On Parameter Synthesis by Parallel Model Checking

J. Barnat, L. Brim, A. Krejčí, A. Streck, D. Šafránek, M. Vejnár and T. Vejpustek
Faculty of Informatics, Masaryk University
Brno, Czech Republic
Email: safranek@fi.muni.cz

Abstract—An important problem in current computational systems biology is to analyse models of biological systems dynamics under parameter uncertainty. This paper presents a novel algorithm for parameter synthesis based on parallel model checking. The algorithm is conceptually universal with respect to the modelling approach employed. We introduce the algorithm, show its scalability, and examine its applicability on several biological models.

Index Terms—biological networks; parallel model checking; dynamical systems; parameter synthesis; systems biology

1 Introduction

Understanding of biological organisms dynamics is a challenging topic methodologically targeted by computational systems biology [1]. The main goal is to reconstruct executable quantitative models that robustly mimic the dynamics of the biochemical machinery underlying living systems. Such models are traditionally built in deterministic framework of population dynamics (given by ordinary differential equations – so-called ODE models [2]). In order to obtain reliable quantitative models, parameters which significantly affect the system dynamics, such as reaction rates or concentration values, need to be specified exactly.

It is a well-known issue that measurement of most quantitative parameters *in vitro*/*in vivo* is hardly possible. In general, the level of detail at which parameter values can be measured is very poor in comparison to the level of detail needed for a quantitative model. An example is measuring protein production rates, which vary with different regulation modes arising during protein transcription. To this end, abstract modelling approaches that focus on qualitative properties of the dynamics are developed. These approaches lift the granularity of model parameter values to qualitative domains, e.g. boolean or multi-valued logic values [3], [4].

Efficient analysis of biological models under parameter uncertainty is an important topic in current computational systems biology and it provides a fundamental cornerstone for synthetic biology [5]. Traditional approaches to identification of parameter values are represented by *parameter estimation* algorithms based on least squares or maximum likelihood methods of fitting the model parameters to experimental data [6]. Computational systems biology provides specifically tuned parameter estimation methods based on optimisation of traditional methods to the specific characteristics of biologically-relevant ODE models [7], [8].

Another commonly used method that helps investigating the role of model parameters is bifurcation analysis [9]–[11] based on identifying variations of model dynamics with respect to certain bifurcations.

In the field of hybrid systems and control theory, the problem of parameter estimation is reformulated as *parameter synthesis problem* [12]. The goal is to find parameter values that ensure satisfaction of a given safety property. The problem has been solved by employing many different approaches, some of which are based on reachability analysis [13], on model checking [14], [15], and on numerical simulations [16].

Automated methods of searching for optimal and robust sets of kinetic parameters w.r.t. required dynamic properties form an important precursor to successful model reconstruction in systems biology. A significant contribution to this field provided by computer science is based on model checking – a technique well-established in the domain of automated verification of software and hardware system designs [17]. The aim of model checking is to automatically show whether the system design (the model) is compliant w.r.t. the given set of required properties (the specification). The specification is stated in a temporal logic and a model checking algorithm is then employed to decide whether the specification is valid for the revised model or not. In the negative case, a counterexample showing the incorrect behaviour of the model is returned. Typical workflow employing model checking is depicted in Fig. 1 (left).

It is worth noting that biological systems are in many aspects similar to complex parallel software and hardware systems. Since parallelism introduces nondeterminism into possible program executions, dynamics of parallel programs becomes hardly predictable.

Such intricacy is inherent in biological systems. With respect to this similarity, the system engineering scenario can be slightly modified and adapted for biological systems, as suggested, e.g., in [18].

Following the analogy mentioned above, the computer-aided workflow for reconstruction of biological models can be reset as depicted in Fig. 1 (right). After the model structure is reconstructed in the form of a biological network, the problem of finding appropriate values of kinetic parameters typically arises. In this stage, wet-lab measurements capturing observations of the organism dynamics can be collected and represented in the form of temporal specification. Some tools automatizing this process already exist, e.g. [19], rapid development in this domain is expected in near future. The general advantage of temporal specification is its ability to focus on certain qualitative aspects of observed behaviour [20] (e.g., temporal ordering of events qualitatively characterising important moments in the system dynamics). By executing model checking w.r.t. the specification over every possible parameter settings of the model, any unacceptable parameter set can be automatically rejected. Since the final goal is to find all admissible parameter settings w.r.t. the given specification, this procedure is reduced to general *parameter synthesis* problem as understood in hybrid systems, provided that the reference property is not limited to safety properties only, but extended to any general temporal property. Once a robust model is available, the original system engineering scenario can be revisited and directly employed for the synthesis of a new organism tuned w.r.t. the given requirements [14].

