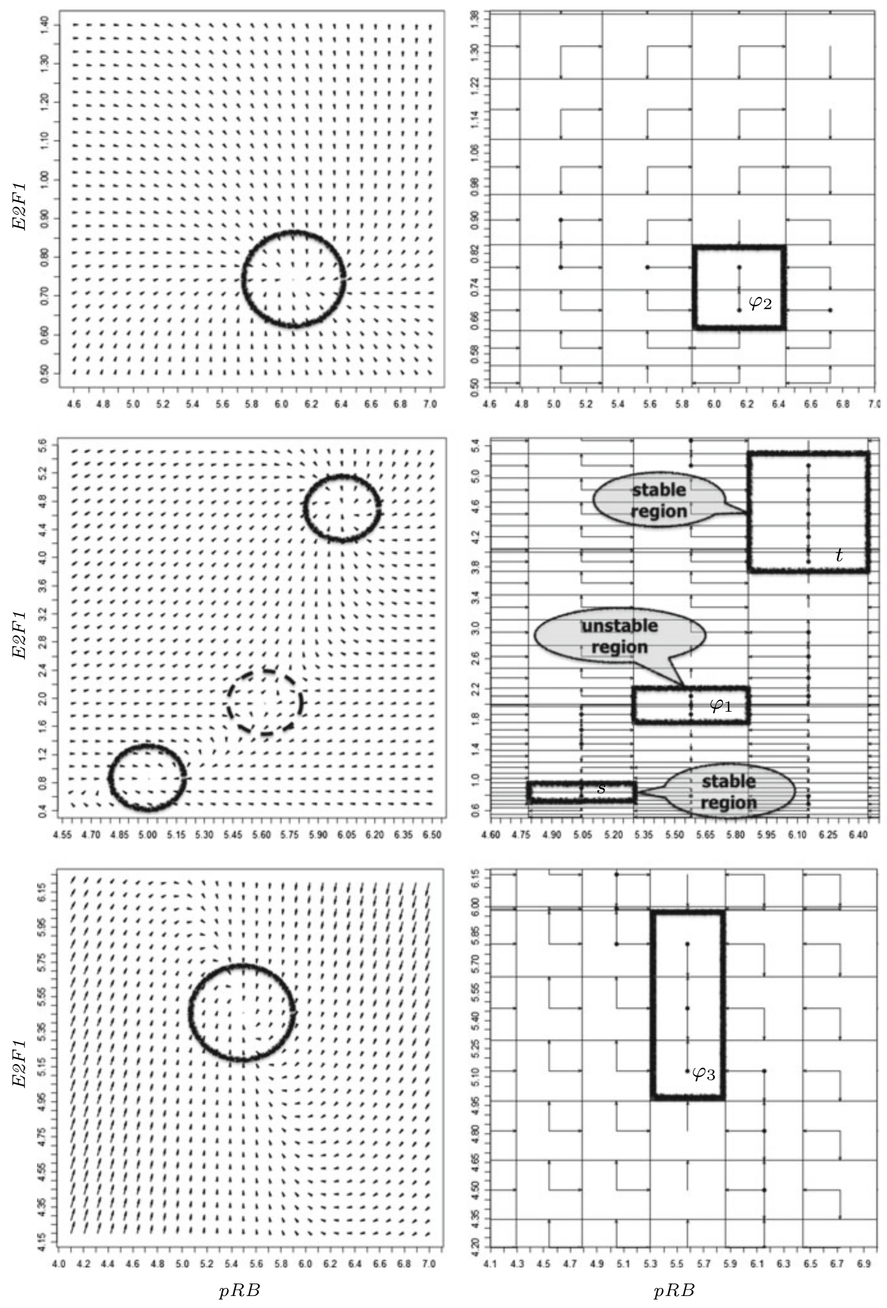


**Fig. 1.8** The vector fields (left) and their corresponding abstractions (right). The situations are displayed for three different settings of $\phi_{pRB}$: 0.0075 (top), 0.0115 (middle) and 0.014 (bottom). The highlighted regions mark the states satisfying the respective formulae. The regions marked $s$ and $t$ denote the states matching the corresponding variables in $\varphi_1$

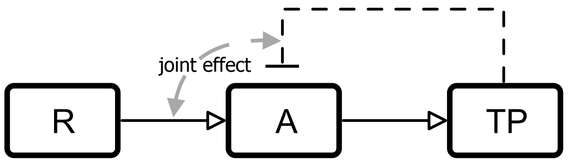


**Fig. 1.9** The topology of the models represented by means of an SBGN activity flow diagram. The dashed line represents optional inhibition (left). The positive and negative Hill functions (right)

The signalling pathways are one of the most important biochemical mechanisms currently studied by systems biology. They represent a complex cellular information processing machinery that evaluates input stimuli and transfers them into genome by means of regulation of specific genes expression. It is important to distinguishing between monotone (*sustained*) and non-monotone (*transient*) time-course behaviour of signalling pathways [87, 95]. It is believed that transition between these two modes may cause significant changes of the nominal cell behaviour that may lead to serious anomalies of the internal cellular processes control.

