# Model Checking of Biological Systems*

Luboš Brim, Milan Češka, and David Šafránek

Systems Biology Laboratory at Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic,
{brim,xceska,safranek}@fi.muni.cz

**Abstract.** Model checking together with other formal methods and techniques is being adapted for applications to biological systems. We present a selection of approaches used for modeling biological systems and formalizing their interesting properties in temporal logics. We also give a brief account of high performance model checking techniques and add a few case studies that demonstrate the use of model checking in computational systems biology. The primary aim is to give a reference for further reading.

## 1 Introduction

All biological systems, from single pathways to multicellular organisms, can be seen as complex systems of interacting components. Biological systems can also be seen as reactive systems, as they continuously interact with their environment. Systems biology thus studies complex *interactions* in biological systems, with the aim to understand better the processes that happen in such a system, as well as to grasp the emergent properties of such a system as a whole.

Computational systems biology can, by drawing upon mathematical approaches developed in the context of computer science and engineering [87, 144], contribute to the creation of powerful simulation, analysis and reasoning tools for working biologists. These tools can be used in devising new experiments and ultimately, for understanding functional properties of genome, proteome, cells, and organisms.

We are experiencing growing collaboration between biologists and computer scientists in the area of systems biology in recent years. This is because it has turned out that formal mathematical approaches to modeling and analysis, that have been developed for distributed computer systems and are referred to as formal methods, are applicable to biological systems as well as both kinds of systems have a lot in common.

* This work has been partially supported by the Czech Science Foundation grant No. GAP202/11/0312. M. Češka has been supported by Ministry of Education, Youth, and Sport project No. CZ.1.07/2.3.00/30.0009 – Employment of Newly Graduated Doctors of Science for Scientific Excellence. D. Šafránek has been supported by EC OP project No. CZ.1.07/2.3.00/20.0256.

In particular, automated formal verification (model checking) is one of the most promising formal methods that have the potential to be exploited in computational systems biology, because model checking is in principle an excellent methodology to verify/refute interesting biological hypotheses.

In this tutorial review, we would like to briefly describe some of the issues related to the application of model checking to the analysis of biological systems.

## 2 Setting the Context

### 2.1 Model Checking of Computer Systems

Model checking is a computer science and engineering technique that grew up from a purely academic research technique to a well-accepted industrial verification method. Nowadays, model checking is widely considered as an enhancement and complement to existing validation and verification techniques such as simulation and testing.

The roots of model checking lay in our never-ending quest to build computer systems that would be bug-free and correct. Our dependency on computer-based applications (both hardware and software) have motivated researchers to develop new techniques to increase our confidence in correctness of developed systems.

Testing is the basic verification technique that is widely used and extremely useful in practise. Another solution is to simulate the behavior of the system on a computer. Simulation does not work directly on the real system, but on a model. A model is an abstract representation of the real system. An advantage of simulation is that one does not need to build the real system and thus it is usually much cheaper than testing.

Both testing and simulation are widespread in industrial applications and their utilization has been shown to be very useful. One drawback, however, is that it is not possible, in general, to simulate or test all the possible scenarios or behaviors of a given system. That is, these techniques are in general not exhaustive and the failure cases may appear among those not tested or simulated.

Formal verification is a technique that complements testing and simulation. Even though the introduction of formal verification is rather costly, it pays off after all as it results in significant reduction in verification time as well as development costs and time-to-market. Attempts are being made to integrate formal verification techniques and tools with other design approaches to support engineering of complex industrial systems.

*Model checking* is a distinguished technique of formal verification of complex hardware and software designs. Founders of the technique, Edmund M. Clarke jr. (CMU, USA), Allen E. Emerson (Texas at Austin, USA), and Joseph Sifakis (IMAG Grenoble, France), were awarded ACM Turing Award in 2007 *for their roles in developing model checking into a highly effective verification technology, widely adopted in the hardware and software industries.* Unfortunately, the model checking procedure is computationally demanding and memory-intensive in general, hence, its applicability to large and complex systems routinely seen in

practise these days is still limited. The major hampering factor is the *state space explosion problem* [61] due to which large industrial models cannot be efficiently handled, unless more sophisticated and scalable methods are used.

A lot of attention has been paid to the development of approaches to fight the state space explosion problem in the field of automated formal verification [139]. Many techniques, such as state compaction [93], compression [107], state space reduction [140, 58, 81], symbolic state space representation [41], etc., were introduced to reduce the memory requirements needed to handle the verification problem with standard sequential algorithms. These techniques allowed to verify larger systems without the need of increased computing power.

However, for large industrial models, the state space, even if significantly reduced using the above mentioned techniques, does not completely fit into the main memory of a computer and hence the model-checking algorithm becomes very slow as soon as the memory is exhausted and the system starts swapping. A typical approach to dealing with these practical limitations is to increase the computational power (especially the amount of random-access memory) by building a powerful parallel computer as a network (cluster) of workstations. Individual workstations communicate through a message-passing interface such as MPI. Observed from outside, a cluster appears as a single parallel computer with high computing power and a large amount of memory. In recent years, a lot of effort has been invested into using parallel and distributed environments in order to solve the computational and space complexity bottlenecks in model checking and therefore we devote a special section to review some parallel and distributed approaches (Section 3.4).

### 2.2 On the Role of Model Checking in Systems Biology

There are many ways how we can improve correctness of computer systems. The used methods and techniques are generally classified as *verification* and/or *falsification* approaches. The role of verification techniques, typically theorem proving, is to guarantee there is no bug in the system while the role of falsification techniques, typically testing, is to demonstrate the presence of errors. Model checking is primarily a verification technique which is, however, often used for falsification (as a bug hunting method).

We might tend to a similar position of model checking when applied to biological systems. The situation is, however, different for many reasons (see Fig. 1). The most important difference is that in biology, the system under investigation already exists. It is not the primary role of biology to create life (at least to some extent). On the other hand, computer systems are man-made. In computer engineering the models are used as abstractions that are step-wise transformed into the final system. This contrasts to experimental sciences where models serve as hypotheses. The role of verification in computer engineering is to ensure the system that was constructed from the model has the same behavior as prescribed by the model. If the verification fails, the system has to be corrected. In the case of successful verification, we are done – the engineer has completed his task. On the other hand, in experimental sciences the goal is to show that the hypothesis

correctly captures some aspects of the real system. Scientific ideas are tested by generating multiple possible hypotheses, coming up with predictions for each of them, and then designing tests (experiments) by which we can falsify the hypotheses. Typically, we test hypotheses in order to refute them, not to try to support them. In computer engineering the model is thus always correct, while in science the system is.

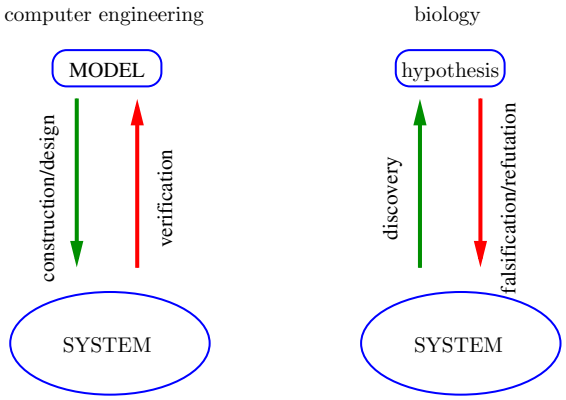



**Fig. 1.** Knowledge discovery in biology and computer engineering

Now let us have a closer look at the possible position of model checking in computational systems biology. Systems biology can be characterized as an approach to the understanding of life through the study of how the properties of biological systems arise through interactions between the system components [137]. From this point of view, biological systems are similar to complex computer systems. Namely, in both kinds of systems the interaction of components is a source of various emergent system properties that are not explicitly encoded in individual system parts. The common problem related to the analysis of such systems is that the emergent properties are difficult to identify and quite often hard to understand because the causes and effects are not obviously related.

For complex parallel and distributed software and hardware systems the process of detection and analysis of emergent properties relates closely to the process of formal verification. It is often the case that the emergent properties of distributed systems, such as deadlocks or non-progressive cycles, are properties that the designers of the system have not the intention to introduce. Methods of automated formal verification, model checking in particular, can be thus used to detect such properties and to prove their absence.

Going beyond verification and/or falsification of properties of biological systems, there are many other interesting questions having sometimes no direct counter-part in computer systems, that can be solved by application of model checking techniques. An example is the problem of *parameter identification* (also called parameter estimation or model calibration). Parameter identification is a

key issue in systems biology, as it represents the crucial step to obtaining better models of biological systems that give more precise predictions. This issue is usually addressed by fitting the model simulations to the observed experimental data. In biological models, the control parameters used to define the behavior of models are kinetic or rate parameters. Some of these parameters usually cannot be experimentally determined which leads to the need to estimate these parameters by computational methods. To this end, model checking provides a promising alternative to fitting – parameters can be identified or fine tuned to satisfy given set of properties. Parameter identification by model checking has been referred to as parameter synthesis [76, 13, 25].

In [40, 147, 76], comprehensive parameter exploration techniques are introduced. They are based on the construction (usually approximation) of a landscape function that maps every model parametrization to a value quantitatively characterizing validity of the properties. Landscape function has direct application in robustness analysis. Robustness can be understood as a feature of a system to maintain a property in the face of parameter perturbation.

## 3 Description of Technique and Tools

In this section we give more technical details about models used in systems biology and their biological properties. We also introduce some parallel and distributed approaches to model checking as high performance techniques to support analysis of complex biological systems.

### 3.1 Models of Biological Systems

Most of the models currently developed in systems biology focus on complex interactions among system components. State-of-the-art biological knowledge is being reconstructed and organized in the form of *biological networks*. Biological networks are built from biological knowledge databases, experimental data and generally understood principles based on many simplifying assumptions. There are two fundamental types of biological networks – *reaction networks* and *regulatory networks*. Recent network reconstructions typically mix the two. Reaction networks provide a detailed view of underlying biochemical interactions – nodes are chemical species and stoichiometry-labeled (multi-)edges represent elementary chemical reactions. Regulatory networks are higher level and focus on feedbacks among individual system components – nodes are species or abstract biological objects and edges represent positive or negative influence. Gene regulatory networks make a typical example [115].

Computational systems biology studies the dynamics of biological networks, in particular, how a population of components affected by network interactions evolves in time. To this end, *biological model* is defined as a biological network associated with a suitable semantics reflecting the system dynamics at a particular level of abstraction. The semantics fulfils the following tasks:

- Network components are given a mathematical interpretation as *variables* (numbers of molecules or molar concentrations),
- Network interactions are given a mathematical interpretation as *rules* specifying dynamical changes in variables.

Both variables and rules can be understood as model quantities modeled at different levels of abstraction. Variables can be treated as either discrete or continuous. *Discrete-value* semantics can capture either a *microscopic or mesoscopic view* of biological particles (e.g., number of molecules) or abstract qualitative interpretations of selected qualities of modeled components (e.g., absence/presence of a species). *Continuous-value* semantics represents a so-called *macroscopic view* where the modeled objects are expected to appear in large quantities provided that it is inconvenient to distinguish small differences (e.g., molar concentration of a species in a cell).

Quantities that can be associated with rules are time and probability. Since each interaction occurs in time with a specific *rate*, the respective rule is executed with this rate implying the inherent time aspect of the system dynamics. Naturally, time is considered as continuous, dense quantity. When the information on rate is unknown or abstracted out due to simplifications, *discrete-time* semantics is employed. It deals with the shortest (discrete) time step which can represent an arbitrary finite time horizon. Discrete-time abstraction allows to treat qualitative models as *untimed*, i.e., the exact duration of a single time step is left unspecified. It is worth noting that in the most of cases the occurrence of any rule is modeled as instantaneous and it occurs immediately after the conditions for occurrence are satisfied. There exist models that refine these aspects of semantics (e.g., delayed interactions [45] or non-instantaneous interactions [11]).

With respect to execution of interactions, rules can be either *deterministic* or *stochastic*. Deterministic rules represent interactions that occur each time all preconditions are satisfied (e.g., if there is a non-zero amount of all reactants, the reaction occurs). There is no noise affecting the interaction. Stochastic interactions reflect noisy environment by assuming a certain probability with which they occur.

Finally, there is yet another notion of quantity that can enhance the model semantics. In particular, interactions and even variable values can be assigned quantitative costs and *rewards*, e.g., time spent in particular concentration levels, energy consumed by particular reactions, etc. By adding this kind of information (if available), models can be adjusted to provide interesting and detailed quantitative predictions resulting from complex dynamics.

Types of semantics mentioned above can be suitably combined resulting in several classes of models varying in the level of abstraction employed, as is overviewed in Fig. 2. On the right side of the scheme, there are models considering continuous component quantities and deterministic interactions. These inherently quantitative models are currently the most widely used in computational systems biology since they have deep roots in mathematical biology. In fact, from the semantics point of view they are purely denotational [87] and thus we call them *mathematical*. On the left side of the scheme, there are discrete-

value models which can be either quantitative (incorporating real-time and/or stochasticity of interactions) or qualitative (abstracting from the timed nature and stochasticity of interactions). These models are closer to computer science or they directly originate from computer science. Fisher and Henzinger [87] classify these models as *executable*, since for any of them the semantics can be considered either denotational or operational. The operational view allows to understand biological systems in the similar way as programs or any formal models in computer engineering. That way these models naturally bring the model checking technique to biology.


**Fig. 2.** Model types sorted according to different level of details captured in their semantics.

Mathematical models are used to represent actual quantitative relations between components in the system. Generally, a system of ordinary differential equations (ODEs) [118, 110] and/or differential algebraic equations (DAEs) [37] is used to represent the interaction and processes among the various components. Determinism and continuity reflect the modeled phenomena in high chemical species concentrations or large cell populations (the macroscopic level) while completely neglecting the noise and differences in individual components and interactions. These models can be simulated, analyzed, and possibly solved, but require detailed knowledge of the biological system, i.e., *quantitative parameters* identifying the physical aspects of system interactions (e.g., kinetic coefficients of chemical reactions).

On the other hand, executable models employ abstract representations to explain biological phenomena. Examples of widely used formalisms are Boolean networks [159, 50, 121], Petri Nets [103, 32, 49, 143], timed automata [27, 153, 97], compact process algebraic representations such as BioPEPA [56], Kappa [68] or

suitable adaptations of $\pi$-calculus [141, 145]. These formalisms have an inherent execution scheme attached to the models, and relate different qualitative configurations (*states*) of model components to each other. The relation among states can be either qualitative or quantitative (with real-time bounded or even stochastic rules). The advantage is the capability to effectively represent the logic behind biological systems dynamics without precise quantitative knowledge about the component interactions. Executable models are inherently discrete provided that the dynamics (execution) occurs in terms of a series of discrete events. In the untimed setting, nondeterminism allows to capture all possible "timings" (orderings) of concurrent events. To quantitatively differentiate among all possible executions in a particular state, rules can be assigned probabilities, resulting in discrete stochastic models [36, 163] most typically represented by discrete-time Markov chains (DTMC). When set appropriately, executable models can be used at any level of view of biological systems dynamics.

Stochastic models allow to incorporate noise which causes fluctuations in component quantities and that way affects the biological system dynamics [80, 120]. In physics, chemistry and related fields, the probabilistic time-evolution of a system with discrete component quantities is described by so-called master equations. In the case of biological phenomena, the chemical master equation (CME) provides an exact mathematical model for stochastic dynamics [95]. It is formalized as a set of differential equations, providing a denotational representation of component quantities distribution in continuous-time. Gillespie [94, 96] has made an important breakthrough in stochastic modeling by introducing techniques for exact simulation of CME. From the computer scientific viewpoint, the CME can be equivalently represented by continuous-time Markov chains (CTMC) which provide operational semantics and allow us to consider continuous-time stochastic models as executable [72].

Although outside the scope of this paper, it is worth mentioning that a significant and general class of models is that of *hybrid models* most typically represented by means of hybrid automata [105] or process algebraic techniques [90, 34]. Hybrid models allow to mix discrete-value components with continuous-value components and discrete-time dynamics with continuous-time dynamics. Such a complicated semantics limits the model analysis [106, 44]. Hybrid models can be satisfactorily used for modeling and simulation [75] and, when simplifying assumptions are employed (e.g., considering linear dynamics of continuous components), also for a more advanced analysis of biological processes [67, 89]. To incorporate noise, stochastic hybrid models [154] (i.e., stochastic hybrid automata) allow both discrete-time and continuous-time dynamics to evolve randomly. Coupling of both kinds of dynamics while keeping their stochasticity complicates analysis even more. To this end, simulation-based (statistical) [70] or fluid-flow approximation techniques [34, 65] are typically employed.

**Model Simplification** It is important to note that component quantities in biological models most typically do not evolve unlimitedly. In particular, concentration (or number of molecules) is always limited by degradation processes.

However, it might not be easy to identify the bounds without a deeper analysis of the model. From the context of an observed phenomenon and the time-scale of relevant model behavior, it can be possible to identify the time-horizon (or the number of steps in the untimed case) for which it is guaranteed that the phenomenon occurs. Even periodically repeating phenomena, e.g., circadian clock, can be approximately detected and analyzed in finite time in the order of an appropriately selected time-scale. Again, a non-trivial analysis has to be performed in some cases to estimate or overapproximate correctly the time horizon. In any case, some hypotheses on maximal (extreme) bounds on component quantities or time can be always considered.