When the specification in the scenario mentioned above is entirely synthesised from experimental observations, the proposed approach basically degrades to traditional parameter estimation methods [7]. In order to maximise the gain from the computer-scientific framework, we need to utilise the process of property specification. In particular, temporal properties w.r.t. which parameters are synthesised can be viewed as global properties independent of particular setting of initial conditions (initial values of the state variables). An example of such property is a presence of multiple steady states. The global view provides biologists a tool which, for a given model and a given property, computes the maximal set of parameter values for which the model entirely fulfils the property. Such an approach is complementary to traditional approaches based on a numerical simulation [21], [22].

It is important to note that there are several levels of complexity that significantly affect the tractability of parameter synthesis for biological models. First, the procedure requires consideration of all possible settings of parameters – points in the parameter space. The size of parameter space grows exponentially with the number of unknown parameters. Second, during each model checking phase – analysis of the model with particular parameter settings – the dynamics of the model has to be explored. More precisely, the state space of the model, which grows exponentially with the number of state variables, has to be traversed in each model checking phase. It is also worth noting that tractability of the method presumes the finiteness of parameter space and state space. In case of abstract discrete models such as boolean networks, this is ensured directly by definition, while in case of continuous ODE models, a suitable finite discrete abstraction of the state space is necessary. There are several abstraction methods defined for several classes of ODE models that fit the format of many biological models [4], [23]. By defining such an abstraction on the state space, the suitable finite discretization of the parameter space is usually achieved directly [14], [24].

Given the complexity of the problem and the need for comprehensive large-scale models, there is a natural call for development of techniques prepared to perform efficiently on high-performance computing platforms [25]. Moreover, individual levels of complexity can be significantly reduced in the average case. The parameter space size is typically treated by means of symbolic parameter space representation [14], [24], which is highly dependent on the modelling approach employed. E.g., techniques defined in [24] use unique properties of piecewise affine ODE models [4], whereas techniques of [14] are developed for multi-affine ODE models. In general, symbolic representation of the parameter space may in certain cases significantly improve the efficiency of the parameter synthesis but at the price of loosing the universality. The complexity caused by the state space size can be reduced by either symbolic or enumerative parallel techniques. The achieved efficiency is again highly dependent on the modelling approach, character of models, and the properties considered. In the case of biological models, symbolic techniques were successfully employed for abstract logical (qualitative) models [24] whereas enumerative parallel techniques have proved to be fruitful for quantitative models [26].

1.1 Our Contribution

In this paper, we contribute to the domain of computational biology by developing a new scalable and universal algorithm for parameter synthesis w.r.t. linear temporal specification.

Scalability: Algorithm is designed for parallel architectures, its power is demonstrated in terms of a multi-core data-parallel implementation. The computation scales with the increase in number of cores and thus allows handling of large parameter spaces in reasonable computation time.

Universality: The algorithm is universal in terms that it works on any finite state transition system with parameterisations that affect directly the transition relation. In particular, it implies independence of the modelling approach employed.

Our approach is based on enumerative model checking techniques and implicit representation of state space