We consider three ODE models describing a general shape of signalling pathways at a high level of abstraction. In particular, we focus on three topologies differing in the presence/absence of feedback mechanisms (Fig. 1.9). We use Hill kinetics employing sigmoidal functions to describe the response of a signalling component with respect to the input signal (see equations in Fig. 1.9). In particular, our models consist of the following entities: the receptor ($R$), the adapter ($A$) and the target protein ($TP$). The receptor concentration is constant. The adapter forms the main dynamical entity of the model. It is activated by the receptor and inhibited by the target protein. The first model includes no inhibition (Eq. 1.3); the second model describes independent inhibition (Eq. 1.4); and the third model describes dependent inhibition (Eq. 1.5) using a negative sigmoidal function (Eq. 1.2). The target protein dynamics is modelled as a positive sigmoidal function (Eq. 1.1) of the adapter in all three models (Eq. 1.6). The topology of the models is illustrated in Fig. 1.9.

$$hill(X, K_{\mathrm{M}}, n) = \frac{[X]^n}{K_{\mathrm{M}}{}^n + [X]^n} \tag{1.1}$$

$$hill^-(X, K_{\mathrm{M}}, n) = \frac{K_{\mathrm{M}}{}^n}{K_{\mathrm{M}}{}^n + [X]^n} \tag{1.2}$$

$$\frac{d[A]}{dt} = V_{\mathrm{MAX\_}A} \cdot hill(R, K_{\mathrm{M\_}A}, n_A) - y_A[A] \tag{1.3}$$

$$\frac{d[A]}{dt} = V_{\mathrm{MAX\_}A1} \cdot hill(R, K_{\mathrm{M\_}A1}, n_{A1}) + V_{\mathrm{MAX\_}A2} \cdot hill^-(TP, K_{\mathrm{M\_}A2}, n_{A2}) - y_A[A] \tag{1.4}$$

$$\frac{d[A]}{dt} = V_{\mathrm{MAX\_}A1} \cdot hill(R, K_{\mathrm{M\_}A1}, n_{A1}) \cdot V_{\mathrm{MAX\_}A2} \cdot hill^-(TP, K_{\mathrm{M\_}A2}, n_{A2}) - y_A[A] \tag{1.5}$$

$$\frac{d[TP]}{dt} = V_{\mathrm{MAX\_}TP} \cdot hill(A, K_{\mathrm{M\_}TP}, n_{TP}) - y_{TP}[TP] \tag{1.6}$$

The Hill function describing the adapter activation is also constant based on the biological assumptions of constant receptor activity. We therefore simplify the model equations as follows:

$$\frac{d[A]}{dt} = V_A - y_A[A] \tag{1.7}$$

$$\frac{d[A]}{dt} = V_A + V_{\mathrm{MAX\_}A} \cdot hill^{-}(TP, K_{\mathrm{M\_}A}, n_A) - y_A[A] \tag{1.8}$$

$$\frac{d[A]}{dt} = V'_A \cdot hill^{-}(TP, K_{\mathrm{M\_}A}, n_A) - y_A[A] \tag{1.9}$$

The parameter $V'_A$ is defined as $V_A \cdot V_{\mathrm{MAX\_A}}$. The default parameters have been set to: $V = V'_A = V_{\mathrm{MAX\_}A} = V_{\mathrm{MAX\_}TP} = 0.001$, $K_{\mathrm{M\_}A} = K_{\mathrm{M\_}TP} = y_A = y_{TP} = 0.1$, $n_A = n_{TP} = 2$. The initial concentrations of all entities have been set to 0: $A(0) = TP(0) = 0$.