From the computational point of view, there exist many well developed and efficient techniques for exhaustive analysis of models appearing in the top left quadrant of the scheme in Fig. 2. The assumptions stated above imply finite number of model states. However, models in other quadrants incorporate continuous or dense quantities which significantly complicate or even disallow the direct exhaustive analysis. We focus on continuous-time discrete-value models first. Reduction of timed automata into untimed finite automata [2] is the crucial procedure enabling exhaustive analysis for continuous-time discrete-value models. A continuous-time stochastic model represented by a continuous-time Markov chain is reduced to a discrete-time Markov chain and a Poisson process (or a birth process) by uniformization techniques [161, 72]. All reductions at this level are exact, provided that no information is lost.

There are techniques to abstract (or approximate) continuous-value models by discrete-value models. Formally defined abstractions allow specific properties to be preserved by means of over-approximation (resp. under-approximation) of model behavior. However, the effect of behavior spuriously added (resp. lost) by the abstraction is usually large. Over-approximative abstractions are conservative in the sense that each execution of the original model is also present in the abstract model but there can appear a new behavior, not present in the original model. Underapproximative abstractions ensure that no execution is added to the abstract model but some can be ignored.

Besides formal abstraction, there are approximations that distort the original behavior rather than adding or removing some. Such approximations do not guarantee preservation of dynamics properties but the deviation of behavior is ensured not to exceed a certain (specified) approximation error.

In the case of continuous-time deterministic models, typically non-linear, the most widely used approximation is provided by numerical simulation (integration) methods. For certain classes of ordinary differential equations, there are also formal abstraction techniques providing a discrete-time discrete-value over-approximation in terms of non-deterministic finite automata [116, 30, 99, 62] or a continuous-time discrete-value over-approximation in terms of timed automata [133]. In the former case, the extent of falsely added executions is usually large whereas the latter case prevents addition of any executions with non-realistic timing and therefore the number of false executions can be reduced.

Some classes of continuous-time deterministic models can be also approximated by continuous-time stochastic models provided that continuous-value variables are approximated by a suitable number of uniformly distributed discrete levels. If calibrated properly, averaged stochastic executions converge to the deterministic solution [42] (see [103] for tutorial).

In reverse direction, some classes of continuous-time stochastic models can be approximated by deterministic continuous-time models (ODEs) by means of fluid-approximation techniques [69]. Based on these techniques, more sophisticated analysis methods for stochastic models, combining fluid-approximation with CTMC analysis have recently been proposed [33]. The advantage of these techniques is that they avoid the state-explosion problem.

In this text we focus on well-established methods developed for discrete- and continuous-time discrete-value models, in particular, we consider techniques targeting the exhaustive analysis of temporal properties of systems dynamics based on qualitative and quantitative model checking. Brief description of concrete techniques is presented in Section 3.3. Properties of biological interest are described in Section 3.2. Examples of models and the application of model checking techniques is presented on several case studies in Section 4.

**Model Parameters** Biological model of any type is determined by a fixed topology (biological network) where the interactions (rules) are parametrized. Parameters provide degrees of freedom in which the model dynamics can be adjusted. In contrast to the network topology which stands on common principles, finding a correct model parametrization is a non-trivial task which makes a critical part of the so-called *inverse problem* [82].

Parameters appear in all kinds of models. In the case of qualitative models, a parameter most typically affects the logic behind a rule, i.e., adjusting the effect of the respective interaction on model components. Sets of possible parmeter values (*parametrizations*) for qualitative models are finite and discrete, but can be very large (e.g., the number of parametrizations for setting the dynamics of a gene $A$ in a Boolean model of a gene regulatory network is exponential wrt the number of genes affecting the expression of $A$). In quantitative models, parameters represent the quantitative aspects associated with the semantics of rules, i.e., how the respective interaction evolves in time. In continuous-time stochastic (resp. deterministic) models the parameters describe the rate (resp. velocity) of respective interactions. In real-time models, the parameters describe time delays (minimal or maximal) between particular interactions. In all these cases, parametrization sets are are uncountable and bounded by laws of physics.

### 3.2 Biologically Relevant Properties

With respect to the nature of phenomena generated by dynamics of biological processes, typical properties studied on biological models can be organized into six elemental categories: reachability properties, temporal ordering of events, variable correlations, (multi)stability properties, monotonic trends, and oscillation properties. To reason about the model types presented in the previous subsection, properties have to be expressed with different levels of detail.

**Qualitative Properties** *Qualitative properties* abstract away from any quantitative information like time aspects or energy costs of targeted biological phenomena. Qualitative properties are in general interpretable on all types of models, especially on untimed discrete-value models. Computer science offers two basic logical formalisms allowing to express qualitative properties of systems dynamics: linear-time temporal logics, interpreted on individual model executions (paths), and branching-time temporal logics, interpreted on trees of (non-deterministically) branching model executions.

The basic linear-time temporal logic is Linear Temporal Logic (LTL) [142]. LTL has been proved to be the basic formal language that is most suitable for qualitatively expressing properties of the six elemental categories. LTL can be interpreted on all kinds of models.

The most basic branching-time logic is Computational Tree Logic (CTL) [57]. In contrast to LTL, CTL allows to reason about non-determinism and, therefore, is used for properties dealing with non-determinism. CTL is also interpretable on all kinds of models, yet, the expressiveness of CTL is limited in the case of deterministic models. Below we give several examples of biologically-relevant properties. Unless otherwise mentioned, these properties will be expressed in LTL.

Qualitative *reachability properties* express reachability of specified concentration levels in given model variables. For example, the formula $\mathbf{F}(2 \leq B \leq 3)$ expresses the property that $B$ reaches the concentration level between 2 and 3 at some point during the progress of the model dynamics. The linear-time formula $\mathbf{F}\varphi$ containing the operator $\mathbf{F}$ (Future) has the intuitive meaning that, on a given path, there must eventually exist a state where $\varphi$ is satisfied. Note that the property tells nothing regarding the moment at which the event occurs. Reachability properties are useful especially for expressing global bounds of reachable concentration values.

To capture the qualitative temporal patterns in the dynamics of inspected variables, the properties expressing *temporal ordering of events* are used. These properties are based on linear-time operator $\mathbf{U}$ (Until), i.e., the formul $\varphi_1 \mathbf{U} \varphi_2$, with an intuitive meaning that, on a given path, $\varphi_2$ must eventually hold in some $i$th state on the path and for all states from the beginning of the path until the $i$th state, $\varphi_1$ must hold. An example of such property is the formula $(A \leq 2)\ \mathbf{U}\ [(2 < A \leq 5)\ \mathbf{U}\ (A > 5)]$ representing the following temporal pattern: species $A$ is initially kept below 2 until it reaches 5 and finally exceeds 5.

*Variable correlations* make important observations revealing cooperations and dependencies in biological processes, e.g., co-expression of certain genes. These properties can be expressed by combining several temporal ordering formulae into a single formula using traditional logical operators. Following this approach, mutual dependencies in the dynamics of inspected variables can be captured. For example, the formula $[(A \leq 2)\ \mathbf{U}\ ((2 < A \leq 5)\ \mathbf{U}\ (A > 5))] \Rightarrow [(B \geq 10)\ \mathbf{U}\ ((5 \leq B < 10)\ \mathbf{U}\ (B < 5))]$ expresses the following correlation in concentration of species $A$ and $B$: if $A$ increases according to the temporal pattern from the previous paragraph then $B$ decreases from a level above 10 to a level below 5.

A specific kind of temporal properties deals with the analysis of presence of stable concentration levels. An example of an elementary *stability property* is the formula $\mathbf{G}(A \leq 2)$ stating that concentration below 2 is stable (attractor) for species $A$. The formula $\mathbf{G}\varphi$, with the operator $\mathbf{G}$ (Globally), expresses the requirement that $\varphi$ must hold in each state of a given path, the intuitive meaning is "forever". Stability properties can be effectively combined with reachability properties and relativized with respect to a specific initial condition. For example, the formula $(A \geq 0) \Rightarrow \mathbf{FG}(A \leq 2)$ states that the stable concentration below 2 is reached from any non-negative initial concentration of $A$. To query for existence of several different stable states (multi-stability), the LTL formula $[(A \leq 5) \Rightarrow \mathbf{G}(A \leq 5)] \wedge [(A > 5) \Rightarrow \mathbf{G}(A > 5)]$ can be employed. It expresses the fact that there are two different stable concentration levels in the dynamics of $A$: the first is below the level 5 and the second is above 5. Note that this formula expresses only the existence of the two stable attractors, 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 this end, 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 hold in only those states. Therefore, the bistability formula is satisfied in every state from which the execution can eventually branch into both attractors.

Important observations of biological dynamics are *monotonous trends* in system variables [9]. Monotonicity is an indicator of robust increasing or decreasing phases observed in individual species dynamics. In the qualitative setting interpreted on discrete-value models, the non-strict monotonicity can be expressed as a special case of temporal ordering property, e.g., $(A = 1)\ \mathbf{U}\ [(A = 2)\ \mathbf{U}\ (A = 3)]$.

Finally, an interesting dynamics phenomenon appearing in biology is *oscillation*, e.g., circadian rhythms. A simple example of an oscillation property is expressed by the formula $(\mathbf{G}[(A \leq 3) \Rightarrow \mathbf{F}(A > 3)]) \wedge (\mathbf{G}[(A > 3) \Rightarrow \mathbf{F}(A \leq 3)])$ representing a permanent oscillation of $A$ around the concentration level 3. Oscillation properties require linear-time operators, they cannot be expressed in CTL. Finer specification of oscillations can be realized by extending the formula with additional constraints identifying the qualitative aspects of the oscillation, e.g., the maximal and minimal amplitude levels.

**Quantitative Properties** Quantitative properties including time aspects, energy consumptions and the stochasticity of a system are essential in the analyses of the dynamics of biological systems. Hence, a wide variety of logical formalisms allowing to reason about quantitative system aspects have been used to study biological systems. These formalisms usually extend the aforementioned logics and can be roughly divided into deterministic and stochastic logics. Deterministic logics are mostly focusing on a quantitative notion of time. The time extension of CTL called Timed Computational Tree Logic (TCTL) has been introduced in [2] and its simplified version is used as a specification language in the tool UPPAAL [29]. The extension allows to specify additional clock constrains, e.g, the TCTL formula $\mathbf{EF}(\varphi \wedge t \leq 3)$, where $t$ is a clock, requires the existence of

an execution branch where $\varphi$ is satisfied within 3 time units. A popular dense time extension of LTL is Metric Interval Logic (MITL) introduced in [3] as a restriction of Metric Temporal Logic (MTL) [123]. It is based on the timed until modality $\mathbf{U}^I$ where the interval $I$ is a nonempty convex subset of $\mathbb{R}_{\geq 0}$. The formula $(A = 1)\ \mathbf{U}^{[a,b]}\ (A = 2)$ is satisfied at any time instant $t$ such that $A = 2$ at some $t' \in [t + a, t + b]$, and $A = 1$ continuously from $t$ to $t'$. Another time extension of LTL called Timed Propositional Temporal Logic (TPTL) [4] is based on freeze-quantification where extra clocks are used to specify temporal constraints. These clocks can be reset at some point and later we can compare their values to some integers. The TPTL formula $\mathbf{G}[(A = 1) \Rightarrow x.\mathbf{F}(B = 3 \wedge x \leq 5)]$ expresses that whenever the population of species $A$ reaches 1, the population of species $B$ will reach 3 in 5 time units.

Motivated by the application of verification and monitoring techniques to continuous-value and hybrid systems, Signal Temporal Logic (STL) has been introduced [134]. It combines the dense time modalities of MITL with the numerical predicates over real numbers. The predicates are given as a real-value signal describing the evolution of the system, e.g, a function from time to a Cartesian product over reals. The formula $\mathbf{G}^{[0,300]}[(x_1 > 0.7) \Rightarrow F^{[3,5]}(x_2 > 0.7)]$, where $x_1, x_2$ are some signals, expresses that for each time point $t \in [0.300]$ it holds that if the value of the signal $x_1$ in $t$ is greater than 0.7 then there exists time $t' \in [t + 3, t + 5]$ such that the value of the signal $x_2$ in $t'$ is also greater than 0.7. For example, the tool Breach [76] employs STL to define temporal logic formulae and check whether they are satisfied on simulated trajectories. A version of LTL with constraints over the reals, named LTL($\mathbb{R}$), has been proposed in [5] to express the temporal properties of molecular concentrations and their derivatives. The quantifier free fragment of the first-order extension of LTL($\mathbb{R}$), named QFLTL($\mathbb{R}$) has been considered in [86]. It allows to use free variables in the atomic propositions and, thus, it enables to analyze numerical data time series in temporal logic and to automatically compute LTL($\mathbb{R}$) specifications from experimental traces. The formula $\mathbf{F}(A \geq p)$ expresses the question what threshold $p$ species $A$ attain in the trace. These two extensions of LTL are used in the tool BIOCHAM [84] to formalize numerical temporal properties.

Stochastic logics provides means to specify the probability and performance measures on Markov chains. In the case of DTMCs, a probabilistic extension of CTL, named PCTL, can be employed [100]. The logic is based on the probabilistic operator $\mathbf{P}_{\sim p}[\phi]$ expressing that the probability of the path formula $\phi$ being satisfied from a given state meets the bound $\sim p$. As a path formula it allows standard bounded and unbounded temporal operators. Note that, PCTL is a discrete-time logic and thus the path formulae are interpreted over discrete time steps. The PCTL formula $\mathbf{P}_{\geq 0.9}[\mathbf{F}^{\leq 5}(A = 3)]$ expresses that the probability that the population of species A will be equal to 3 within 5 time steps is at least 0.9.

To formalize properties of CTMC, Continuous Stochastic Logic (CSL) [6] has been introduced. It is a probabilistic extension of CTL with continuous-time semantics. In contrast to PCTL, the path formulae in CSL uses an interval of non-negative reals, rather than simply an integer upper bound. The CSL formula $\mathbf{P}_{\geq 0.9}[\mathbf{F}^{[1,2]}(A = 3)]$ expresses that the probability that the population of species A will be equal to 3 between 1 and 2 time units is at least 0.9. The logic also

includes the steady-state operator **S** describing the steady-state behavior of a CTMC. The CSL formula $\mathbf{S}_{\leq 0.05}[A > 10]$ expresses that the long run probability that the population of species A will be higher than 10 is at most 0.05.

To further broaden the scope of possibly expressible behavior, PCTL and CSL have been extended to allow the specification over reward-based stochastic models, i.e., Markov chains with real-valued rewards/costs attached to states and transitions [126]. The extension enables to express properties such as the expected time a system spends in a specified set of states over a time interval or the expected number of times that a particular reaction occurrs.

The only way to combine temporal operators in PCTL and CSL is to use a nested formula whose meaning can be too subtle. Therefore, a probabilistic extension of LTL has been introduced in [64] allowing to express the probability of more complex events. The semantics of the logic is defined over Markov Decision Processes (MDPs) [71] which are a widely used formalism for modeling systems that exhibit both probabilistic and nondeterministic behavior, see e.g., [88] for more details.

Expressing biological phenomena can require extensions of existing logics. Biologically relevant temporal logic extensions target precise quantitative description of oscillations [74, 24] or qualitative properties combining linear-time properties with branching-time [136].

### 3.3 Model Checking Techniques for Analysis of Biological Systems

Model checking techniques for the analysis of biological systems can be roughly divided into *exhaustive techniques* and *monitoring techniques*. The exhaustive techniques consist of checking whether all executions – state-event sequences – generated by a given system $\mathcal{S}$, satisfy the inspected property described as the formula $\varphi$, i.e., they effectively decide the language inclusion $\|\mathcal{S}\| \subseteq \|\varphi\|$ ($\|\varphi\|$ is the set of all executions that satisfy $\varphi$). In order to generate all executions, the whole state-space has to be stored and evaluated. This is why the exhaustive techniques generally suffer from the state-space explosion problem. There exist several techniques allowing to reduce this problem, e.g., efficient symbolic representation, state-space reductions or iterative abstraction refinement. For systems which are outside the scope of exhaustive techniques, either due to the incorporation 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 the inclusion, test the monitoring techniques are based on the membership test $\omega \in \|\varphi\|$ of an individual simulation trace $\omega \in \|\mathcal{S}\|$, where the responsibility for exhaustive coverage is delegated to the procedure that generates the traces. The key observation behind their efficiency is that for large and complex systems, the simulation is generally easier and faster than building a concise representation of global transition systems required for the exhaustive model checking approach. However, since a single simulation generates a single trajectory out of all the possible executions of a system, usually the average values among several simulations need to be considered to achieve the necessary level of confidence in the results obtained.

A possible way to improve the accuracy of monitoring techniques is to employ the statistical model checking that addresses general stochastic systems in terms of statistical inference. It samples the behaviors (simulations) of a model, verifies their conformance with respect to a temporal formula (i.e. performs the membership test), and finally applies a statistical estimation technique to compute an approximate value for the probability that the formula is satisfied. The accuracy of statistical model checking is affected by the accuracy of stochastic simulations techniques that are employed and also by the structure of the model or more precisely by the level of details (initial conditions, parameters, etc.) we have about the system under study.

Exhaustive model checking, statistical model checking and monitoring techniques have been applied to the study of biological systems. They allow researchers to make predictions and test hypotheses on models of different kinds (see Fig. 2). For deterministic models with continuous-value semantics the exhaustive techniques cannot be used due to the infinite number of possible executions. Therefore, advance monitoring techniques for various temporal logics have been designed in order to analyze complex non-linear systems, see [135] for a survey. These techniques have been further extended for application in systems biology. For example, the tool Breach [76] provides a coherent set of simulation-based techniques aimed at the analysis and parameter identification of deterministic models of complex biological and hybrid systems. Its primary features facilitate the computation and the property investigation of a large set of trajectories and also provide information about the sensitivity with respect to parameter perturbations. A successful application of this approach to systems biology has been demonstrated in [77] where a model of the acute inflammatory response to bacterial infection is analyzed.