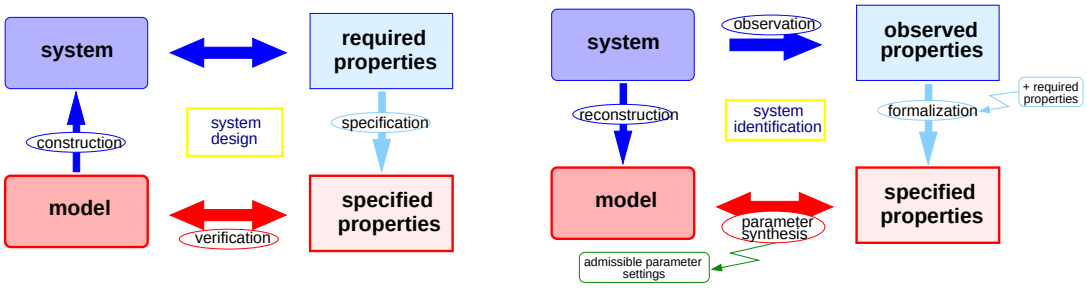


Fig. 1. (left) In systems engineering supported by formal methods, the system is constructed in a three-phase process: stating of the system requirements (the specification), developing a model (design) for which model checking is used to ensure that the design is correct w.r.t. the specification, and finally, the system is constructed (programmed) according to the correct model. (right) In systems biology based on formal methods, the complex biological system is examined by several wet-lab experiments the results of which can be captured in the form of qualitative (or quantitative) temporal properties. Parameterized model is reconstructed from the structure of the biological network. Finally, the model checking method is employed for synthesis of reliable parameter values that fit the in vitro/in vivo observations and additional requirements.

and parameter space. The central idea of our algorithm is considering the state space as a single object which is universal with respect to parameter valuations. If we denote each parameter valuation by a distinct color and assume that the respective (valuation-specific) state space has all its transitions marked by this color, we can construct a global state space as a union of all the valuation-specific state spaces. Since different parameters setting 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 synthesis problem is based on analysis of infinite mono-colored paths in a graph with multi-colored edges. Since in many parameterisations small perturbations in parameter values lead to small locally distributed variations in the transition relation, the respective mono-coloured graphs can exhibit significant similarity. For parameterisations amenable to such property our algorithm achieves good efficiency in the average case.

The presentation proceeds as follows. In Section 2 we briefly recall two different modelling approaches on which we further examine our algorithm. We consider a purely discrete (qualitative) approach based on boolean networks and a quantitative approach based on piecewise multi-affine abstraction of continuous ODE models. Further, we formulate the parameter synthesis problem and introduce the central notion of a *parameterised Kripke structure* (PKS) that unifies a family of state spaces sharing the same set of states but differing in some transitions. In Section 3 we present the universal algorithm for LTL model checking of PKSs. In Section 4, we provide several case studies that show the applicability of the proposed approach to real biological models. Finally, the parallel shared-memory implementation of our algorithm is described and evaluated in Section 5.

1.2 Related Work

The very first algorithm for parameter synthesis of ODE models employing model checking over discrete abstractions has been introduced by Batt et al. in [14], [15]. The work defines a general technique for finite partitioning of the parameter space into classes of behaviourally equivalent valuations. We employ this technique for quantitative models in a restricted form – we consider rectangular partitioning provided that only linearly independent parameters can be considered unknown. This restriction has been employed purely due to implementation efficiency – we avoided the use of computationally expensive algorithms [14], [15] which would be otherwise necessary for more complicated parameter spaces.

The algorithm in [14] is sequential and relies on execution of two model checking procedures per each class of valuations. In the average case, the number of analysed parameter classes can be reduced by suitable BDD representation of the parameter space. On the contrary, our algorithm is based on a principally different idea. Enumerative LTL model checking procedure (reduced to detection of accepting cycles in the model-property product automaton) is employed directly on a graph that compactly represents the dynamics of all valuations. The computational effort can be significantly reduced in the average case due to small variations in subgraphs corresponding to different valuations. Moreover, we take the advantage of the choice of enumerative model checking and develop a parallel algorithm that accelerates the computation with the increasing number of CPUs.

The traditional approach based on numerical simulation is employed in [21]. Numerical simulation produces approximation of trajectories generated with respect to a given starting point in the model phase space – the initial condition. In contrast to that work, the abstraction procedure employed in this paper allows model checking of global dynamic properties – the initial condition is generalised to any compact subset of the phase space.