In order to prepare the model for model checking analysis, we perform the steps described at the end of Sect. 1.3.2. First, we construct the piece-wise affine approximation (PAA) of the original nonlinear continuous models by applying the automatised approximation procedure introduced in [61]. In particular, we approximate each nonlinear function appearing in the equations with a sum of ten piece-wise affine ramp functions. Second, we apply the abstraction procedure based on [15, 42]. As a result of the abstraction, we obtain a parametrised direction transition system (PDTS) that exactly over-approximates the PAA model.

We formulate the properties of interest in terms of UCTL formulae. The usage of a branching-time logic is motivated by the fact that the rectangular over-abstraction of the original model is nondeterministic. Furthermore, the combination of action and state predicates is used to express both local patterns of the dynamics (state predicates) and the character of the transitions (action predicates).

Here, we use the extension of the coloured CTL model checking algorithm that is able to deal with the UCTL operators. This extension is a straightforward enhancement of the algorithm described in [32]. Using this procedure, we obtain a global result that describes all the initial states and corresponding parameter values for which the given formula holds. It is important to note the consequences of the over-approximating abstraction for the parameter synthesis results. On the one hand, satisfaction of a formula is guaranteed for all universally quantified UCTL formulae. On the other hand, falsification of a formula is guaranteed for all existentially quantified UCTL formulae. To preserve these guarantees, the formulae may not contain alternations of quantifiers. In both of the cases where results are guaranteed, the obtained parameter value regions are under-approximated.

In this case, the parameters for synthesis are chosen as follows: $V_A$ in the first model, $V_{\mathrm{MAX\_}A}$ in the second model, $V'_A$ in the third model and $V_{\mathrm{MAX\_}TP}$ in all three models. The range of these parameters is set to [0.0001, 10]; other constants are fixed at the default values with the only exception of $y_A$, $y_{TP}$ (set to 0.5009) and $V_A$ in the second model (set to 0.001).

**Table 1.1** The results obtained for the given properties in the three different models. The initial concentration of $A$ is the union of the concentration values in states where the formula holds for the stated parameters. $V^*$ is the production parameter of $A$ that represents $V_A$ in the first model, $V_{\mathrm{MAX}\_A}$ in the second model, $V'_A$ in the third model. Each parameter interval represents the range of all satisfying parameter values across all the states of the respective model where the particular property holds

| Model type | Property | Initial concentration of $A$ | $V^* \times V_{\mathrm{MAX}\_TP}$ |
|---|---|---|---|
| Model 1 | $\varphi_1$ | [0.22, 11.9] | $[0.11, 5.96] \times [0.0, 6.81]$ |
| | $\neg\varphi_2$ | [0.22, 12.0] | $[0.11, 10.0] \times [7.23, 10.0]$ |
| | $\neg\varphi_3$ | [0.01, 12.0] | $[0.23, 10.0] \times [6.26, 10.0]$ $\cup[5.96, 10.0] \times [0.0, 10.0]$ |
| Model 2 | $\varphi_1$ | [0.01, 11.5] | $[0.0, 5.76] \times [0.0, 0.01]$ |
| | $\neg\varphi_2$ | [0.22, 12.0] | $[2.29, 10.0] \times [7.23, 10.0]$ |
| | $\neg\varphi_3$ | [0.0, 12.0] | $[0.09, 10.0] \times [7.23, 10.0]$ $\cup[4.47, 10.0] \times [0.0, 10.0]$ |
| Model 3 | $\varphi_1$ | [0.01, 11.5] | $[0.01, 5.76] \times [0.0, 0.01]$ |
| | $\neg\varphi_2$ | [0.22, 12.0] | $[2.31, 10.0] \times [7.23, 10.0]$ |
| | $\neg\varphi_3$ | [0.0, 12.0] | $[0.11, 10.0] \times [7.23, 10.0]$ $\cup [4.49, 10.0] \times [0.0, 10.0]$ |

To express the combined characteristics of the two behaviours of $TP$, we employ the following UCTL-specified properties:

$$\begin{aligned}
\varphi_1 &= \mathit{Init} \wedge \mathbf{AX}_{\uparrow TP}(\mathbf{AF}_{\neg\downarrow TP}\ \mathit{Stable}),\\
\varphi_2 &= \mathit{Init} \wedge \mathbf{EF}(\mathbf{EX}_{\uparrow TP}(\mathbf{EF}_{\neg\downarrow TP}(\mathbf{EF}_{\neg\uparrow TP}(\mathbf{EX}_{\downarrow TP}\ \mathit{True})))),\\
\varphi_3 &= \mathit{Init} \wedge \mathbf{EF}(\mathbf{EX}_{\uparrow TP}(\mathbf{EF}_{\neg\downarrow TP}(\mathbf{EF}_{\neg\uparrow TP}(\mathbf{EX}_{\downarrow TP}\ \mathit{NotUp})))).
\end{aligned}$$