A similar extension to monitoring techniques has been proposed in [86] where the authors generalize the trace-based model checking algorithm [43] to a constraint solving algorithm for QFLTL($\mathbb{R}$) with numerical constraints over the reals. Given an ODE model and a temporal property to verify within a finite time horizon, the computation of a finite simulation trace by numerical integration provides a linear Kripke structure (each state has a single successor). Afterwards, the QFLTL($\mathbb{R}$) generalization provides the ability to compute those instantiations of a formula that are true in a finite trace, by giving the complete domain of the real-valued variables occurring in the formula for which it is true. This approach has been implemented in the tool BIOCHAM [84]

Techniques for the verification of a temporal logic property against stochastic models can be either exact, based on probabilistic model checking, or approximate, based on statistical model checking using stochastic simulation such as Gillespie's algorithm [94] or Monte Carlo sampling [114, 8]. Probabilistic model checking answers quantitative temporal queries by performing an exhaustive exploration of all the possible paths through the system. The probabilistic model checking techniques can be roughly divided into the techniques for discrete-time and forcontinuous-time systems. A discrete-time system is usually described by discrete-time Markov chain (DTMC) where the transitions between the states

are governed by a probability distribution. The inspected properties of such systems are mostly specified in PCTL. The model checking algorithm for PCTL over DTMC constructs the parse tree of a given formula $\Phi$ and for each node it recursively computes the set of states satisfying the corresponding subformula. For more details, see, e.g., [64].

As mentioned in Section 3.1 a continuous-time system is usually described by a continuous-time Markov chain (CTMC). While each transition between states in a DTMC corresponds to a discrete-time step, transitions in a CTMC occur in real time. The transitions between the states in CTMC are governed by the transition rate matrix. It assigns a rate $\lambda$ to each pair of states in the CTMC, which are used as parameters of the exponential distribution, i.e., the probability of the transition being triggered within $t$ time-units equals $1-e^{-\lambda \cdot t}$. To reflect the real time aspects, the inspected properties of such systems are mostly specified in CSL. Efficient model checking algorithm for CSL over CTMC has been proposed in [7]. It reduces the model checking problem to the transient analysis, i.e., to the computation of transient probability, having started in state $s$, of being in state $s'$ at time instant $t$. The reduction is based on a modification of the rate matrix such that certain states are made absorbing (all outgoing transitions are ignored) according to their validity with respect to the inspected formula. A standard technique for computing transient probabilities is based on uniformization. The key idea is for a given CTMC to construct the uniformized DTMC where all exponential delays in the CTMC are normalized with respect to the fastest transition rate $q$. Then each step of the uniformized DTMC corresponds to a single exponentially distributed delay with the parameter $q$. The $i$th matrix power of the uniformized DTMC gives the probability of jumping between each pair of states in the DTMC in $i$ steps. The transient probability in time $t$ is computed as the sum of the matrix powers weighted by Poisson probabilities giving the probability of $i$ such steps occurring in time $t$. For more details about the probabilistic model checking techniques, see, e.g., [126].

The exact probabilistic model checking suffers from the state-space explosion problem, which is even more critical than in the non-probabilistic case. Therefore, for systems with too many states (more that $10^{10}$) the described techniques become intractable. As result, additional reduction techniques or the statistical model checking have to be used in order to effectively analyze complex biological systems. In [42], the authors consider signal transduction in the RKIP-inhibited ERK pathway. They overcome the state-space explosion problem of probabilistic model checking by rescaling model component quantities to lower numbers of population levels. Probabilistic model checking has also been employed to the analysis of gene regulatory circuits where an automatized translation of models into a CTMC, based on quasi-steady-state approximation (QSSA), has been proposed [132].

The main problem with statistical model checking is caused by rare events, i.e., temporal formulae whose satisfaction probability is very small. When estimating the probability of such formulae, the number of simulations needed to ensure a good estimate becomes unfeasible. In [60], the authors show that the im-

portance sampling, a variance-reduction technique for the Monte Carlo method, and the cross-entropy method, a general Monte Carlo approach to combinatorial and continuous multi-extremal optimization and importance sampling, can efficiently address this problem. They use Bounded Linear Temporal Logic, a variant of LTL where the temporal operators are equipped with time bounds, to reason about biochemical reactions in systems biology.

Both exact and approximate model checking techniques have been implemented in several tools, e.g., PRISM [125], MARCIE [152]. These tools have been successfully employed in the analysis of biological systems, e.g., in [102] the authors apply PRISM to analyze the complex FGF (Fibroblast Growth Factor) signalling pathway, in [151], the authors analyze stochastic Petri nets model of a biological network using efficient state-space representation based on interval decision diagrams. Advanced techniques for exact CSL model checking that allow to reduce the state-space explosion problem for some classes of biological systems have been implemented in the prototype tool SABRE [73].

For real-time models, model checking techniques are based on transforming the uncountable continuous-time model into an equivalent finite discrete structure (the so-called zone automaton). The two main real-time model checking tools, UPPAAL [29] and KRONOS [164], have also been used for the analysis of biological models. In the case of UPPAAL, applications to gene regulatory networks [153, 97] and signaling pathways [150] have been realized. KRONOS was applied to gene regulatory networks [27] and to real-time abstractions of continuous-time deterministic models [133].

At the end of this section we briefly introduce model checking techniques for qualitative models of biological systems. These techniques have been extensively studied, see [59] for a good starting point, and there also exist several matured tools providing their efficient implementations.

Application of the qualitative model checking to systems biology is highly-relevant for Boolean models of genetic regulatory networks [50, 31] and signaling networks [149, 79], provided that symbolic verification techniques are usually employed. In [43], the tool BIOCHAM is used to verify the qualitative properties (specified in CTL) of asynchronous state transitions with Boolean semantics using standard symbolic model checker NuSMV [55].

Explicit model checking techniques are used in [121] where the authors propose new methodology for parameter identification and the analysis of discrete gene networks based on colored LTL model checking [13]. They improve the standard automata-based algorithm for LTL model checking [160] that consists of the following steps. The inspected LTL formula $\varphi$ is negated and translated into a Büchi automaton $A_{\neg\varphi}$ describing all the executions violating $\varphi$. Afterwards, the synchronous product of $A_{\neg\varphi}$ and a finite state automaton describing the system under study is constructed. The system satisfies the formula $\varphi$ if and only if the language of the product automaton is empty, which is if and only if there is no reachable accepting cycle (cycle containing an accepting state) in the underlying graph of the product automaton. Instead of employing this standard algorithm for each possible parametrization individually, the authors propose

a heuristics reducing the computation effort by means of operating on entire parametrization space. A concrete application of these techniques is presented in Section 4.

Another formal method for qualitative analysis of biological systems is presented in [103], where Petri nets have been used to describe the mitogen-activated protein kinase. The authors study general properties (boundedness, liveness, reversibility, invariants), structural properties (reflecting the modeling approach) and also properties specified in temporal logic.

Model checking is also employed to qualitative abstractions of quantitative models, examples are given in Section 4. Techniques for finite discrete abstraction of the continuous state space are used in the tool BioDiVinE [18] to analyze biological models specified in terms of a set of chemical reactions. Chemical reactions are transformed into a system of multi-affine differential equations that are further discretized to a finite state automaton in order to employ the standard LTL model checking techniques including property-driven parameter identification.

### 3.4 Parallel and Distributed Model Checking

As already stated above model checking is a computationaly demanding procedure and techniques to fight the state explosion problem are an unavoidable ingredient of it. To verify even larger systems, however, no option was left out than to employ combined computing power of multiple computing devices. Attempts to use hard drives or parallel computers for the verification of large systems have appeared in the very early years of the automated formal verification era. However, the inaccessibility of cheap parallel computers with sufficiently fast external memory devices together with the negative theoretical complexity results excluded these approaches from the main stream in formal verification. Moreover, due to the Moore's law, the performance of software tools kept improving continuously for years as the power of a single-cored CPU grew. The situation changed dramatically with the introduction of multi-core CPU chips. The progress in computer design over the past decades had measured several orders of magnitude with respect to various physical parameters such as power consumption, efficiency, physical size or cost. As a result, it became more efficient for chip producers to introduce multiple CPU cores on a single chip rather than to increase the speed of a single core. As the speed of a single core virtually stopped growing, every piece of software that was built upon a serial algorithm could not take the advantage of technological progress anymore. The focus of parallel and distributed-memory computing community shifted away from unique massively parallel systems competing for world records towards smaller and more cost-effective systems built up from small and cheap personal computer parts. Suddenly, the need for parallel processing became rather general and widespread in all science fields relying on complex computation operations, automated formal verification being not an exception. As a matter of fact, the interest in the *platform-dependent* formal verification has been revived.

Unfortunately, some verification techniques cannot preserve their efficiency if adapted to non-sequential models of computation, and therefore an urgent need for new and quite different verification procedures emerged. Many new techniques have been introduced. There were attempts to consider both the symbolic as well as the enumerative techniques, theorem-provers as well as sat-solvers, etc. Some of those approaches are applicable across a broad range of computing platforms, some of them are tailored to the specific capabilities of a particular hardware architectures. Examples include techniques to fight the memory limits with an efficient utilization of external memory devices [156], techniques that introduce cluster-based algorithms to employ the aggregate power of network-interconnected computers [155, 129, 92], techniques to speed-up the verification process on multi-core processors [109, 14, 128], etc.

**Parallel Algorithms for LTL Model Checking** The need for parallel processing in automated formal verification stemmed from the desire to fight the state space explosion problem by employing the aggregate memory of multiple network interconnected workstations. The crucial problem is how to distribute the work among participating processors in order to take advantage of the aggregate memory and parallel processing at the same time.

Based on a parallel algorithm for state space generation [47], a static partitioning scheme relying on a hash function was introduced [52]. As observed by multiple researchers, the hash-based partitioning yields better space locality if only some parts of the state descriptor are used as the input to the partitioning function. There were considered approaches requiring the user of the tool to specify the concrete parts of the state descriptor to be used for partitioning [52], other approaches employed automated or semi-automated techniques to do it [53]. Techniques for load balancing the set of visited states, also known as re-partitioning techniques, have been suggested [1, 130, 124] as well as state space generation schemes employing probability aspects [122].

The first known public implementation of a distributed memory tool for the verification of communication protocols was the parallel implementation of the Mur$\varphi$ tool [155]. Mur$\varphi$'s parallel work-flow relied on the standard MPI-like approach to messaging, nevertheless, active messages were later introduced into Mur$\varphi$ to improve its efficiency. The successful story of Mur$\varphi$ was followed by other verification tools: SPIN [130], CADP [92], DiVinE [19], UPPAAL [28], etc. Distributed-memory techniques of automated formal verification also appeared in the context of Petri Nets [52, 104], Markov chains [101], and symbolic BDD-based model checkers [98, 83].

As a demonstration of distributed-memory approaches to verification we consider explicit state parallel LTL model checking. The LTL model checking problem can be reformulated as a cycle detection problem in an oriented graph and the basic principles behind presented algorithms rely on efficient solutions to detecting cycles in a distributed environment. The best known enumerative sequential algorithms for the detection of accepting cycles are the *Nested DFS* algorithm [63] (implemented, e.g. in the model checker SPIN [107]) and *SCC-*

*based algorithms* originating in Tarjan's algorithm for the decomposition of the graph into strongly connected components (SCCs) [158]. While Nested DFS is more space efficient, SCC-based algorithms produce shorter counterexamples in general. The linear time complexity of both algorithms relies on the postorder as produced by the depth-first search traversal. It is a well known fact that computing depth-first search postorder is P-complete [146], hence probably inherently sequential. This means that none of the two algorithms can be easily adapted to work on a parallel machine. A few fundamentally different cluster-based techniques for accepting cycle detection appeared, though. They typically perform repeated reachability over the graph. Unlike the postorder problem, reachability is a graph problem which can be parallelized, hence the algorithms might be transformed to cluster-based algorithms that work with reasonable increase in time and space.

The algorithms employ specific structural properties of the underlying graphs (often pre-computed in advance from the system specification), use additional data structures to divide the problem into independent sub-problems, or translate the model-checking problem to another one, which admits efficient parallel solution. Several of the algorithms are based on sequentially less efficient but well parallelizable breadth-first exploration of the graph or on placing bounds limiting the size of the graph to be explored.

The first parallel algorithm for LTL model checking employed the so-called dependency structure [17] to record the reachability relation among accepting states of a distributed graph and applied the topological sort algorithm [117] to detect the presence of a self-reachable accepting state. Other parallel algorithms appeared with the time, building upon various ideas. They have differed in theoretic complexity as well as practical efficiency, see [39] for a survey. The two most successful parallel algorithms for LTL model checking are the OWCTY algorithm [48] based on explicit-state implementation of symbolic SCC hull detection and the MAP algorithm [38] based on value propagation.

Distributed-memory processing cannot attack the state space explosion problem alone and must be combined with other techniques. One of the most successful techniques to fight the state space explosion in explicit-state model checking is *Partial Order Reduction* [140]. DiVinE is able to perform this reduction, even though a new topological sort proviso had to be developed in order to maintain efficiency of parallel and distributed-memory processing [16].

Another important algorithmic improvement relates to the classification of LTL formulas. For some classes of LTL formulas (weak LTL), the parallel algorithms may by significantly improved. With this observation the OWCTY algorithm can be improved so that its complexity even meets the complexity of the optimal sequential Nested DFS algorithm and it allows for on-the-fly verification in most verification instances [15].

**Parallelism in Distributed and Shared-Memory** The general idea in distributed-memory explicit state model checking is to aggregate the computational power of multiple network interconnected workstations (clusters) in order to fa-

cilitate the verification of large model checking instances [17]. The set of vertices of the graph to be processed is partitioned among participating computation nodes using a static partitioning function. When a computation node processes a vertex, it enumerates all its immediate successors and checks them for their ownership. If a newly generated vertex is local according to the partitioning function, it is pushed to the local queue where it waits for further processing. Otherwise a network message containing the vertex is created and sent to the queue of the owning computation node. With this work-flow, a message is generated with every edge connecting vertices from different partitions of the graph.

Message aggregation and buffering are the standard techniques in parallel computing to alleviate the burden of network communication overhead. Therefore, the model checker maintains buffers of messages to be sent to individual computing nodes. A buffer is flushed (messages sent to network) upon one of the following situations: 1) the buffer was explicitly flushed by the executed graph algorithm, 2) the maximal number of messages for the buffer has been reached, and 3) the local computing node was (otherwise) idle.

Most techniques and results known from the distributed-memory setting are straightforwardly applicable to shared-memory architectures. In particular, the graph to be processed is partitioned among individual parallel shared-memory threads in the same way as it would be in the distributed-memory setting. Each individual thread maintains its own hash table and its own pool of vertices to be processed. Vertices belonging to different threads are pushed to their local pools by means of lock-free shared-memory queues [14]. Relative advantages and disadvantages of shared versus private hash tables, within the context of thread-private pools of vertices to be processed, have been discussed in [22].

Nevertheless, the scalability of parallel distributed-memory solutions to shared-memory is often limited. Therefore, shared-memory specific techniques are needed to improve the efficiency and scalability of existing parallel distributed-memory solutions on shared-memory architectures. Examples of successful shared-memory specific techniques include, e.g. shared communication data structures [111, 14], specific termination detection techniques [14], dual-core algorithms [109], or quite a unique partitioning scheme [108].

**Many-Core Parallelism** After NVIDIA's CUDA technology [66] was introduced, a lot of computational demanding tasks have been accelerated by GPU-aware algorithms. Examples of GPU accelerated procedures include, but are not limited to, sorting [148], sparse matrix-vector multiplication [51], or numerous biological and physical simulations, such as protein folding [113]. As for the graph theory, successful adaptation of general graph traversal algorithms have been reported too [138] demonstrating the tremendous computational power of the CUDA device. On the other hand, graphs to be explored efficiently with a CUDA accelerated algorithm must be encoded explicitly in a compact way.

The CUDA technology as a computing platform, attracted also researches in the field of automated formal verification. The key challenge, for which no satisfactory solution is known yet, is how to accelerate the generation of explicitly

encoded state space graph from implicit definition. Preliminary attempts to do so relate to explicit model checking. They suggest to employ a massively parallel check for enabled transitions emanating from the vertices on the frontier of the search and their massively parallel execution [78].

Once the state space is generated and explicitly represented in an appropriate sparse matrix-like structure, many verification tasks can be accelerated using CUDA technology. This has been successfully demonstrated, e.g. on verification of probabilistic systems [35], LTL model checking [23] or the acceleration of strongly connected components decomposition [12].

### 3.5 Model Checking Tools for Biological Systems

There are several specialized tools for the analysis of biological systems that employ model checking. Some of these tools are well accepted by the community and routinely used in the process of model development. In addition, several model checking tools were experimentally used for the analysis of models in systems biology. In this section we point to three of them that we found to be closest to our own interest in exhaustive model checking. For richer reviews we would like to refer to [46, 10, 112].

**BioCham** (Fages et al. [85], see [84] for tutorial)

BioCham stands for BIOCHemical Abstract Machine. The tool provides a modeling environment for systems biology, with some unique features for static analysis or for inferring unknown model parameters from temporal logic constraints. BioCham covers qualitative (Boolean) models as well as quantitative models (continuous-time deterministic and continuous-time stochastic). Models are specified in its native rule-based language. An important feature is that quantitative models specified at the level of reaction networks can be automatically analyzed at the level of qualitative (Boolean) semantics.