Here, *Init* represents the set of initial states (with $TP$ constrained in the range [0.0, 0.0001]), *NotUp* represents a state in which the abstracted vector field disallows the increase in the $TP$ concentration, *Stable* represents an equilibrium where both species are stable, and *True* represents the set of all states. Intuitively, the formula $\varphi_1$ is a specification of the sustained behaviour; its satisfaction guarantees this behaviour in a given model for the given initial states. Formally, this formula requires all admissible runs to start in *Init*, to increase the $TP$ concentration in the very next step, and not to decrease it before reaching *Stable*. The formulae $\varphi_2$ and $\varphi_3$ specify the necessary conditions for the presence of transient behaviour. Both formulae require all admissible runs to start in *Init* and to increase the $TP$ concentration at least once before decreasing it also at least once. Using the parametrisations that violate these properties, the system is guaranteed the absence of the transient behaviour. All the obtained results are summarised in Table 1.1.

Note that the abstraction together with the chosen logic UCTL allow us to exactly characterise the inevitability of the sustained behaviour. However, as reported in [30], the state predicate *Stable* is exactly preserved in the abstraction only if a particular equilibrium in the respective state rectangles is hyperbolic. As a consequence, we are unable to fully cover the transient behaviour. More specifically, the transient behaviour might asymptotically converge to an equilibrium that is asymptotically stable. Such a property is not preserved by the abstraction. We therefore limit ourselves to only refuting the *absolutely transient* behaviour, i.e. the transient behaviour without any oscillations around the target equilibrium.

**Conclusions**

Recent advances in formal analysis and verification techniques for computer systems and formal modelling of biological and biomedical systems brought us to a point where formal methods might be applicable to biological systems as well. In this little review, we pointed out to a few areas where biologists could benefit from the application of formal methods. Namely, we have focused on techniques working with parameter-uncertain models.

In general, the ultimate prerequisite is to develop formal models of systems we want to understand. For an excellent position paper on formal modelling in systems biology, we refer to [24]. A recent overview covering a broad set of techniques to formal specification and analysis of biological systems is provided in [10].

**Acknowledgements** This work has been partially supported by the Czech Science Foundation grant No. 18-00178S.

## References

1. Andreychenko A, Mikeev L, Wolf V (2015) Model reconstruction for moment-based stochastic chemical kinetics. ACM Trans Model Comput Simul 25(2):12:1–12:19
2. Areces C, ten Cate B (2007) Hybrid logics. In: Blackburn P, van Benthem J, Wolter F (eds) Handbook of modal logic, vol 3, 1st edn. Elsevier
3. Arellano G, Argil J, Azpeitia E, Benítez M, Carrillo M, Góngora P, Rosenblueth D, Alvarez-Buylla E (2011) “Antelope” a hybrid-logic model checker for branching-time boolean grn analysis. BMC Bioinform 12(1):490
4. Backenköhler M, Bortolussi L, Wolf V (2018) Moment-based parameter estimation for stochastic reaction networks in equilibrium. IEEE/ACM Trans Comput Biol Bioinform 15(4):1180–1192
5. Baier C, Katoen JP (2008) Principles of model checking. The MIT Press
6. Barnat J, Brim L, Krejčí A, Štreck A, Šafránek D, Vejnár M, Vejpustek T (2012) On parameter synthesis by parallel model checking. IEEE/ACM Trans Comput Biol Bioinform 9(3):693–705
7. Barnat J, Brim L, Černá I, Dražan S, Fabriková J, Láník J, Šafránek D, Ma H (2009) BioDiVinE: A framework for parallel analysis of biological models. In: Computational models for cell processes (COMPMOD). EPTCS, vol 6, pp 31–45
8. Barnat J, Brim L, Šafránek D (2010) High-performance analysis of biological systems dynamics with the DiVinE model checker. Brief Bioinform 11(3):301–312
9. Bartocci E, Corradini F, Merelli E, Tesei L (2010) Detecting synchronisation of biological oscillators by model checking. Theor Comput Sci 411(20):1999–2018
10. Bartocci E, Liò P (2016) Computational modeling, formal analysis, and tools for systems biology. PLOS Comput Biol 12(1):1–22
11. Batt G, Belta C, Weiss R (2007) Model checking liveness properties of genetic regulatory networks. In: TACAS. LNCS, vol 4424. Springer, pp 323–338

12. Batt G, Page M, Cantone I, Gössler G, Monteiro P, de Jong H (2010) Efficient parameter search for qualitative models of regulatory networks using symbolic model checking. Bioinformatics 26(18):603–610
13. Batt G, Ropers D, Jong HD, Geiselmann J, Mateescu R, Schneider D (2005) Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in escherichia coli. Bioinformatics 21:19–28
14. Batt G, Salah RB, Maler O (2007) On timed models of gene networks. In: Formal modeling and analysis of timed systems (FORMATS). LNCS, Springer, Berlin, pp 38–52
15. Batt G, Yordanov B, Weiss R, Belta C (2007) Robustness analysis and tuning of synthetic gene networks. Bioinformatics 23(18):2415–2422
16. ter Beek MH, Fantechi A, Gnesi S, Mazzanti F (2011) A state/event-based model-checking approach for the analysis of abstract system properties. Sci Comput Prog 76:119–135
17. Behrmann G, David A, Larsen KG (2004) A tutorial on UPPAAL. In: 4th international school on formal methods for the design of computer, communication, and software systems in formal methods for the design of real-time systems (SFM-RT), No. 3185. LNCS, Springer, Berlin, pp 200–236
18. Belta C, Habets LCGJM (2006) Controlling a class of nonlinear systems on rectangles. IEEE Trans Automat Contr 51(11):1749–1759
19. Beneš N, Brim L, Demko M, Pastva S, Šafránek D (2016) Parallel SMT-based parameter synthesis with application to piecewise multi-affine systems. ATVA. LNCS 9938:192–208
20. Beneš N, Brim L, Demko M, Pastva S, Šafránek D (2016) A model checking approach to discrete bifurcation analysis. In: Fitzgerald J, Heitmeyer C, Gnesi S, Philippou A (eds.) FM 2016, vol 9995. LNCS, Springer, pp 85–101
21. Bernot G, Comet JP, Richard A, Guespin J (2004) Application of formal methods to biological regulatory networks: extending thomas' asynchronous logical approach with temporal logic. J Theor Biol 229(3):339–347
22. Biere A, Cimatti A, Clarke E, Zhu Y (1999) Symbolic model checking without BDDs. In: Cleaveland WR (ed) Tools and algorithms for the construction and analysis of systems, vol 6806. LNCS, Springer, Berlin, pp 193–207
23. Bogomolov S, Schilling C, Bartocci E, Batt G, Kong H, Grosu R (2015) Abstraction-based parameter synthesis for multiaffine systems. In: Hardware and software: verification and testing, lecture notes in computer science, vol 9434, Springer International Publishing, pp 19–35
24. Bonzanni N, Feenstra KA, Fokkink W, Krepska E (2009) What can formal methods bring to systems biology? In: FM 2009: formal methods, second world congress, eindhoven, The Netherlands, November 2-6, 2009. Proceedings, Lecture notes in computer science, vol 5850, Springer, Berlin, pp 16–22
25. Bortolussi L, Cardelli L, Kwiatkowska M, Laurenti L (2016) Approximation of probabilistic reachability for chemical reaction networks using the linear noise approximation. In: Quantitative evaluation of system (QEST 2016), vol 9826, Springer, Berlin, pp 72–88
26. Bortolussi L, Milios D, Sanguinetti G (2015) U-check: model checking and parameter synthesis under uncertainty. In: Campos J, Haverkort BR (eds) Quantitative evaluation of systems. Springer International Publishing, Cham, pp 89–104
27. Bortolussi L, Milios D, Sanguinetti G (2016) Smoothed model checking for uncertain continuous-time markov chains. Inf Comput 247:235–253
28. Brim L, Češka M, Šafránek D (2013) Model checking of biological system. In: 13th International school on formal methods for the design of computer, communication and software systems: dynamical systems
29. Brim L, Barnat J (2007) Tutorial: Parallel model checking. In: Bosnacki D, Edelkamp S (eds) Model checking software, 14th International SPIN workshop, Berlin, Germany, July 1-3, 2007, Proceedings, Lecture notes in computer science, vol 4595, Springer, Berlin, pp 187–203
30. Brim L, Demko M, Pastva S, Šafránek D (2015) High-performance discrete bifurcation analysis for piecewise-affine dynamical systems. In: Hybrid systems biology, Springer, Berlin, pp 58–74
31. Brim L, Dluhoš P, Šafránek D, Vejpu stek T (2014) STL*: extending signal temporal logic with signal-value freezing operator. Information and computation 236, 52–67, special Issue on Hybrid Systems and Biology