*Qualitative Models* CTL is employed to formalize the temporal properties of a biological system and validate models with respect to such specifications. Symbolic model checker NuSMV [54] is used to handle this analysis task. Moreover, BioCham has an update component for automatically modifying a network that does not satisfy a given CTL formula. The algorithm of this component is based on the counterexamples computed by NuSMV. Although incomplete (in the sense of sometimes not being able to find the appropriate changes to networks), such a component is useful because of being able to handle large networks [43].

*Continuous-time Deterministic Models* BioCham introduces LTL with numerical constraints (LTL($\mathbb{R}$)) to specify properties of numerical simulations of ODE models. Since simulations always produce finite discretely-sampled trajectories bounded by the requested time horizon, there is a natural monitoring algorithm built in. Furthermore, the tool is able to compute the violation degree of a formula. Intuitively, a violation degree is the distance between a particular behavior

of a system, given as a path, and the expected behavior, given as a temporal-logic formula [147]. Such a violation measure can be used to estimate a fitness function with evolutionary optimization methods. This is done by finding kinetic parameter values satisfying a set of biological properties formalized in temporal logic. In addition, such a measure can be used to estimate the robustness of a biological model with respect to its temporal specification.

Finally, probabilistic model checking is also provided. BioCham estimates the probability of an LTL formula satisfaction by sampling stochastic simulations.

**GNA** (de Jong et al. [116])

Genetic Network Analyzer (GNA) provides support for modeling and simulation of genetic regulatory networks using knowledge about regulatory interactions in combination with gene expression data. GNA operates on piece-wise affine models providing a clear relation between quantitative and qualitative semantics. Instead of exact numerical values for the parameters, which are often not available for gene networks, the piece-wise affine models allow to specify inequality constraints. This information is sufficient to generate a state transition graph that describes the qualitative dynamics of the network overapproximating the ODE model.

GNA is able to export the resulting qualitative model to the finite state transition system and check properties by means of standard model-checking tools, either locally installed or accessible through a remote web server. The tool is connected with NuSMV and CADP (Garavel et al. [91]) model checkers. GNA supports an extension of CTL logic, CTRL [136], allowing to express a significant set of biologically relevant properties not expressible in plain CTL.

Additionally, parameter identification techniques have also been introduced for GNA [26]. Based on symbolic model checking, the method avoids enumerating all possible parametrizations in searching for parametrizations satisfying the given temporal specification.

**BioDiVinE** (Barnat et al.[18, 20])

BioDiVinE[1] is a tool-box for automated analysis of biological systems by means of applying model checking to qualitative and quantitative biological models. Emphasis is put on the computational aspects and algorithms are adapted to enable their effective distribution and/or parallelization. Currently, the tool-box contains the following tools:

BioDiVinE 1.0 is a tool created for model checking LTL properties over continuous-time deterministic biological models given by means of multi-affine ODEs ([18], see Section 3.1), which are abstracted by employing the rectangular abstraction [62], translating the continuous model into a finite automaton. The tool makes an evolutionary branch of enumerative LTL model checker DiVinE [19] by adapting the OWCTY [48] and MAP [38] algorithms for distributed

[1] `http://sybila.fi.muni.cz/tools`

analysis of biological models. Properties are specified by means of Büchi automata allowing even more flexibility than is provided by LTL.

Parsybone and PEPMC are tools for property-driven identification of biological models. Parsybone focuses on logical parameters in qualitative models [121], in particular, in gene regulatory networks encoded using the formalism of R. Thomas [159]. PEPMC provides parameter identification for quantitative models [13], in particular, continuous-time deterministic models represented as piece-wise multi-affine ODEs (this model class generalizes multi-affine models to capture regulatory dynamics). Both tools are based on colored LTL model checking technique providing a heuristics for effective exploration of models with finite parametrizations. As in BioDiVinE 1.0, specification of the temporal properties is realized by means of Büchi automata.

PARASIM is a tool for approximative analysis of robustness of continuous-time deterministic models. For sets of perturbations of kinetic parameters (or initial conditions) and temporal properties specified in Signal Temporal Logic (STL) [134], the so-called landscape function, giving the property's validity for parametrizations in the required perturbation set, is computed.

Usage of the BioDiVinE toolset is demonstrated in Section 4.

## 4 Model Checking in Action – Application Examples

In this section we give case studies on qualitative and quantitative representations of two biologically relevant models. The main purpose is to demonstrate the application of selected techniques based on model checking. We focus on techniques implemented in the BioDiVinE tool set, in particular, explicit LTL model checking and parameter identification techniques built on the top of it. Additionally, we provide a demonstration of probabilistic model checking recently extended to property-driven exploration of model parameters.

### 4.1 *E. coli* Ammonium Transport Model

We consider a simple biological model that describes ammonium transport from the external environment into the cells of *Escherichia Coli*. This simplified model is based on a published model of the *E. Coli* ammonium assimilation system [131].

The model is a typical example of the dynamical models appearing in current computational systems biology. In particular, the model is represented as a reaction network associated with a continuous-time deterministic semantics given in terms of (non-linear) ODEs. Parameters were taken from the literature.

We employ model checking to explore the model dynamics from a global perspective. The term *global* has two meanings here. First, we want to analyze the model dynamics starting at any possible initial concentration of the species, not only at a single initial condition, as allowed by traditionally used simulation methods. Second, we want to explore the model dynamics without restricting

ourselves to a given parametrization. In particular, we want to explore how parameters affect the expected (or required) behavior of the model.

As stated in Section 3.1, exhaustive exploration of the system states cannot be directly employed on continuous-time deterministic models due to the uncountability of time and variable domains. In order to allow the model checking analysis, the model must be simplified in terms of Section 3.1. In this particular case, we employ the rectangular abstraction technique [62] that allows to transform an ODE model of a specific class into a finite automaton provided that the dynamics properties are (conservatively) preserved.

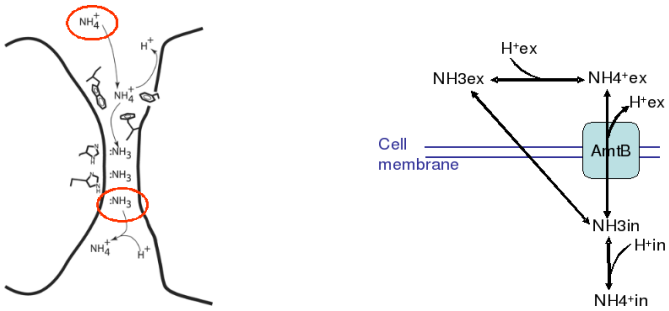



**Fig. 3.** *E. Coli* ammonium transport mechanism and the respective pathway

**Model Description** *E. coli* can express membrane-bound transport proteins for the transportation of small molecules from the environment into the cytoplasm at certain conditions. At normal ammonium concentration, the free diffusion of ammonium can provide enough flux for the growth requirement of nitrogen. When ammonium concentration is very low, E. coli cells express $AmtB$ (an *ammonium transporter*) to complement the deficient diffusion process. Three molecules of $AmtB$ (trimer) form a channel for the transportation of ammonium. Protein structure analysis revealed that $AmtB$ binds $NH4^+$ at the entrance gate of the channel, deprotonates it and conducts $NH_3$ into the cytoplasm as illustrated in Figure 3 (left) [119]. At the periplasmic side of the channel there is a wider vestibule site capable of recruiting $NH_4^+$ cations. The recruited cations are passed through the hydrophobic channel where the pKa of $NH_4^+$ was shifted from 9.25 to below 6, thereby shifting the equilibrium toward the production of $NH_3$. $NH_3$ is finally released at the cytoplasmic gate and converted to $NH_4^+$ because the intracellular pH (7.5) is far below the pKa of $NH_4^+$.

In addition to the above mentioned $AmtB$ mediated transport, the bidirectional free diffusion of the uncharged ammonium through the membrane is also included in the simplified model. The intracellular $NH_4^+$ is then metabolised by Glutamine Synthetase (GS). The whole model is depicted in Figure 3 (right). The external ammonium is represented in the uncharged and charged forms denoted $NH_3ex$ and $NH_4^+ex$. Analogously, the internal ammonium forms are denoted

$NH_3in$ and $NH_4^+in$. The reaction network that combines $AmtB$ transport with $NH_3$ diffusion is given in Table 1.

$$
\begin{array}{cc}
AmtB + NH_4ex \overset{k_1}{\underset{k_2}{\leftrightarrows}} AmtB:NH_4 & k_1 = 5 \cdot 10^8, k_2 = 5 \cdot 10^3 \\
AmtB:NH_4 \overset{k_3}{\rightarrow} AmtB:NH_3 + H_{ex} & k_3 = 50 \\
AmtB:NH_3 \overset{k_4}{\rightarrow} AmtB + NH_3in & k_4 = 50 \\
NH_4in \overset{k_5}{\rightarrow} & k_5 = 80 \\
NH_3in + H_{in} \overset{k_6}{\underset{k_7}{\leftrightarrows}} NH_4in & k_6 = 1 \cdot 10^{15}, k_7 = 5.62 \cdot 10^5 \\
NH_3ex \overset{k_8}{\underset{k_9}{\leftrightarrows}} NH_3in & k_8 = k_9 = 1.4 \cdot 10^4
\end{array}
$$

**Table 1.** The model of ammonium transport

The reaction network is assigned a set of ODEs as listed in Table 2 (employing the law of mass action kinetics). It is worth observing that the form of the ODE right-hand sides is in all cases made by polynomials of degree one. Since we are especially interested in how the concentrations of internal ammonium change with respect to the external ammonium concentrations, we employ the following simplifications:

- We do not consider the dynamics of the external ammonium forms, thus we take $NH_3ex$ and $NH_4^+ex$ as constants (the input parameters for the analysis).
- We assume constant intracellular pH (7.5) and extracellular pH (7.0), thus $H_{ex}$ and $H_{in}$ are calculated to be $3{\cdot}10^{-8}$ and $10^{-7}$. Based on the extracellular pH and the total ammonium concentration, concentrations of $NH_3ex$ and $NH_4^+ex$ can be calculated.

Without loss of correctness, we simplify the notation of the cation $NH_4^+$ as $NH_4$.

$$
\begin{array}{c}
\frac{d[AmtB]}{dt} = -k_1 \cdot [AmtB] \cdot [NH_4ex] + k_2 \cdot [AmtB:NH_4] + k_4 \cdot [AmtB:NH_3] \\
\frac{d[AmtB:NH_3]}{dt} = k_3 \cdot [AmtB:NH_4] - k_4 \cdot [AmtB:NH_3] \\
\frac{d[AmtB:NH_4]}{dt} = k_1 \cdot [AmtB] \cdot [NH_4ex] - k_2 \cdot [AmtB:NH_4] - k_3 \cdot [AmtB:NH_4] \\
\frac{d[NH_3in]}{dt} = k_4 \cdot [AmtB:NH_3] - k_7 \cdot [NH_3in] + k_6 \cdot [NH_4in] \\
\frac{d[NH_4in]}{dt} = k_5 \cdot [NH_4in] + k_7 \cdot [NH_3in] \cdot [H_{in}] - k_6 \cdot [NH_4in]
\end{array}
$$

**Table 2.** The mathematical model of ammonium transport

**Model Simplification** The restricted polynomial form of ODEs implies that the model falls into the class of so-called multi-affine systems for which a simplification, the so-called rectangular abstraction, is defined [30] (see [62] for the relation with model checking). Each variable is assigned a set of specific (arbitrarily defined) points, the so-called *thresholds*, expressing concentration levels of

special interest. This set contains two specific thresholds – the maximal and the zero concentration level (bounding of the state space has been discussed in Section 3.1). The intermediate thresholds then define a partition of the (bounded) continuous state space. The individual regions of the partition are called *rectangles*. An example of a partition is given in Figure 4.

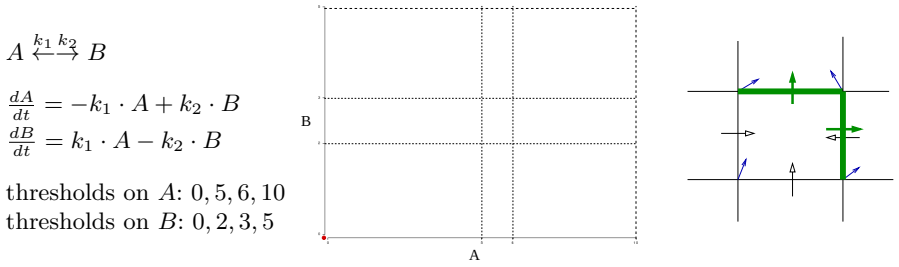



**Fig. 4.** Example of a rectangular partition of a two-dimensional system (left) and the intuition behind the construction of the abstracted transition system (right).

The partition of the system gives us directly the finite discrete abstraction of the dynamic system. In particular, the BioDiVinE tool implements a (discrete) state space generator that constructs a finite automaton representing the rectangular abstraction of the system dynamics. Since the states of the automaton are made by the rectangles in the phase-space partition, the automaton is called *rectangular abstraction transition system* (RATS). The main point is that for each rectangle the exit faces are determined. The intuition is depicted in Figure 4(right). There is a transition from a rectangle to its neighbouring rectangle only if, in the vector field considered in the shared face, there is at least one vector whose particular component agrees with the direction of the transition. The important result is that in a multi-affine system it suffices to consider only the vector field in the vertices of the face. In Figure 4(right), the exit faces of the central rectangle are emphasised by bold lines. In Figure 5 there is depicted the rectangular abstraction transition system constructed for the affine system from Figure 4(left). It is known that the rectangular abstraction is an overapproximation with respect to trajectories of the original dynamic system.

There is one specific issue when considering the time progress of the abstracted trajectories. If there exists a point in a rectangle from which there is no trajectory diverging out through some exit face, then there is a self-transition defined for the rectangle. In particular, this situation signifies an equilibrium inside the rectangle. Such a rectangle is called non-transient. For affine systems, a sufficient and necessary condition is known, that characterizes non-transient rectangles by the vector field in the vertices of those rectangles. However, for multi-affine systems, only the necessary condition is known. Hence, for multi-affine systems BioDiVinE treats as non-transient some states which are not necessarily non-transient.

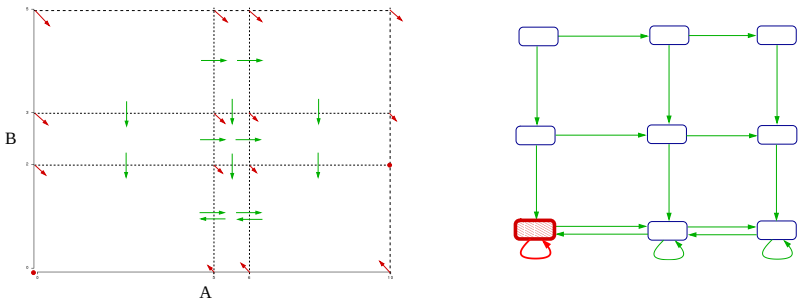



**Fig. 5.** Example of a rectangular abstraction transition system. The emphasized state and transition make a counterexample contradicting the property $\mathbf{F}(B > 3)$.

We use LTL logic to encode the dynamics properties of the model. Given a dynamic system $S$ with a particular initial state we can then say that $S$ satisfies a formula $\varphi$, written $S \models \varphi$, only if the trajectory starting at the initial state satisfies $\varphi$. In the context of automata, LTL logic is interpreted universally provided that a formula $\varphi$ is satisfied by the automaton $A$, written $A \models \varphi$, only if each execution of the automaton starting from any initial state satisfies $\varphi$. The following theorem characterizes the relation between validity of $\varphi$ in the rectangular abstraction automaton and in the original dynamic system, taken from [62].

**Theorem 1.** *Consider a dynamic system $S$ and the associated RATS $A$. If $A \models \varphi$ then $S \models \varphi$.*

The theorem states that when the model checking of a particular property on a RATS returns true, we are sure that the property is satisfied in the original dynamic system. However, when the result is negative, the counterexample returned does not necessarily reflect any trajectory in the original system.

The system in Figure 5 satisfies a formula $\mathbf{FG}(B \leq 3)$ expressing the temporal property stating that whatever the choice of the initial state, the system eventually stabilizes at states where concentration of $B$ is kept below 3. Now let us consider a formula $\mathbf{F}(B > 3)$ expressing the property that whatever the initial settings, the concentration of $B$ will eventually exceed the concentration level 3. In this case the model checking returns one of the counterexamples as emphasized in Figure 5(right) stating that if initially $A < 5$ and $B < 3$ then $B$ is not increased while staying indefinitely long in the emphasised state.

We apply the rectangular abstraction method to the ammonium transport model. We consider the set of states from which we want to explore the dynamics given by the following intervals of concentration values:

$$AmtB \in \langle 0, 1{\cdot}10^{-5}\rangle, \quad AmtB : NH_3 \in \langle 0, 1{\cdot}10^{-5}\rangle, \quad AmtB : NH_4 \in \langle 0, 1{\cdot}10^{-5}\rangle,$$
$$NH_3in \in \langle 1 \cdot 1^{-6}, 1.1 \cdot 10^{-6}\rangle, \quad NH_4in \in \langle 2 \cdot 10^{-6}, 2.1 \cdot 10^{-6}\rangle$$

The upper bounds as well as the intervals of internal ammonium forms have been set with respect to the available data obtained from the literature. The

partition used for rectangular abstraction has been set by thresholds as given in Table 3.

| | |
|---|---|
| $AmtB$ | 0, $10^{-12}$, $10^{-10}$, $9.9 \cdot 10^{-8}$, $10^{-7}$, $5 \cdot 10{-}6$, $10^{-5}$ |
| $AmtB : NH_3$ | 0, $10^{-7}$, $10^{-5}$ |
| $AmtB : NH_4$ | 0, $10^{-7}$, $10^{-5}$ |
| $NH_3in$ | 0, $10^{-6}$, $1.1 \cdot 10^{-6}$, $3 \cdot 10^{-6}$, $8 \cdot 10^{-6}$, $10^{-5}$ |
| $NH_4in$ | 0, $2 \cdot 10^{-6}$, $2.1 \cdot 10^{-6}$, $10^{-5}$, $5 \cdot 10^{-4}$, $5.3 \cdot 10^{-4}$, $5.4 \cdot 10^{-4}$, $10^{-3}$ |

**Table 3.** Partitioning of the continuous state space.

**Model Checking Analysis** From the essence of biophysical laws, it is clear that the maximal reachable concentration level accumulated in the internal ammonium forms directly depends on the ammonium sources available in the environment. However, it is not directly clear what particular maximal level of internal ammonium is achievable at given amount of external ammonium (distributed into the two forms). In the analysis we have focused on just this phenomenon. More precisely, the problem to solve was to analyze how the setting of the model parameters $NH_3ex$ and $NH_4^+ex$ affects the maximal concentration level of $NH_3in$ and $NH_4^+in$ reachable from given initial conditions.