32. Brim L, Češka M, Demko M, Pastva S, Šafránek D (2015) Parameter synthesis by parallel coloured CTL model checking. In: Roux O, Bourdon J (eds) Computational methods in systems biology, Lecture notes in computer science, vol 9308, pp 251–263. Springer International Publishing (2015)
33. Burch JR, Clarke EM, McMillan KL, Dill DL, Hwang LJ (1992) Symbolic model checking: $10^{20}$ states and beyond. Inf Comput 98(2):142–170
34. Calzone L, Chabrier-Rivier N, Fages F, Soliman S (2006) Machine learning biochemical networks from temporal logic properties. In: Transactions on computational systems biology VI, LNCS, Springer, Berlin, Heidelberg, pp 68–94
35. Champneys A, Tsaneva-Atanasova K (2013) Dynamical systems theory, bifurcation analysis. In: Encyclopedia of systems biology, Springer, Berlin, pp 632–637
36. Chaouiya C, Remy E, Mossé B, Thieffry D (2003) Qualitative analysis of regulatory graphs: a computational tool based on a discrete formal framework. In: Positive systems, vol 294, LNCIS, Springer, pp 830–832
37. Cimatti A, Clarke E, Giunchiglia E, Giunchiglia F, Pistore M, Roveri M, Sebastiani R, Tacchella A (2002) NuSMV 2: an OpenSource tool for symbolic model checking. Computer aided verification (CAV), vol 2404, LNCS, Springer, Berlin, Heidelberg, pp 359–364
38. Clarke EM, Enders R, Filkorn T, Jha S (1996) Exploiting symmetry in temporal logic model checking. Form Methods Syst Des 9(1–2):77–104
39. Clarke EM, Grumberg O, Peled DA (1999) Model checking. MIT Press, Cambridge
40. Clarke E, Zuliani P (2011) Statistical model checking for cyber-physical systems. Automated technology for verification and analysis (ATVA), vol 6996. LNCS, Springer, Berlin Heidelberg, pp 1–12
41. Clarke E, Grumberg O, Jha S, Lu Y, Veith H (2001) Progress on the state explosion problem in model checking. In: Informatics - 10 Years back. 10 Years ahead, vol 2000, LNCS, Springer, Berlin, pp 176–194
42. Collins P, Habets LC, van Schuppen JH, Černá I, Fabriková J, Šafránek D (2011) Abstraction of biochemical reaction systems on polytopes. In: Proceedings of the 18th IFAC world congress, vol 18, pp 14869–14875
43. Dang T, Donze A, Maler O, Shalev N (2008) Sensitive state-space exploration. In: IEEE conference on decision and control, pp 4049–4054
44. Didier F, Henzinger TA, Mateescu M, Wolf V (2010) Sabre: a tool for stochastic analysis of biochemical reaction networks. CoRR arXiv:abs/1005.2819
45. Dluhoš P, Brim L, Šafránek D (2012) On expressing and monitoring oscillatory dynamics. In: Hybrid systems and biology (HSB), vol 92, EPTCS, pp 73–87
46. Donaldson R, Gilbert D (2008) A model checking approach to the parameter estimation of biochemical pathways. In: CMSB, vol 5307, LNCS, Springer, Berlin, pp 269–287
47. Donzé A (2010) Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Computer aided verification (CAV). LNCS, Springer, Berlin, Heidelberg, pp 167–170
48. Donzé A (2010) Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: CAV. vol 10, Springer, pp 167–170
49. Donzé A, Clermont G, Langmead CJ (2010) Parameter synthesis in nonlinear dynamical systems: application to systems biology. J Comput Biol 17(3):325–336
50. Eker S, Knapp M, Laderoute K, Lincoln P, Meseguer J, Sonmez K (2002) Pathway logic: symbolic analysis of biological signaling. In: Pacific symposium on biocomputing, pp 400–412
51. Emerson EA, Sistla AP (1996) Symmetry and model checking. Form Methods Syst Des 9(1–2):105–131
52. Fages F, Soliman S (2008) Formal cell biology in Biocham. In: 8th International school on formal methods for the design of computer, communication and software systems: computational systems biology SFM08, vol 5016, pp 54–80
53. Fages F, Rizk A (2008) On temporal logic constraint solving for analyzing numerical data time series. Theor Comput Sci 408(1):55–65
54. Fisher J, Henzinger TA (2007) Executable cell biology. Nat Biotechnol 25(11):1239–1249

55. Fröhlich F, Theis F, Hasenauer J (2014) Uncertainty analysis for non-identifiable dynamical systems: profile likelihoods, bootstrapping and more. In: CMBS, vol 8859, LNCS, Springer, Berlin, pp 61–72
56. Gábor, A., Banga, J.R.: Improved parameter estimation in kinetic models: selection and tuning of regularization methods. In: CMSB, vol 8859, LNCS, Springer, Berlin, pp 45–60
57. Gao S, Kong S, Clarke EM (2013) dReal: An SMT solver for nonlinear theories over the reals. In: CADE-24. , vol. 7898, LNCS, Springer, Berlin, pp 208–214
58. Geldenhuys J, de Villiers PJA (1999) Runtime efficient state compaction in SPIN. In: Model checking software (SPIN), vol 1680. LNCS, Springer, Berlin, pp 12–21
59. Gilbert D, Breitling R, Heiner M, Donaldson R (2009) An introduction to biomodel engineering, illustrated for signal transduction pathways. Membrane computing, vol 5391, LNCS, Springer, Berlin, pp 13–28
60. Goethem SV, Jacquet JM, Brim L, Šafránek D (2013) Timed modelling of gene networks with arbitrary expression level discretization. In: Interactions between computer science and biology. ENTCS, Elsevier
61. Grosu R, Batt G, Fenton FH, Glimm J, Guernic CL, Smolka SA, Bartocci E (2011) From cardiac cells to genetic regulatory networks. CAV LNCS 6806:396–411
62. Hajnal M, Šafránek D, Demko M, Pastva S, Krejčí P, Brim L (2016) Toward modelling and analysis of transient and sustained behaviour of signalling pathways. In: Hybrid systems biology - 5th international workshop, HSB 2016, Grenoble, France, October 20-21, 2016, Proceedings. Springer, Berlin, pp 57–66
63. Heath J, Kwiatkowska M, Norman G, Parker D, Tymchyshyn O (2008) Probabilistic model checking of complex biological pathways. Theor Comput Sci 319(3):239–257
64. Heiner M, Gilbert D, Donaldson R (2008) Petri nets for systems and synthetic biology. In: Formal methods for the design of computer, communication, and software systems 8th international conference on formal methods for computational systems biology (SFM), vol 5016, LNCS, Springer, Berlin, pp 215–264
65. Holzmann GJ (2003) The Spin model checker: primer and reference manual. Addison-Wesley
66. Jha SK, Clarke EM, Langmead CJ, Legay A, Platzer A, Zuliani P (2009) A bayesian approach to model checking biological systems. In: Computational methods in systems biology. Springer, Berlin, pp 218–234
67. Klarner H, Streck A, Šafránek D, Kolčák J, Siebert H (2012) Parameter identification and model ranking of thomas networks. In: Computational methods in systems biology (CMSB), LNCS, Springer, Berlin, pp 207–226
68. Kwiatkowska M, Norman G, Parker D (2011) PRISM 4.0: verification of probabilistic real-time systems. In: Computer aided verification (CAV), vol 6806, LNCS, Springer, Berlin, pp 585–591
69. Legay, A., Delahaye, B., Bensalem, S (2010) Statistical model checking: an overview. In: Runtime verification, Springer, Berlin, Heidelberg, pp 122–135
70. Li Y, Albarghouthi A, Kincaid Z, Gurfinkel A, Chechik M (2014) Symbolic optimization with SMT solvers. In: POPL '14. ACM, pp 607–618
71. Liu B, Kong S, Gao S, Zuliani P, Clarke EM (2014) Parameter synthesis for cardiac cell hybrid models using $\delta$-decisions. In: CMSB, vol 8859, LNCS, Springer, Berlin, pp 99–113
72. Madsen C, Shmarov F, Zuliani P (2015) BioPSy: An SMT-based tool for guaranteed parameter set synthesis of biological models. In: CMSB'15, vol 9308, LNCS, Springer, Berlin, pp 182–194
73. Maler O, Batt G (2008) Approximating continuous systems by timed automata. In: Formal methods in systems biology (FMSB), LNCS, Springer, Berlin, pp 77–89
74. Maler O, Nickovic D, Pnueli A (2008) Checking temporal properties of discrete, timed and continuous behaviors. In: Pillars of computer science, Springer, Berlin, Heidelberg, pp 475–505
75. Mateescu R, Monteiro PT, Dumas E, de Jong H (2011) CTRL: extension of CTL with regular expressions and fairness operators to verify genetic regulatory networks. Theor Comput Sci 412(26):2854–2883