It is very difficult to provide *in vitro* measurements of $AmtB$ concentration (and also the concentration of dimers $AmtB : NH_3$ and $AmtB : NH_4$). This gives a strong motivation to analyze the model globally (with uncertain initial conditions).

We have conducted several model checking experiments in order to determine the maximal reachable concentration levels of $NH_3in$ and $NH_4^+in$. In particular, we have searched for the lowest $\alpha$ satisfying the property $\mathbf{G}(NH_3in < \alpha)$ and the lowest $\beta$ satisfying $\mathbf{G}(NH_4in < \beta)$. The property $\mathbf{G}\,p$ requires that all paths available in the rectangular abstraction from the states specified by the initial condition must satisfy the given proposition $p$ at every state. Note that if the model checking method finds the property $\mathbf{G}\,p$ false in the model, it also returns a counterexample for that. The counterexample satisfies the negation of the checked formula, which is in this case $\mathbf{F}\neg p$. Interpreting this observation intuitively for the above formulae, we use model checking to find a path on which the species $NH_3in$ (resp. $NH_4in$) exceeds the level $\alpha$ (resp. $\beta$).

The procedure was the following: At the starting point, we substituted for $\alpha$ (resp. $\beta$) the upper initial bounds of the respective variables. Then we found the requested values by iteratively increasing and decreasing $\alpha$ (resp. $\beta$). The obtained results are summarized in Table 4.

The results have shown that $NH_3in$ does not exceed its initial level no matter how the external ammonium is distributed between $NH_3ex$ and $NH_4^+ex$. The upper bound concentration considered for both $NH_3ex$ and $NH4^+ex$ has been

| $\alpha$ | $\mathbf{G}(NH_3in < \alpha)$ | # states | Time |
|---|---|---|---|
| $1.1 \cdot 10^{-6}$ | true | 1081 | 0.36 s |
| $\beta$ | $\mathbf{G}(NH_4in < \beta)$ | # states | Time |
| $1 \cdot 10^{-3}$ | true | 2161 | 0.45 s |
| $5 \cdot 10^{-4}$ | false | 4753 | 1.9 s |
| $6 \cdot 10^{-4}$ | true | 2161 | 0.43 s |
| $5.4 \cdot 10^{-4}$ | true | 1441 | 0.27 |
| $5.3 \cdot 10^{-4}$ | false | 3421 | 1.2 s |

**Table 4.** Experiments on detecting maximal reachable levels of internal ammonium

set to $1 \cdot 10^{-5}$ which corresponds to common concentration level of the gas in the cell environment.

In the case of $NH_4in$ we have found that the upper bound to maximal reachable level is in the interval $\beta \in \langle 5.3 \cdot 10^{-4}, 5.4 \cdot 10^{-4} \rangle$. Since the counterexample achieved can be a spurious one due to the overapproximating abstraction, the exact maximal reachable value may be lower. This can be explored by numerical simulation, an important fact is that the range for setting $\beta$ is now limited to the detected interval.

The results have been achieved by running the OWCTY algorithm on a single computation node. In [18], there is presented a refined variant (a finer partition) of the model leading to $10^5$ reachable states. For that variant, distributing of the computation to 36 nodes was needed to achieve good times (in the order of seconds).

**Parameter Exploration** Owing to the membrane location of *AmtB*, *in vitro* measuring of the concentration of *AmtB*-based species is impossible and therefore the estimation of kinetic parameters of this model is very difficult.

To identify parameter values computationally, we employ the colored model checking technique [13] implemented in the PEPMC tool of the BioDiVinE toolset (see Section 3.5). If we denote each parametrization by a distinct color and assume that the respective (parametrization-specific) state space has all its transitions marked by this color, we can construct a global state space as a union of all the parametrization-specific state spaces. Since a change in parameter values affects the model dynamics, which is entirely represented by state transitions, the parameter space is completely projected onto the transition relation defined on the universal state space. In this setting, our solution to the parameter identification problem is based on analysis of mono-colored paths in a graph with multi-colored edges. Since in many parametrizations small perturbations in parameter values lead to small locally distributed variations in the transition relation, the respective mono-colored graphs can exhibit significant similarity. For parametrizations amenable to such property, the algorithm achieves good efficiency. In Fig. 6, the basic idea of solving the parameter identification problem by automata-based LTL model checking is illustrated. The automaton represent-

ing the model dynamics can be extended with colored edges, where each color corresponds to a certain parametrization. We expect that transitions are to a large extent shared among individual colors. This allows us to accelerate the computation.

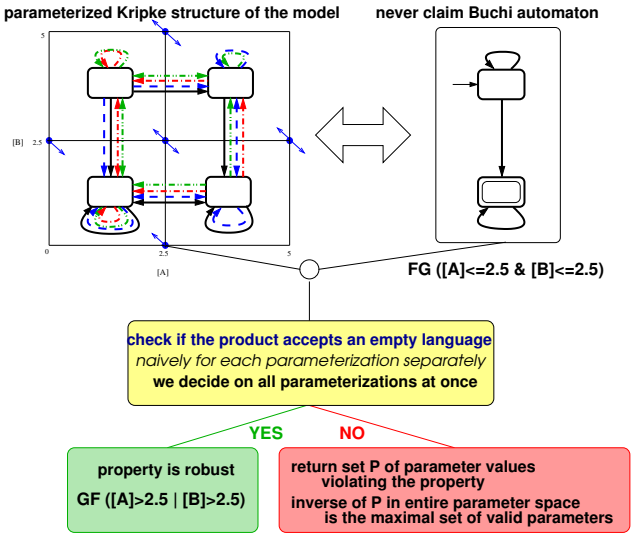


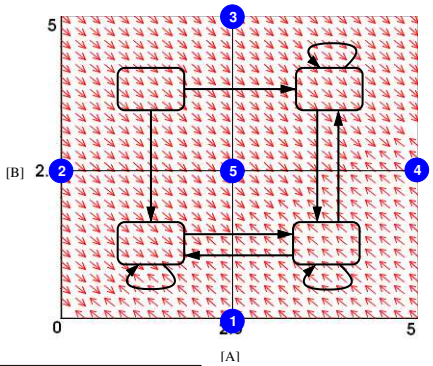

**Fig. 6.** Intuition behind colored model checking.

It is important to note that in the considered model the parameters are quantitative and their domain is uncountable (but bounded). However, as shown in [25], the rectangular abstraction partitions the parameter domains into a finite number of intervals where each interval contains parameter values producing an isomorphic state transition system. Intuition behind the application of this result is illustrated in Fig. 7.

In our model, we investigate the effect of different parameter settings to the production of the model output – the internal ammonium forms $NH_3in$ and $NH_4in$. In particular, we look for perturbations in individual kinetic parameters that lead to an increase of internal ammonium concentration above the standard values. In the terms of LTL model checking, we formulate the negation of this requirement – we check whether the standard value is never exceeded. We formalize the discussed requirement by safety LTL properties $\varphi_1 = \mathbf{G}(NH_3in < 1.1 \cdot 10^6)$ and $\varphi_2 = \mathbf{G}(NH_4in < 2.1 \cdot 10^6)$ stating that $NH_3in$ (resp. $NH_4in$) never exceeds the given concentration. We performed two sets of parameter identification tasks. In the first group, each single parameter was considered unknown ( remaining parameters were set w.r.t. literature [131]). In the second group, we considered a collection of three unknown parameters. In all experiments, the range for every parameter was set to $(1 \cdot 10^{-12}, 1 \cdot 10^{12})$. In Table 5, the most interesting results

$$\frac{dA}{dt} = -k_1 \cdot A + k_2 \cdot B$$
$$\frac{dB}{dt} = k_1 \cdot A - k_2 \cdot B$$

$k_2 = 0.8$
$k_1 = ?$




| | value of k1: (0,0.4) | (0.4,0.8) | (0.8,1.6) | (1.6,max) |
|---|---|---|---|---|
| 1 | ↖ | ↖ | ↖ | ↖ |
| 2 | ↘ | ↘ | ↘ | ↘ |
| 3 | ↘ | ↘ | ↘ | ↖ |
| 4 | ↘ | ↖ | ↖ | ↖ |
| 5 | ↘ | ↘ | ↖ | ↖ |

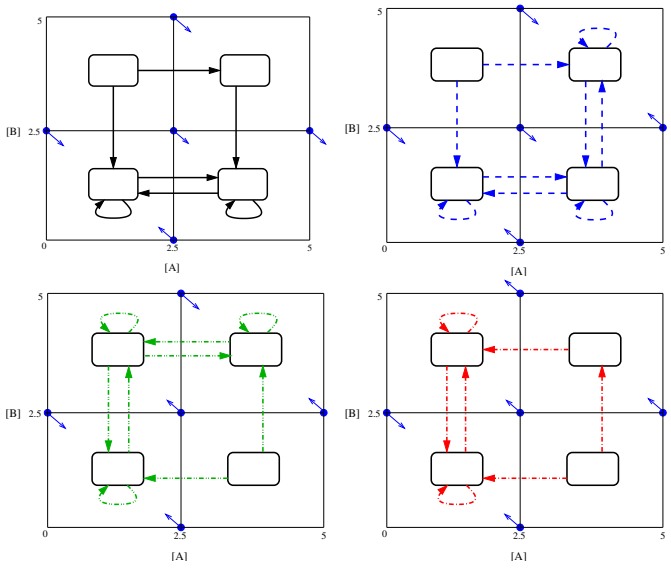



**Fig. 7.** Partitioning of uncountable parameter space into a finite number of intervals. Parameter $k_1$ is considered to be unknown. Rectangular abstraction of the model is determined by thresholds $0, 2.5, 5$ imposed on both species $A$ and $B$ producing five intersection points. By substituting any of these points into the ODEs while setting the left-hand sides to zero (equilibrium), we can solve the resulting homogeneous system of linear equations for $k_1$. The solution gives us those values of $k_1$ where the sign of any of the derivatives changes. By iterating this procedure for each of the points, the domain of $k_1$ is partitioned into four intervals as can be seen in the table above. Each of the intervals makes a class of equivalence with respect to the derivative sign in a particular intersection point. Accordingly, in this example we get four (qualitatively) different automata abstracting the model dynamics.

are summarized. The presented data show the scanned parameter set, the analyzed property with the computed valid parametrizations, number of reached states, and computation times. Note that if a parameter is not mentioned it led trivially to validity on the entire $(1 \cdot 10^{-12}, 1 \cdot 10^{12})$.

| $P$ | prop. | intervals of validity | # states reached | time |
|---|---|---|---|---|
| $k_4$ | $\varphi_1$ | $(1 \cdot 10^{-12}, 2.7 \cdot 10^6)$ | 124580 | 30 s |
| $k_5$ | $\varphi_2$ | $(1.5 \cdot 10^7, 1 \cdot 10^{12})$ | 3068 | 0.40 s |
| $k_6$ | $\varphi_1$ | $(5.2 \cdot 10^6, 1 \cdot 10^{12})$ | 67572 | 22 s |
| $k_6$ | $\varphi_2$ | $\emptyset$ | 6319 | 1.8 s |
| $k_7$ | $\varphi_1$ | $(1 \cdot 10^{-12}, 3.3 \cdot 10^6)$ | 126458 | 33 s |
| $k_7$ | $\varphi_2$ | $(1.6 \cdot 10^7, 1 \cdot 10^{12})$ | 12523 | 3.5 s |
| $k_9$ | $\varphi_1$ | $(1 \cdot 10^{-12}, 2.7 \cdot 10^6)$ | 97495 | 20 s |
| $k_9$ | $\varphi_2$ | $\emptyset$ | 5779 | 1.5 s |
| $k_{1,6,9}$ | $\varphi_1$ | $k_9 \in (1 \cdot 10^{-12}, 2.7 \cdot 10^6) \vee$ $[k_9 \in (2.7 \cdot 10^6, 3.2 \cdot 10^6) \wedge k_6 \in (1 \cdot 10^{-12}, 1.07 \cdot 10^6)]$ | 202638 | 51 min |
| $k_{1,6,10}$ | $\varphi_2$ | $[k_1 \in (1 \cdot 10^{-12}, 1 \cdot 10^7) \wedge k_6 \in (1 \cdot 10^{-12}, 1.4 \cdot 10^5) \wedge k_{10} \in (1.18 \cdot 10^6, 1 \cdot 10^{12})]$ $\vee [k_6 \in (1.4 \cdot 10^5, 1.07 \cdot 10^6) \wedge k_{10} \in (1 \cdot 10^{-12}, 1.18 \cdot 10^5)]$ | 19473 | 19 min |

**Table 5.** Parameter exploration experiments.

Of special interest are individual scans of $k_6$ and $k_9$ for $\varphi_2$. In particular, the results show that, in the given parameter value range, there is no perturbation which would satisfy the property. With respect to both parameters, the model is robust in the negative property $\mathbf{F}(NH_4in > 2.1 \cdot 10^6)$ stating that $NH_4in$ eventually exceeds the given concentration. Thus, regardless the setting of $k_6, k_9$, on each trajectory leading from the range specified by initial conditions, $NH_4in$ must exceed the standard concentration range.

### 4.2 Gene Regulation of Mammalian Cell Cycle

In the second case study, we focus on parameter identification by model checking for a model of a regulatory network. In particular, we investigate a model representing the central module of the genetic regulatory network governing the $G_1/S$ cell cycle transition in mammalian cells [157]. In particular, the model considers a two-gene network describing interaction of the tumor suppressor protein $pRB$ and the central transcription factor $E2F1$ (see Fig. 8(left)).

This simple model demonstrates the feature of bistability, i.e., the occurrence of two stable states. Bistable networks can drive the systems response to some stimulus: With no stimulus, the system keeps (once reaching it) a certain stable state. In particular, in the stable state, the concentrations of the species does not change, because production and degradation had reached an equilibrium. A stimulus, which is in the form of a change of some protein concentration caused from outside the system, evokes deflection from the stable state. If the deflection is weak enough, the system returns to the previous stable state afterwords.

But when it overruns some threshold, the system approaches the other stable state, with no chance of returning to the first one, even after the end of the stimulus. That way the system can decide whether a stimulus is strong enough to permanently switch to a particular mode. The first stable state of our system represents the low level of the $E2F1$ protein concentration. In this state, the cell stays in $G_1$-phase. When increased enough, the concentration of $E2F1$ grows higher, to the level of the second stable state, which causes the cell approaches to $S$-phase.

**Qualitative model** First, we consider a Boolean model of the network, we formulate the required properties, and we employ the parameter identification algorithm to find parametrizations of unknown parameters (denoted by question marks in Fig. 8).

We employ the Boolean model of gene regulatory networks as introduced by Thomas [159]. The concrete modeling approach including the parametrization is taken from [13]. The *Boolean model* is determined by the structure (topology) of the GRN and the regulatory logic that controls the network dynamics. The Boolean model is defined as a tuple $\mathcal{B} = \langle G, \sigma, \theta, \rho, L \rangle$ where

- $G = (V, E)$ is a directed graph with vertices $V = \{g_1, ..., g_n\}$ denoting *genes* and set of edges $E \subseteq V \times V$ denoting *regulations*.
- $\sigma(e) \in \{+, -\}$ denotes the type of regulation $e \in E$: positive $(+)$ or negative $(-)$,
- $\theta(e) \in \mathbb{N}_{\geq 1}$ denotes the activation *threshold* of $e \in E$,
- $\rho(g_i) \in \mathbb{N}_{\geq 1}$ denotes the *maximum expression level* of $g_i \in V$ determining the expression domain $\{0, ..., \rho(g_i)\}$,
- $L$ is the *regulatory logic* defined as the set $L = \{K_{i,R} \mid 1 \leq i \leq n, R \subseteq \{v \in V \mid \langle v, g_i \rangle \in E\}\}$ where $K_{i,R}$ denotes the *target expression level* of $g_i$ when regulated by all genes in $R$, $0 \leq K_{i,R} \leq \rho(g_i)$.

In our example, the model, depicted in Fig. 8, consists of two genes $pRB$ and $E2F1$. To differentiate between the two outgoing regulations from both E2F1 and pRB, we choose the genes maximal activity levels $\rho(\text{pRB}) = \rho(\text{E2F1}) = 2$. Negative and positive interactions together with thresholds are set with respect to data presented in [157]. The regulatory logic is known only for the basal gene activity, in particular, $pRB$ under the empty context (no incoming regulation active) has the tendency to attain the expression level 1. A significant role for the model behavior has the positive autoregulation of E2F1. In order to become active (i.e., the resource for E2F1, since E2F1 is not a target of any other positive regulation), we need to set $K_{\text{E2F1},\emptyset} = 2$. We do not know target levels for other regulatory contexts of both genes, therefore we consider them as parameters. Note that since the expression levels of both genes is bounded by the maximal activity levels, the number of possible parametrizations is finite.

Once the regulatory logic is set (all parameters are assigned), the semantics in terms of a finite automaton capturing the dynamics of a network $\mathcal{B}$ can be defined as the tuple $BTS(\mathcal{B}) = \langle S, T, S_0 \rangle$ where $S = \prod_{i=1}^{n} \{0, ..., \rho(g_i)\}$ is set of

states with $S_0 \subseteq S$ initial states and $T \subseteq S \times S$ is the transition relation defined as follows.

First we denote the level of $g_i$ in the state $s \in S$ by $l_i(s)$. Assume there is a regulation $e = \langle g_i, g_j \rangle \in E$ between genes $g_i, g_j$. We say that $g_i$ is a *resource* for $g_j$ in $s$ if $\sigma(e) = +$ and $l_i \geq \theta(e)$, or $\sigma(e) = -$ and $l_i < \theta(e)$. Let $Re(s, g_i)$ denote the set of all resources of $g_i$ in $s$. There is a transition $s \to s'$ according to the following rules:

- If there exists $u$ such that $K_{u,Re(s,g_u)} > l_u$ then $l_u(s') = l_u(s) + 1$.
- If there exists $u$ such that $K_{u,Re(s,g_u)} < l_u$ then $l_u(s') = l_u(s) - 1$.

The presented semantics requires that the level of at most one gene can be affected in a single transition. This represents the so-called *asynchronous semantics* [159] that models all possible time-orderings of individual expression level updates by the means of non-determinism (an update on a single gene is considered an atomic operation).

The formulae specifying behavior of E2F1 are built over the following atomic propositions:

$$AP = \{\text{E2F1} < x \mid 1 \leq x \leq \rho(\text{E2F1})\}$$

For the purpose of our analysis, we establish the set of initial states $S_0$ as those satisfying $l_{\text{pRB}} = 0$ and $l_{\text{E2F1}} \in \{0, 1, 2\}$.

To filter out certain trivial executions, the concept of explicitly stated accepting states in the Büchi automaton (BA) representing the LTL property is used. Such a restriction is called a *fairness constraint* as is frequently used in formal verification by model checking [59]. From the above regulatory logic settings follows the selection of accepting states with $l_{\text{pRB}} \geq 1$, denoted $F_1$. A more restricting filter may be created by choosing states satisfying $l_{\text{pRB}} = 2$, denoted $F_2$. The former fairness constraint can be formulated in LTL as a formula $\mathbf{GF}(\text{pRB} \geq 1)$, the latter one as $\mathbf{GF}(\text{pRB} = 2)$.

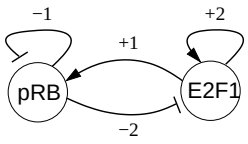
We understand bistability as the following set of properties (described in the form of LTL formulae). These properties are used to detect certain paths in the model state space with respect to the behavior observed *in vitro*. In the following we assume $\theta \in \{1, 2\}$:

- expression of E2F1 begins below the threshold and does not exceed it ($\mathbf{G}(\text{E2F1} < \theta)$)
- expression of E2F1 begins above the threshold and remains such ($\mathbf{G}(\text{E2F1} \geq \theta)$)
- expression of E2F1 begins under the threshold and at certain moment exceeds it and stays above the exceeded level ($(\text{E2F1} < \theta) \to \mathbf{FG}(\text{E2F1} \geq \theta)$)

We used the colored model checking algorithm implemented in the Parsybone tool of the BioDiVinE toolset (see Section 3.5) to identify admissible parametrizations for these properties with the setting of accepting states $F_1$. In particular, the properties listed above have been used as an observer (BA) that makes a witness for the respective dynamic phenomenon. Only a small parameter restriction was synthesized by employing the observer (BA) for both

settings $\theta = 1$ and $\theta = 2$: $K_{\mathrm{pRB},\{\mathrm{E2F1}\}} \neq 0$. When employing the accepting states $F_2$, the observer demanded $K_{\mathrm{pRB},\{\mathrm{pRB,E2F1}\}} = 2$ as well and, additionally, $K_{\mathrm{pRB},\{\mathrm{pRB}\}} = 2$ for $\theta = 1$. Apparently, the regulatory network is considerably robust regarding the choice of $\theta$ and the setting of regulatory logic on E2F1 (i.e. $K_{\mathrm{E2F1},R}$ where $R \subseteq \{\mathrm{E2F1}, \mathrm{pRB}\}$).

**Continuous-time deterministic model** Now we consider a quantitative model of the gene regulatory network traditionally formalized by means of so-called Hill kinetics that abstracts from unknown elementary reactions occurring during processes of protein identification and its regulation. However, from the perspective of computer analysis, Hill kinetics introduces rational polynomial functions into the right-hand sides if ODEs. Course of the Hill function modeling positive regulation is shown in Fig. 9.




$K_{pRB},\emptyset = 1$ $\quad$ $K_{E2F1},\emptyset = 2$
$K_{pRB,\{pRB\}} =?$ $\quad$ $K_{E2F1,\{E2F1\}} =?$
$K_{pRB,\{E2F1\}} =?$ $\quad$ $K_{E2F1,\{pRB\}} =?$
$K_{pRB,\{E2F1,pRB\}} =?$ $\quad$ $K_{E2F1,\{E2F1,pRB\}} =?$

$$\frac{d[pRB]}{dt} = k_1 \frac{[E2F1]}{0.5+[E2F1]} \frac{0.5}{0.5+[pRB]} - \gamma_{pRB}[pRB]$$
$$\frac{d[E2F1]}{dt} = k_p + k_2 \frac{a^2+[E2F1]^2}{16+[E2F1]^2} \frac{5}{5+[pRB]} - \gamma_{E2F1}[E2F1]$$

**Fig. 8.** (left) Genetic regulatory network controlling the $G_1/S$ transition. (right) Regulatory logic employed for the qualitative model. (bottom) The original ODE model system that makes the quantitative model of the network.

To analyze the model at the level of quantitative kinetics, we again need to simplify the continuous-time deterministic model. Similarly as in the previous case study, we translate the model into the discrete-time discrete-value domain. To this end, we employ the piece-wise multi-affine abstraction (PMA) of the non-linear ODE model shown in Fig. 8. This abstraction has two consecutive steps:

- transforming the non-linear ODE model into a piece-wise multi-affine (PMA) model and
- performing rectangular abstraction to transform the PMA model into a finite automaton (the so-called rectangular abstraction transition system)

The first step has been defined in [25], the main idea is to get rid of the rational polynomial functions appearing in right-hand sides of ODEs. As illustrated in Fig. 9, this is done by approximating each of them by a so-called ramp function

defined in the following way:

$$r^+(x_i, \theta_i, \theta_i') = \begin{cases} 0, & \text{if } x_i \leq \theta_i, \\ \frac{x_i - \theta_i}{\theta_i' - \theta_i}, & \text{if } \theta_i < x_i < \theta_i', \\ 1, & \text{if } x_i \geq \theta_i'. \end{cases}$$

The ramp function approximates the non-linear sigmoid function by a piece-wise affine curve. A sum of scaled ramp functions approximating individual segments of the original non-linear curve can be employed.

Second, the PMA model is abstracted by using the rectangular abstraction as introduced in Section 4.1. Since Theorem 1 extends to piece-wise multi-affine models with no restrictions [25], the abstraction procedure is the same as in the case of multi-affine systems. Again, the rectangular abstraction partitions the domain of every unknown parameter domain into a finite number of intervals.

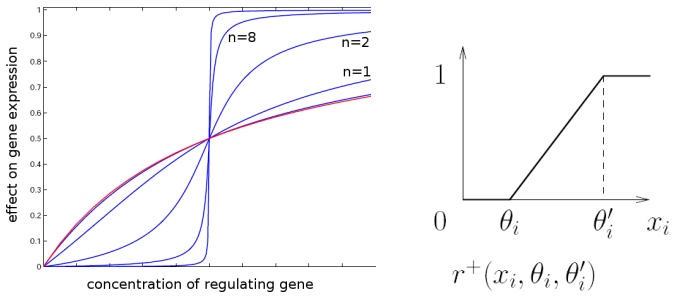



**Fig. 9.** Sigmoid (Hill) function for positive regulation abstracted by a corresponding ramp function. The steepness is affected by the exponent appearing in Hill functions, here denoted $n$.

The parameters in the original ODE model have been estimated by employing the bifurcation analysis [157]. We show how our alternative method based on model checking can be employed for identification of parametrizations satisfying the required specification. Our PMA abstraction of this system is shown in Fig. 10. Each function $\varrho_i(x)$ is defined as a sum of several ramp-functions that gradually approximate the respective regulatory Hill curve by a polyline.

Since we detected bistability by using the qualitative model above, it follows to find how this phenomenon is affected by the setting of (quantitative) kinetic parameters. As shown in [157], the steady behavior of this system is strongly influenced by the degradation coefficient $\gamma_{pRB}$. Fig. 11 shows the vector field of the above system for two different values of $\gamma_{pRB}$.

Similarly to the previous case, to express dynamical properties of paths in rectangular abstraction of an $n$-dimensional model $\mathcal{M}$ we employ traditional Linear Temporal Logic (LTL) built over atomic propositions $AP$:

$$AP = \left\{ x_i \odot \theta_j^i \mid 1 \leq i \leq n, 1 \leq j \leq \zeta_i \}, \odot \in \{<, >\} \right\}.$$

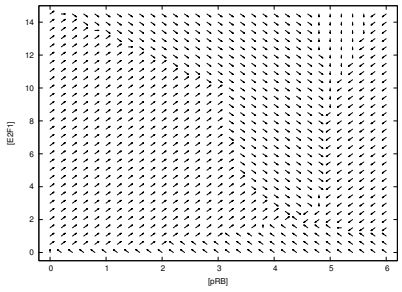
$$\frac{d[pRB]}{dt} = k_1 \varrho_1(pRB, E2F1) - \gamma_{pRB}[pRB]$$
$$\frac{d[E2F1]}{dt} = k_p + k_2 \varrho_2(pRB, E2F1) - \gamma_{E2F1}[E2F1]$$

$$\begin{aligned}\varrho_1(pRB, E2F1) \;=\; & (0.85r^+(E2F1, 0, 3) + 0.1r^+(E2F1, 3, 10) + 0.05r^+(E2F1, 10, 50)) \\ & \cdot(0.85r^-(pRB, 0, 2) + 0.1r^-(pRB, 2, 5) + 0.05r^-(pRB, 5, 80)) \\ \varrho_2(pRB, E2F1) \;=\; & (0.85r^+(E2F1, 0, 10) + 0.1r^+(E2F1, 10, 20) + 0.05r^+(E2F1, 20, 80)) \\ & \cdot(0.5r^-(pRB, 0, 5) + 0.2r^-(pRB, 5, 10) + 0.15r^-(pRB, 10, 30) + 0.15r^-(pRB, 30, 130))\end{aligned}$$

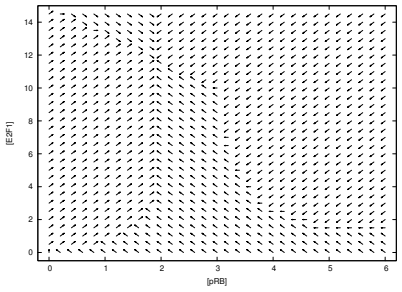
**Fig. 10.** Piece-wise multi-affine abstraction (PMA model) for the $G_1/S$ transition regulatory network.

Our goal is to determine the set of parameters in the range $[0.01, 1]$ for which the concentration of $E2F1$ is greater than 8 in the stable state. This phenomenon can be specified in terms of an LTL formula $\varphi = \mathbf{FG}([E2F1] > 8)$.

We executed the colored model checking algorithm implemented in the PEPMC tool (see Section 3.5) for the model described above and the property $\varphi$. Since the abstraction technique has the overapproximative character, some of counterexamples found by model checking can be false-positive paths. Therefore the result is an under-approximated set of parameter valuations under which the property $\varphi$ is satisfied. Owing to the fact that the property $\varphi$ is a liveness property, many of the counterexamples can be paths on which the time does not really proceed (the so-called time-convergent paths). In [21] we have shown a way of how the model checking procedure for this specific model can be elaborated to avoid unwanted time-convergent paths. By applying the algorithm to the model described above we were able to prove that for $\gamma_{pRB} > 0.053$, the system stabilizes with $E2F1 > 8$.




(a) $\gamma_{pRB} = 0.01$, the system stabilizes with $E2F1 < 3$




(b) $\gamma_{pRB} = 0.1$, the system stabilizes with $E2F1 > 11$

**Fig. 11.** Vector field of the liveness model.

| Gene $a$ interactions | | Gene $b$ interactions | |
|---|---|---|---|
| $a \to a + A$ | 1 | $b \to b + B$ | 0.05 |
| $aB \to aB + A$ | 1 | $bB \to bB + B$ | 1 |
| $A + a \leftrightarrow aA$ | $100; 10$ | $A + b \leftrightarrow bA$ | $100; 10$ |
| $B + a \leftrightarrow aB$ | $100; 10$ | $B + b \leftrightarrow bB$ | $100; 10$ |
| Protein degradation | | | |
| $A \to$ | $\gamma_A$ | $B \to$ | $\gamma_B$ |

(a)

| Property | # iter. | # subsp. | time[h] |
|---|---|---|---|
| (1a) | $1.2{\cdot}10^6$ | 153 | 9 |
| (2a) | $2.0{\cdot}10^6$ | 69 | 5.5 |
| (3a) | $2.0{\cdot}10^6$ | 66 | 4.5 |
| (1b) | $4.0{\cdot}10^6$ | 159 | 10.5 |
| (2b) | $4.0{\cdot}10^6$ | 132 | 8 |
| (3b) | $4.0{\cdot}10^6$ | 80 | 5 |

(b)

**Fig. 12.** ($a$) Stochastic mass action model of the $G_1/S$ regulatory circuit – $A$ denotes the protein pRB, $B$ denotes E2F1, $a, b$ represent genes, $aA, aB, bA, bB$ represent transcription factor-gene promoter complexes ($b$) Computation results.

**Continuous-time stochastic model** We have translated the original ODE model into the framework of stochastic mass action kinetics [94]. The resulting reactions are shown in Fig. 12a. Since the detailed knowledge of elementary chemical reactions occurring in the process of transcription and translation is incomplete, we use the simplified form as suggested in [80]. In the minimalistic setting, the reformulation requires addition of rate parameters describing the transcription factor–gene promoter interaction while neglecting cooperativeness of transcription factors activity. Our parametrization is based on time-scale orders known for the individual processes [162]. Moreover, we assume the numbers of $A$ and $B$ are bounded by 10 molecules. The bound is calibrated with respect to the original ODE model and reflects the character of the two steady states. All other species are bounded by the initial number of DNA molecules (genes $a$ and $b$) which is conserved and set to 1. The model is translated into a CTMC which has 1078 states and 5919 transitions.

An interesting biologically relevant problem is to predict how the population of cells implements this regulatory circuit in reaction to mitogenic stimulation and under presence of noise. Low molecular numbers typical for DNA and proteins molecules make the gene regulation highly sensitive to noise. Since mitogenic stimulation influences the degradation rate of $A$, our goal is to study the population distribution around the low and high steady state.

In particular, we consider three hypotheses: (1) stabilization in the low mode where $B < 3$, (2) stabilization in the high mode where $B > 5$, (3) stabilization in the high mode where $B > 7$ ((3) is more focused than (2)). All the hypotheses are expressed within time horizon 1000 seconds reflecting the time scale of gene regulation response. We employ two alternative CSL formulations to express each of the three hypothesis.

First, we express the property of being inside the given bound during the time interval $I = [500, 1000]$ using globally operator: (1a) $\mathsf{P}_{\sim ?}[\mathsf{G}^I\,(B < 3)]$, (2a) $\mathsf{P}_{\sim ?}[\mathsf{G}^I\,(B > 5)]$ and ($3a$) $\mathsf{P}_{\sim ?}[\mathsf{G}^I\,(B > 7)]$. The interval starts from 500 seconds in order to bridge the initial fluctuation region and let the system stabilize.

For the fixed valuations of parameters, quantitative CSL model checking can be used to answer the above mentioned questions. For this purpose, PRISM pro-

vides the most suitable tool [125]. The papers on applying PRISM to biological models [102, 127], the book [112], and the tutorials available at the tool webpage provide a good source for this topic.

Here we focus on analysing the above stated properties with respect to the potential uncertainty in parameters. In particular, we explore the effect of the parameter $\gamma_A$ on the probability of the properties. According to [157], we consider the parameter space $\gamma_A \in [0.005, 0.5]$.

The technique employed for the parameter exploration is described in [40], it is implemented on the top of PRISM. The basic notion is the *landscape function* that for each parameter point from the inspected parameter space returns the quantitative model checking result for the respective CTMC determined by the parameter point and the given property. Computation of the landscape function is based on automatic decomposition of the given parameter space with respect to how it influences the model dynamics. The computation is approximative and provides the result within a required absolute error bound.