76. Meijer H, Dercole F, Oldeman B (2011) Numerical bifurcation analysis. In: Mathematics of complexity and dynamical systems, Springer, Berlin, pp 1172–1194
77. Mittnacht S (1998) Control of pRB phosphorylation. Curr Opin Genet Dev 8(1):21–27
78. Monteiro PT, Ropers D, Mateescu R, Freitas AT, de Jong H (2008) Temporal logic patterns for querying qualitative models of genetic regulatory networks. In: ECAI, vol 178, FAIA, IOS Press, pp 229–233
79. Nenzi L, Silvetti S, Bartocci E, Bortolussi L (2018) A robust genetic algorithm for learning temporal specifications from data. In: Quantitative evaluation of systems, Springer International Publishing, Cham, pp. 323–338
80. Niu W, Wang D (2008) Algebraic analysis of bifurcation and limit cycles for biological systems. In: Algebraic biology, AB '08, Springer, Berlin, pp 156–171
81. Pelánek R (2009) Fighting state space explosion: review and evaluation. In: Formal methods for industrial critical systems (FMICS), vol 5596, LNCS, Springer, Berlin, pp 37–52
82. Peled D (1988) Ten years of partial order reduction. In: Computer aided verification (CAV), LNCS, Springer, Berlin, pp 17–28
83. Priami C (2009) Algorithmic systems biology. Commun ACM 52(5):80–88
84. Raman V, Donzé A, Sadigh D, Murray RM, Seshia SA (2015) Reactive synthesis from signal temporal logic specifications. In: HSCC'15, ACM, New York, NY, USA, pp 239–248
85. Raue A, Karlsson J, Saccomani MP, Jirstrand M, Timmer J (2014) Comparison of approaches for parameter identifiability analysis of biological systems. Bioinformatics
86. Rizk A, Batt G, Fages F, Soliman S (2009) A general computational method for robustness analysis with applications to synthetic gene networks. Bioinformatics 25(12)
87. Sasagawa S, Ozaki Yi, Fujita K, Kuroda S (2005) Prediction and validation of the distinct dynamics of transient and sustained ERK activation. Nat Cell Biol 7(4):365–373
88. Schaub M, Henzinger T, Fisher J (2007) Qualitative networks: a symbolic approach to analyze biological signaling networks. BMC Syst Biol 1(1):4
89. Schivo DS, Scholma J, Wanders, B, Urquidi Camacho R, van der PV, Karperien H, Langerak R, van de JP, Post J (2012) Modelling biological pathway dynamics with timed automata. In: IEEE international conference on bioinformatics and bioengineering (ICBB), IEEE Computer Society, pp 447–453
90. Schivo S, Scholma J, van der Vet PE, Karperien M, Post JN, van de Pol J, Langerak R (2016) Modelling with animo: between fuzzy logic and differential equations. BMC Syst Biol 10(1):56
91. Schwarick M, Heiner M (2009) CSL model checking of biochemical networks with interval decision diagrams. In: Computational methods in systems biology (CMSB), vol 5688, LNCS/LNBI, Springer, Berlin, pp 296–312
92. Schwarick M, Rohr C, Heiner M (2011) MARCIE - Model checking And Reachability analysis done effiCIEntly . In: Quantitative evaluation of systems (QEST 2011). IEEE Computer Society, pp 91–100
93. Siebert H, Bockmayr A (2006) Incorporating time delays into the logical analysis of gene regulatory networks. Computational Methods in Systems Biology (CMSB), vol 4210. LNCS, Springer, Berlin Heidelberg, pp 169–183
94. Swat M, Kel A, Herzel H (2004) Bifurcation analysis of the regulatory modules of the mammalian G1/S transition. Bioinformatics 20(10):1506–1511
95. Yamada S, Taketomi T, Yoshimura A (2004) Model analysis of difference between EGF pathway and FGF pathway. Biochem Biophys Res Commun 314(4):1113–1120
96. Yovine S (1997) Kronos: a verification tool for real-time systems. Int J Softw Tools Technol Transf 1:123–133