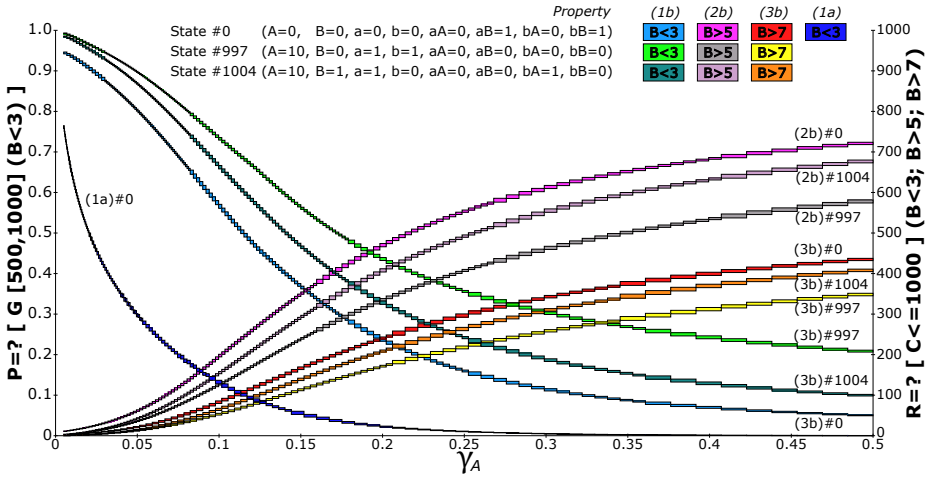



**Fig. 13.** Landscape functions of properties *(1a,1b,2b,3b)* for parameter $\gamma_A \in [0.005, 0.5]$ and initial states #0, #997 and #1004. The left Y-axis scale corresponds to *(1a)*, the right to *(1b,2b,3b)*.

Since the stochastic noise causes molecules to repeatedly escape the requested bound, the resulting probability is significantly lower than expected. Namely, in cases (2a) and (3a) the resulting probability is close to 0 for the whole parameter space. Moreover, the selection of an initial state has only a negligible impact on the result. Therefore, in Fig. 13 only the resulting probability for case (1a) and a single selected initial state is visualized.

Second, we use a cumulative reward property [126] to capture the fraction of the time the system has the required number of molecules within the time interval

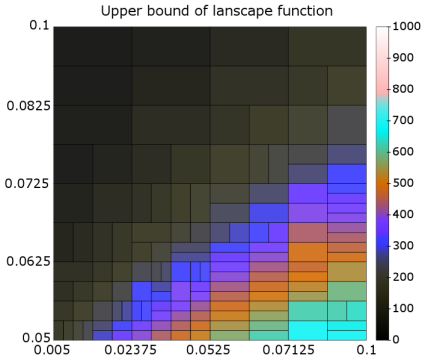



**Fig. 14.** Landscape function for property $(3b)$, initial state #0 $(A = 0, B = 0, a = 0, b = 0, aA = 0, aB = 1, bA = 0, bB = 1)$ and two-dimensional parameter space $(\gamma_A, \gamma_B) \in [0.005, 0.1] \times [0.05, 0.1]$ (represented by X and Y axes, respectively). The upper bound of the landscape function is illustrated.

$[0, 1000]$: $(1b)$ $\mathsf{R}_{\sim ?}[\mathsf{C}^{\leq t}](B < 3)$, $(2b)$ $\mathsf{R}_{\sim ?}[\mathsf{C}^{\leq t}](B > 5)$, $(3b)$ $\mathsf{R}_{\sim ?}[\mathsf{C}^{\leq t}](B > 7)$ where $t = 1000$ and $\mathsf{R}_{\sim ?}[\mathsf{C}^{\leq t}](B \sim X)$ denotes that state reward $\rho$ is defined such that $\forall s \in \mathbb{S}.\rho(s) = 1$ iff $B \sim X$ in $s$. The result is visualized for three selected initial states in Fig. 13.

Fig. 13 also illustrates inaccuracy of our approach with respect to the absolute error bound $\textsc{Err} = 0.01$ by means of small rectangles depicting approximations of the resulting probabilities and expected rewards. The analyses predict that the distribution of the low steady mode interferes with the distribution of the high steady mode. It confirms bistability predicted in [157] but in contrast to ODE analysis our method shows how the population of cells distributes around the two stable states. Results of computations including the number of iterations performed during parametrized uniformization, numbers of resulting subspaces and execution times in hours, are presented in Fig. 12b.

Finally, to see how degradation rates of $A$ and $B$ cooperate in affecting property $(3b)$, we explore two-dimensional parameter space $(\gamma_A, \gamma_B) \in [0.005, 0.1] \times [0.05, 0.1]$. Fig. 14 illustrates the computed upper bound of the landscape function for initial state #0. The result predicts antagonistic relation between the degradation rates which is in agreement with the ODE model [157].

## References

1. Allmaier, S., Dalibor, S., Kreische, D.: Parallel Graph Generation Algorithms for Shared and Distributed Memory Machines. In: Parallel Computing Conference (PARCO). LNCS, vol. 1253, pp. 207–218. Springer (1997)
2. Alur, R., Courcoubetis, C., Dill, D.: Model-checking in dense real-time. Information and Computation 104, 2–34 (1993)
3. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM 43(1), 116–146 (1996)

4. Alur, R., Henzinger, T.A.: A really temporal logic. J. ACM 41(1), 181–203 (1994)
5. Antoniotti, M., Policriti, A., Ugel, N., Mishra, B.: Model building and model checking for biochemical processes. Cell Biochemistry and Biophysics 38, 271–286 (2003)
6. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.: Verifying continuous time Markov chains. In: Conference on Computer Aided Verification (CAV), LNCS, vol. 1102, pp. 269–276. Springer (1996)
7. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model Checking Continuous-Time Markov Chains by Transient Analysis. In: Conference on Computer Aided Verification (CAV), LNCS, vol. 1855, pp. 358–372. Springer (2000)
8. Ballarini, P., Forlin, M., Mazza, T., Prandi, D.: Efficient parallel statistical model checking of biochemical networks. In: Parallel and Distributed Methods in verifiCation (PDMC). EPTCS, vol. 14, pp. 47–61 (2009)
9. Ballarini, P., Guerriero, M.L.: Query-based verification of qualitative trends and oscillations in biochemical systems. Theor. Comput. Sci. 411(20), 2019–2036 (2010)
10. Ballarini, P., Guido, R., Mazza, T., Prandi, D.: Taming the complexity of biological pathways through parallel computing. Briefings in Bioinformatics 10(3), 278–288 (2009)
11. Barbuti, R., Caravagna, G., Maggiolo-Schettini, A., Milazzo, P., Tini, S.: Foundational aspects of multiscale modeling of biological systems with process algebras. Theor. Comput. Sci. 431, 96–116 (2012)
12. Barnat, J., Bauch, P., Brim, L., Češka, M.: Computing Strongly Connected Components in Parallel on CUDA. In: International Parallel & Distributed Processing Symposium (IPDPS). pp. 541–552. IEEE Computer Society (2011)
13. Barnat, J., Brim, L., Krejci, A., Streck, A., Safranek, D., Vejnar, M., Vejpustek, T.: On Parameter Synthesis by Parallel Model Checking. IEEE/ACM Transactions on Computational Biology and Bioinformatics 9(3), 693–705 (2012)
14. Barnat, J., Brim, L., Ročkai, P.: Scalable Multi-core LTL Model-Checking. In: Model Checking Software (SPIN). LNCS, vol. 4595, pp. 187–203. Springer (2007)
15. Barnat, J., Brim, L., Ročkai, P.: A Time-Optimal On-the-Fly Parallel Algorithm for Model Checking of Weak LTL Properties. In: Formal Methods and Software Engineering (ICFEM). LNCS, vol. 5885, pp. 407–425. Springer (2009)
16. Barnat, J., Brim, L., Ročkai, P.: Parallel Partial Order Reduction with Topological Sort Proviso. In: Software Engineering and Formal Methods (SEFM). pp. 222–231. IEEE Computer Society (2010)
17. Barnat, J., Brim, L., Stříbrná, J.: Distributed LTL Model-Checking in SPIN. In: Model Checking Software (SPIN). LNCS, vol. 2057, pp. 200–216. Springer (2001)
18. Barnat, J., Brim, L., Černá, I., Dražan, S., Fabriková, J., Láník, J., Šafránek, D., Ma, H.: BioDiVinE: A Framework for Parallel Analysis of Biological Models. In: Computational Models for Cell Processes (COMPMOD). EPTCS, vol. 6, pp. 31–45 (2009)
19. Barnat, J., Brim, L., Černá, I., Moravec, P., Ročkai, P., Šimeček, P.: DiVinE – A Tool for Distributed Verification. In: Computer Aided Verification (CAV). LNCS, vol. 4144, pp. 278–281. Springer (2006)
20. Barnat, J., Brim, L., Šafránek, D.: High-Performance Analysis of Biological Systems Dynamics with the DiVinE Model Checker. Briefings in Bioinformatics 11(3), 301–312 (2010)
21. Barnat, J., Brim, L., Šafránek, D., Vejnár, M.: Parameter Scanning by Parallel Model Checking with Applications in Systems Biology. In: Parallel and Dis-

tributed Methods in Verification and High Performance Computational Systems Biology (HiBi/PDMC 2010). pp. 95–104. IEEE Computer Society (2010)
22. Barnat, J., Ročkai, P.: Shared Hash Tables in Parallel Model Checking. In: Parallel and Distributed Methods in verifiCation (PDMC). ENTCS, vol. 198, pp. 79–91 (2008)
23. Barnat, J., Bauch, P., Brim, L., Češka, M.: Designing fast LTL model checking algorithms for many-core GPUs. Journal of Parallel and Distributed Computing 72(9), 1083–1097 (2012)
24. Bartocci, E., Corradini, F., Merelli, E., Tesei, L.: Detecting synchronisation of biological oscillators by model checking. Theoretical Computer Science 411(20), 1999 – 2018 (2010)
25. Batt, G., Belta, C., Weiss, R.: Model checking genetic regulatory networks with parameter uncertainty. In: Hybrid Systems: Computation and Control (HSCC). LNCS, vol. 4416, pp. 61–75. Springer (2007)
26. Batt, G., Page, M., Cantone, I., Goessler, G., Monteiro, P., de Jong, H.: Efficient parameter search for qualitative models of regulatory networks using symbolic model checking. Bioinformatics 26(18), 603–610 (2010)
27. Batt, G., Salah, R.B., Maler, O.: On timed models of gene networks. In: Formal Modeling and Analysis of Timed Systems (FORMATS). pp. 38–52. LNCS, Springer (2007)
28. Behrmann, G., Hune, T.S., Vaandrager, F.W.: Distributed Timed Model Checking — How the Search Order Matters. In: Computer Aided Verification (CAV). LNCS, vol. 1855, pp. 216–231. Springer (2000)
29. Behrmann, G., David, A., Larsen, K.G.: A tutorial on UPPAAL. In: Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, (SFM-RT). pp. 200–236. No. 3185 in LNCS, Springer–Verlag (2004)
30. Belta, C., Habets, L.: Controlling a class of nonlinear systems on rectangles. IEEE Transactions on Automatic Control 51(11), 1749–1759 (2006)
31. Bernot, G., Comet, J.P., Richard, A., Guespin, J.: Application of formal methods to biological regulatory networks: extending thomas asynchronous logical approach with temporal logic. Journal of Theoretical Biology 229(3), 339–347 (2004)
32. Bonzanni, N., Krepska, E., Feenstra, K.A., Fokkink, W., Kielmann, T., Bal, H.E., Heringa, J.: Executing multicellular differentiation: quantitative predictive modelling of *C.elegans* vulval development. Bioinformatics 25(16), 2049–2056 (2009)
33. Bortolussi, L., Hillston, J.: Fluid model checking. In: Concurrency Theory (CONCUR), LNCS, vol. 7454, pp. 333–347. Springer Berlin Heidelberg (2012)
34. Bortolussi, L., Policriti, A.: Hybrid systems and biology. In: Formal Methods for Computational Systems Biology, 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2008. pp. 424–448. LNCS, Springer (2008)
35. Bosnacki, D., Edelkamp, S., Sulewski, D.: Efficient Probabilistic Model Checking on General Purpose Graphics Processors. In: Model Checking Software (SPIN). LNCS, vol. 5578, pp. 32–49. Springer (2009)
36. Bosnacki, D., ten Eikelder, H.M.M., Steijaert, M.N., de Vink, E.P.: Stochastic analysis of amino acid substitution in protein synthesis. In: Computational Methods in Systems Biology (CMSB). pp. 367–386 (2008)
37. Brenan, K.E., Campbell, S.L., Petzold, L.R.: Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. SIAM (1987)

38. Brim, L., Černá, I., Moravec, P., Šimša, J.: Accepting predecessors are better than back edges in distributed LTL model-checking. In: Formal Methods in Computer Aided Design (FMCAD). LNCS, vol. 4144, pp. 352–366. Springer (2004)
39. Brim, L., Barnat, J.: Platform Dependent Verification: On Engineering Verification Tools for 21st Century. In: Parallel and Distributed Methods in verifiCation (PDMC). EPTCS, vol. 72, pp. 1–12 (2011)
40. Brim, L., Češka, M., Dražan, S., Šafránek, D.: Exploring parameter space of stochastic biochemical systems using quantitative model checking. Tech. rep., Faculty of Informatics, Masaryk University (2013), `http://sybila.fi.muni.cz/TR-01-2013.pdf`
41. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, L.J.: Symbolic model checking: $10^{20}$ states and beyond. Information and Computation 98(2), 142–170 (1992)
42. Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R.: Analysis of signalling pathways using continuous time markov chains. In: Transactions on Computational Systems Biology VI, LNCS, vol. 4220, pp. 44–67. Springer (2006)
43. Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine learning biochemical networks from temporal logic properties. In: Transactions on Computational Systems Biology VI, pp. 68–94. LNCS, Springer-Verlag, Berlin, Heidelberg (2006)
44. Campagna, D., Piazza, C.: Hybrid automata in systems biology: How far can we go? In: From Biology to Concurrency and Back (FBTC). ENTCS, vol. 229, pp. 93 – 108 (2009)
45. Caravagna, G., Hillston, J.: Modeling biological systems with delays in Bio-PEPA. In: Proceedings Fourth Workshop on Membrane Computing and Biologically Inspired Process Calculi 2010. EPTCS, vol. 40, pp. 85–101 (2010)
46. Carrillo, M., Góngora, P.A., Rosenblueth, D.A.: An overview of existing modeling tools making use of model checking in the analysis of biochemical networks. Front Plant Sci. 3:155, 1–13 (2012)
47. Caselli, S., Conte, G., Marenzoni, P.: Parallel state space exploration for GSPN models. In: Applications and Theory of Petri Nets 1995. LNCS, vol. 935, pp. 181–200. Springer Verlag (1995)
48. Černá, I., Pelánek, R.: Distributed explicit fair cycle detection. In: Model Checking Software (SPIN). LNCS, vol. 2648, pp. 49–73. Springer (2003)
49. Chaouiya, C.: Petri net modelling of biological networks. Briefings in Bioinformatics 8(4), 210–219 (2007)
50. Chaouiya, C., Remy, E., Mossé, B., Thieffry, D.: Qualitative analysis of regulatory graphs: A computational tool based on a discrete formal framework. In: Positive Systems, LNCIS, vol. 294, pp. 830–832. Springer (2003)
51. Che, S., Li, J., Sheaffer, J., Skadron, K., Lach, J.: Accelerating Compute-Intensive Applications with GPUs and FPGAs. In: IEEE Symposium on Application Specific Processors (SASP). pp. 101–107. IEEE Computer Society (2008)
52. Ciardo, G., Gluckman, J., Nicol, D.: Distributed state-space generation of discrete-state stochastic models. INFORMS J. Comp. 10(1), 82–93 (1998)
53. Ciardo, G.: Automated parallelization of discrete state-space generation. J. Parallel Distrib. Comput. 47, 153–167 (1997)
54. Cimatti, A., Clarke, E., Giunchiglia, F., Roveri, M.: NuSMV: a new symbolic model checker. J. Softw. Tools Technol. Transf. 2(410–425) (2000)
55. Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: An OpenSource Tool for Symbolic Model

Checking. In: Computer Aided Verification (CAV), LNCS, vol. 2404, pp. 359–364. Springer Berlin Heidelberg (2002)
56. Ciocchetta, F., Hillston, J.: Bio-PEPA: A framework for the modelling and analysis of biological systems. Theor. Comput. Sci. 410(33-34), 3065–3084 (2009)
57. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. 8(2), 244–263 (1986)
58. Clarke, E.M., Enders, R., Filkorn, T., Jha, S.: Exploiting symmetry in temporal logic model checking. Form. Methods Syst. Des. 9(1-2), 77–104 (1996)
59. Clarke, Jr., E.M., Grumberg, O., Peled, D.A.: Model checking. MIT Press (1999)
60. Clarke, E., Zuliani, P.: Statistical Model Checking for Cyber-Physical Systems. In: Automated Technology for Verification and Analysis (ATVA), LNCS, vol. 6996, pp. 1–12. Springer Berlin Heidelberg (2011)
61. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Progress on the State Explosion Problem in Model Checking. In: Informatics - 10 Years Back. 10 Years Ahead. LNCS, vol. 2000, pp. 176–194. Springer (2001)
62. Collins, P., Habets, L.C., van Schuppen, J.H., Černá, I., Fabriková, J., Šafránek, D.: Abstraction of biochemical reaction systems on polytopes. In: Proceedings of the 18th IFAC World Congress. vol. 18, pp. 14869–14875 (2011)
63. Courcoubetis, C., Vardi, M., Wolper, P., Yannakakis, M.: Memory-Efficient Algorithms for the Verification of Temporal Properties. Formal Methods in System Design 1, 275–288 (1992)
64. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. J. ACM 42(4), 857–907 (Jul 1995)
65. Crudu, A., Debussche, A., Radulescu, O.: Hybrid stochastic simplifications for multiscale gene networks. BMC Systems Biology 3(1), 89 (2009)
66. NVIDIA CUDA Compute Unified Device Architecture - Programming Guide Version 2.0, (2009), `http://www.nvidia.com/object/cuda\_develop.html`
67. Dang, T., Guernic, C.L., Maler, O.: Computing reachable states for nonlinear biological models. Theor. Comput. Sci. 412(21), 2095–2107 (2011)
68. Danos, V., Laneve, C.: Formal molecular biology. Theor. Comput. Sci. 325(1), 69–110 (2004)
69. Darling, R., Norris, J.: Differential equation approximations for markov chains. Probab. Surveys 5, 37–79 (2008)
70. David, A., Du, D., Larsen, K.G., Legay, A., Mikucionis, M., Poulsen, D.B., Sedwards, S.: Statistical model checking for stochastic hybrid systems. In: Hybrid Systems and Biology (HSB). EPTCS, vol. 92, pp. 122–136 (2012)
71. Derman, C.: Finite State Markovian Decision Processes. Academic Press, Inc., Orlando, FL, USA (1970)
72. Didier, F., Henzinger, T.A., Mateescu, M., Wolf, V.: Fast Adaptive Uniformization for the Chemical Master Equation. In: Parallel and Distributed Methods in Verification and High Performance Computational Systems Biology (HiBi/PDMC 2009). pp. 118–127. IEEE Computer Society (2009)
73. Didier, F., Henzinger, T.A., Mateescu, M., Wolf, V.: Sabre: A tool for stochastic analysis of biochemical reaction networks. CoRR abs/1005.2819 (2010)
74. Dluhoš, P., Brim, L., Šafránek, D.: On expressing and monitoring oscillatory dynamics. In: Hybrid Systems and Biology (HSB). EPTCS, vol. 92, pp. 73–87 (2012)
75. Doi, A., Fujita, S., Matsuno, H., Nagasaki, M., Miyano, S.: Constructing Biological Pathway Models with Hybrid Functional Petri Nets. In Silico Biology 4(3), 271–291 (2004)

76. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Computer Aided Verification (CAV). pp. 167–170. LNCS, Springer-Verlag, Berlin, Heidelberg (2010)
77. Donzé, A., Clermont, G., Langmead, C.J.: Parameter synthesis in nonlinear dynamical systems: Application to systems biology. Journal of Computational Biology 17(3), 325–336 (2010)
78. Edelkamp, S., Sulewski, D.: Parallel State Space Search on the GPU (2009), symposium on Combinatorial Search (SoCS)
79. Eker, S., Knapp, M., Laderoute, K., Lincoln, P., Meseguer, J., Sonmez, K.: Pathway logic: Symbolic analysis of biological signaling. In: Pacific Symposium on Biocomputing. pp. 400–412 (2002)
80. El Samad, H., Khammash, M., Petzold, L., Gillespie, D.: Stochastic Modelling of Gene Regulatory Networks. Int. J. of Robust and Nonlinear Control 15(15), 691–711 (2005)
81. Emerson, E.A., Sistla, A.P.: Symmetry and model checking. Form. Methods Syst. Des. 9(1-2), 105–131 (1996)
82. Engl, H.W., Flamm, C., Kügler, P., Lu, J., Müller, S., Schuster, P.: Inverse problems in systems biology. Inverse Problems 25(12), 123014 (2009)
83. Ezekiel, J., Lüttgen, G., Ciardo, G.: Parallelising symbolic state-space generators. In: CAV. LNCS, vol. 4590, pp. 268–280. Springer (2007)
84. Fages, F., Soliman, S.: 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 (2008)
85. Fages, F., Soliman, S., Rivier, C.N.: Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. Journal of Biological Physics and Chemistry 4(2), 64–73 (2004)
86. Fages, F., Rizk, A.: On temporal logic constraint solving for analyzing numerical data time series. Theor. Comput. Sci. 408(1), 55–65 (2008)
87. Fisher, J., Henzinger, T.A.: Executable cell biology. Nature Biotechnology 25(11), 1239–1249 (2007)
88. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated verification techniques for probabilistic systems. In: Formal Methods for Eternal Networked Software Systems (SFM). LNCS, vol. 6659, pp. 53–113. Springer (2011)
89. Fromentin, J., Eveillard, D., Roux, O.: Hybrid modeling of biological networks: mixing temporal and qualitative biological properties. BMC Systems Biology 4(1), 79 (2010)
90. Galpin, V., Hillston, J., Bortolussi, L.: HYPE Applied to the Modelling of Hybrid Biological Systems. ENTCS 218, 33–51 (2008)
91. Garavel, H., Lang, F., Mateescu, R.: CADP 2006: A toolbox for the construction and analysis of distributed processes. In: Computer Aided Verification (CAV). LNCS, vol. 4590, pp. 153–163. Springer (2007)
92. Garavel, H., Mateescu, R., Smarandache, I.: Parallel State Space Construction for Model-Checking. In: Model Checking Software (SPIN). LNCS, vol. 2057, pp. 216–234. Springer (2001)
93. Geldenhuys, J., de Villiers, P.J.A.: Runtime efficient state compaction in SPIN. In: Model Checking Software (SPIN). LNCS, vol. 1680, pp. 12–21. Springer (1999)
94. Gillespie, D.T.: Exact Stochastic Simulation of Coupled Chemical Reactions. Journal of Physical Chemistry 81(25), 2340–2381 (1977)
95. Gillespie, D.T.: A rigorous derivation of the chemical master equation. Physica A: Statistical Mechanics and its Applications 188(13), 404 – 425 (1992)

96. Gillespie, D.T.: Stochastic Simulation of Chemical Kinetics. Annual Review of Physical Chemistry 58(1), 35–55 (2007)
97. Goethem, S.V., Jacquet, J.M., Brim, L., Šafránek, D.: Timed modelling of gene networks with arbitrary expression level discretization. In: Interactions between Computer Science and Biology. ENTCS, Elsevier (2013), in press.
98. Grumberg, O., Heyman, T., Schuster, A.: A work-efficient distributed algorithm for reachability analysis. In: Computer Aided Verification (CAV). LNCS, vol. 2725, pp. 54–66. Springer (2003)
99. Habets, L., van Schuppen, J.H.: A control problem for affine dynamical systems on a full-dimensional polytope. Automatica 40(1), 21 – 35 (2004)
100. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6, 512–535 (1994)
101. Haverkort, B.R., Bell, A., Bohnenkamp, H.C.: On the efficient sequential and distributed generation of very large Markov chains from stochastic Petri nets. In: Petri Net and Performance Models (PNPM). pp. 12–21. IEEE Computer Society Press (1999)
102. Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. Theoretical Computer Science 319(3), 239–257 (2008)
103. Heiner, M., Gilbert, D., Donaldson, R.: 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). LNCS, vol. 5016, pp. 215–264. Springer (2008)
104. Heljanko, K., Khomenko, V., Koutny, M.: Parallelisation of the Petri Net Unfolding Algorithm. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS). LNCS, vol. 2280, pp. 371–385. Springer (2002)
105. Henzinger, T.: The theory of hybrid automata. In: Logic in Computer Science (LICS). pp. 278 –292. IEEE Computer Society (1996)
106. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing. pp. 373–382. ACM (1995)
107. Holzmann, G.J.: The Spin Model Checker: Primer and Reference Manual. Addison-Wesley (2003)
108. Holzmann, G.J.: A Stack-Slicing Algorithm for Multi-Core Model Checking. Electonic Notes in Theoretical Computer Science 198(1), 3–16 (2008)
109. Holzmann, G.J., Bosnacki, D.: The design of a multicore extension of the spin model checker. IEEE Trans. Software Eng. 33(10), 659–674 (2007)
110. Horn, F., Jackson, R.: General mass action kinetics. Archive for Rational Mechanics and Analysis 47, 81–116 (1972), 10.1007/BF00251225
111. Inggs, C.P., Barringer, H.: CTL$^*$ Model Checking on a Shared-Memory Architecture. Electronic Notes in Theoretical Computer Science 128(3), 107–123 (2005)
112. Iyengar, M.S.: Symbolic Systems Biology: Theory and Methods. Jones & Bartlett Publishers (2010)
113. Jayachandran, G., Vishal, V., Pande, V.S.: Using massively parallel simulations and Markovian models to study protein folding: examining the villin head-piece. Journal of Chemical Physics 124(6), 903914 (2006)
114. Jha, S.K., Clarke, E.M., Langmead, C., Legay, A., Platzer, A., Zuliani, P.: A bayesian approach to model checking biological systems. In: Computational Methods in Systems Biology (CMSB), LNCS, vol. 5688, pp. 218–234. Springer (2009)
115. de Jong, H.: Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. Journal of Computational Biology 9(1), 67–103 (2002)

116. de Jong, H., Gouzé, J., Hernandez, C., Page, M., Sari, T., Geiselmann, J.: Qualitative simulations of genetic regulatory networks using piecewise linear models. Bull. Math. Biol. 66, 301–340 (2004)
117. Kahn, A.B.: Topological sorting of large networks. Commun. ACM 5(11), 558–562 (1962)
118. Keener, J.P., Sneyd, J.: Mathematical Physiology. Springer (1998)
119. Khademi, S., III, J.O., Remis, J., Robles-Colmenares, Y., Miercke, L., Stroud, R.: Mechanism of ammonia transport by Amt/MEP/Rh: Structure of AmtB at 1.35. Science 305(5690), 1587–1594 (2004)
120. Kholodenko, B.N.: Cell-signalling dynamics in time and space. Nature Molecular Cell Biology 7, 165–176 (2006)
121. Klarner, H., Streck, A., Šafránek, D., Kolčák, J., Siebert, H.: Parameter identification and model ranking of thomas networks. In: Computational Methods in Systems Biology (CMSB), pp. 207–226. LNCS, Springer (2012)
122. Knottenbelt, W., Mestern, M., Harrison, P., , Kritzinger, P.: Probability, parallelism and the state space exploration problem. In: Modelling, Techniques and Tools (TOOLS). LNCS, vol. 1469, pp. 165–179. Springer (1998)
123. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems 2, 255–299 (1990)
124. Kumar, R., Mercer, E.G.: Load Balancing Parallel Explicit State Model Checking. In: Parallel and Distributed Methods in Verification (PDMC). ENTCS, vol. 128, pp. 19–34. Elsevier (2005)
125. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Computer Aided Verification (CAV). LNCS, vol. 6806, pp. 585–591. Springer (2011)
126. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. In: Formal Methods for Performance Evaluation: 7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM), LNCS, vol. 4486, pp. 220–270. Springer (2007)
127. Kwiatkowska, M.Z., Norman, G., Parker, D.: Using probabilistic model checking in systems biology. SIGMETRICS Performance Evaluation Review 35(4), 14–21 (2008)
128. Laarman, A., van de Pol, J., Weber, M.: Boosting Multi-Core Reachability Performance with Shared Hash Tables. In: Formal Methods in Computer-Aided Design (FMCAD). pp. 247–255. IEEE Computer Science (2010)
129. Lerda, F., Sisto, R.: Distributed-memory Model Checking with SPIN. In: Proc. of the 5th International SPIN Workshop. LNCS, vol. 1680, pp. 22–39. Springer-Verlag (1999)
130. Lerda, F., Visser, W.: Addressing Dynamic Issues of Program Model Checking. In: Model Checking of Software (SPIN). LNCS, vol. 2057, pp. 80–102. Springer (2001)
131. Ma, H., Boogerd, F., Goryanin, I.: Modelling nitrogen assimilation of *Escherichia coli* at low ammonium concentration. Journal of Biotechnology 144(3), 175–83 (2009)
132. Madsen, C., Myers, C., Roehner, N., Winstead, C., Zhang, Z.: Utilizing Stochastic Model Checking to Analyze Genetic Circuits. In: IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB). pp. 379–386 (2012)
133. Maler, O., Batt, G.: Approximating continuous systems by timed automata. In: Formal Methods in Systems Biology (FMSB). pp. 77–89. LNCS, Springer (2008)

134. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS/FTRTFT). LNCS, vol. 3253, pp. 152–166. Springer (2004)
135. Maler, O., Nickovic, D., Pnueli, A.: Checking temporal properties of discrete, timed and continuous behaviors. In: Pillars of Computer Science, pp. 475–505. Springer-Verlag, Berlin, Heidelberg (2008)
136. Mateescu, R., Monteiro, P.T., Dumas, E., de Jong, H.: CTRL: Extension of CTL with regular expressions and fairness operators to verify genetic regulatory networks. Theoretical Computer Science 412(26), 2854–2883 (2011)
137. Melham, T., Bard, J., Werner, E., Noble, D.: Conceptual foundations of systems biology. Prog. Biophys. Mol. Biol. (2012)
138. Merrill, D., Garland, M., Grimshaw, A.: Scalable GPU Graph Traversal. In: ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP). pp. 117–128. ACM (2012)
139. Pelánek, R.: Fighting State Space Explosion: Review and Evaluation. In: Formal Methods for Industrial Critical Systems (FMICS). LNCS, vol. 5596, pp. 37–52. Springer (2009)
140. Peled, D.: Ten years of partial order reduction. In: Computer Aided Verification (CAV). pp. 17–28. LNCS, Springer-Verlag (1998)
141. Phillips, A., Cardelli, L.: Efficient, correct simulation of biological processes in the stochastic $\pi$-calculus. In: Computational Methods in Systems Biology (CMSB). pp. 184–199. LNCS, Springer-Verlag (2007)
142. Pnueli, A.: The temporal semantics of concurrent programs. Theoretical Computer Science 13(1), 45 – 60 (1981)
143. Popova-Zeugmann, L., Heiner, M., Koch, I.: Time Petri Nets for Modelling and Analysis of Biochemical Networks. Fundam. Inform. 67(1-3), 149–162 (2005)
144. Priami, C.: Algorithmic systems biology. Commun. ACM 52(5), 80–88 (2009)
145. Regev, A., Silverman, W., Shapiro, E.Y.: Representation and Simulation of Biochemical Processes Using the $\pi$-Calculus Process Algebra. In: Pacific Symposium on Biocomputing. pp. 459–470 (2001)
146. Reif, J.: Depth-first Search is Inherently Sequential. Information Proccesing Letters 20(5), 229–234 (1985)
147. Rizk, A., Batt, G., Fages, F., Soliman, S.: A general computational method for robustness analysis with applications to synthetic gene networks. Bioinformatics 25(12) (2009)
148. Satish, N., Harris, M., Garland, M.: Designing efficient sorting algorithms for manycore gpus. In: IEEE International Parallel & Distributed Processing Symposium (IPDPS). pp. 1–10. IEEE Computer Society (2009)
149. Schaub, M., Henzinger, T., Fisher, J.: Qualitative networks: a symbolic approach to analyze biological signaling networks. BMC Systems Biology 1(1), 4 (2007)
150. Schivo, D.S., Scholma, J., Wanders, B., Urquidi Camacho, R., van der, P.V., Karperien, H., Langerak, R., van de, J.P., Post, J.: Modelling biological pathway dynamics with timed automata. In: IEEE International Conference on Bioinformatics and Bioengineering (ICBB). pp. 447–453. IEEE Computer Society (2012)
151. Schwarick, M., Heiner, M.: CSL model checking of biochemical networks with interval decision diagrams. In: Computational Methods in Systems Biology (CMSB). LNCS/LNBI, vol. 5688, pp. 296–312. Springer (2009)
152. Schwarick, M., Rohr, C., Heiner, M.: MARCIE - Model checking And Reachability analysis done effiCIEntly . In: Quantitative Evaluation of SysTems (QEST 2011). pp. 91–100. IEEE Computer Society (2011)

153. Siebert, H., Bockmayr, A.: Incorporating time delays into the logical analysis of gene regulatory networks. In: Computational Methods in Systems Biology (CMSB), LNCS, vol. 4210, pp. 169–183. Springer Berlin Heidelberg (2006)
154. Singh, A., Hespanha, J.a.P.: Stochastic hybrid systems for studying biochemical processes. Physical and Engineering Sciences 368(1930), 4995–5011 (2010)
155. Stern, U., Dill, D.L.: Parallelizing the mur$\varphi$ verifier. In: Computer Aided Verification (CAV). LNCS, vol. 1254, pp. 256–267. Springer (1997)
156. Stern, U., Dill, D.L.: Using Magnetic Disk Instead of Main Memory in the Mur$\varphi$ Verifier. In: Computer Aided Verification (CAV). pp. 172–183 (1998)
157. Swat, M., Kel, A., Herzel, H.: Bifurcation analysis of the regulatory modules of the mammalian G1/S transition. Bioinformatics 20(10), 1506–1511 (2004)
158. Tarjan, R.: Depth First Search and Linear Graph Algorithms. SIAM Journal on Computing 1(2), 146–160 (1972)
159. Thomas, R.: Regulatory networks seen as asynchronous automata: A logical description. Journal of Theoretical Biology 153(1), 1–23 (1991)
160. Vardi, M.Y., Wolper, P.: An Automata-Theoretic Approach to Automatic Program Verification. In: IEEE Symposium on Logic in Computer Science (LICS). pp. 332–344. IEEE Computer Society Press (1986)
161. W.J.Stewart: Introduction to the Numerical Solution of Markov Chains. Princeton University Press (1995)
162. Yang, E., van Nimwegen, E., Zavolan, M., Rajewsky, N., Schroeder, M., Magnasco, M., Darnell, J.E.: Decay Rates of Human mRNAs: Correlation With Functional Characteristics and Sequence Attributes. Genome Research 13(8), 1863–1872 (2003)
163. Yang, H.T., Ko, M.S.H.: Stochastic modeling for the expression of a gene regulated by competing transcription factors. PLoS ONE 7(3), e32376 (2012)
164. Yovine, S.: Kronos: a verification tool for real-time systems. International Journal on Software Tools for Technology Transfer 1, 123–133 (1997)