Batt et al. [24] targets the piece-wise affine framework adapted to regulatory networks. The notion of uncertainty is lifted to a higher level of abstraction provided that qualitative (symbolic) valuation is considered. In comparison, our approach is universal in the sense of working on both quantitative and qualitative parameters. We do not employ symbolic techniques to compact the parameter space representation but rather use an explicit representation of parameters in terms of parameterised transition relation. This is a crucial aspect which allows us to distribute the computation among multiple threads. Such a distinction is similar to difference between symbolic and enumerative model checking algorithms. In particular, symbolic representation is one of the ways of fighting the state explosion. We employ another way based on parallel enumerative algorithms. The advantage we obtain is the scalability.

Aforementioned approaches come from a large group of methods for hybrid systems analysis. Owing to the non-linear character of studied systems, these methods are usually applicable only to certain subclasses of biological models. The approach of [27] supports arbitrary non-linear ODE models. It is based on a hybrid approach built on sensitivity analysis. Approaches [12], [13], [15] inherently rely on reachability analysis in hybrid systems and are sufficiently applicable only to piece-wise affine or multi-affine systems. These algorithms technically rely on computing reachability for parameterised systems by means of polyhedral operations. This restricts the parameter synthesis procedure to safety properties only (liveness properties, e.g. oscillations, for which any contradicting counterexample is an infinite path requiring monotonous time progress, can be considered only in special cases [14]). On the contrary, our approach is purely a graph-based technique that works directly at the level of discrete state space. Parameter synthesis is realized in terms of general model checking over Kripke structures with parameterised transition relation. This setting allows application of our algorithm to hybrid and continuous models, provided that a suitable finite discrete abstraction of continuous parts is possible. Again, for this domain of quantitative models, the applicability of our algorithm is reduced to piece-wise affine and multi-affine systems, for which such abstractions exist [23].

In case of discrete (boolean) modelling approaches, usage of model checking for guiding synthesis of admissible parameters has been originally targeted in [18] with a tool support described in [28]. The approach uses the naive algorithm of separate model checking for every parameter valuation of the model. Here we revisit the same parameterisation, but with a computational approach that can significantly reduce the execution time. Moreover, our algorithm adds scalability when executed on a multi-core platform.

Finally, it is worth noting that a notion similar to parameterised Kripke structure has been also mentioned in the context of hardware verification, in particular, to capture delay parameters in logical circuits [29], [30]. Our notion is conceptually different, since we define parameterised Kripke structure as a general structure that encapsulates the family of Kripke structures for each possible valuation of the parameters. In our case, parameterisation targets the entire transition relation.

2 Methods and Definitions

2.1 Qualitative Modeling Approach

As a significant representative of qualitative dynamics modeling approaches in biology we consider Thomas' extension of boolean networks [3], designed for discrete modeling of genetic regulatory networks (GRNs). In particular, we employ the discrete framework according to GINsim tool [31]. 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 *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 *target expression level* of g_i when regulated by all genes in R, $0 \leq K_{i,R} \leq \rho(g_i)$.

In Fig. 2 (left) there is depicted a simple model representing interactions between two genes. Since maximum expression levels of both genes are set to 1, the model is purely boolean. Gene A, when expressed above the activation threshold, positively regulates the expression of B ($K_{B,\{A\}} = 1$). Regulation type as well as the activation threshold are depicted in the label of the respective edge. B is not expressed when A is inactive ($K_{B,\emptyset} = 0$). Gene A, when expressed above threshold 1, blocks its own expression ($K_{A,\{A\}} = 0$). Otherwise it is expressed ($K_{A,\emptyset} = 1$).

Once the regulatory logic is set, the semantics in terms of the state-transition system 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 $S_0 \subseteq S$ being 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 transition $s \to s'$ according to the following rules:

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

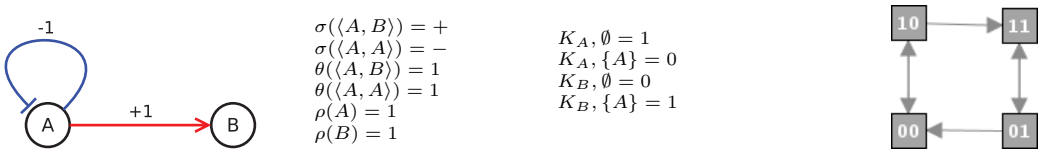


Fig. 2. (left) An example of a simple GRN with the given regulatory logic. (right) The state space of the network. The first and second component of a state denotes the current expression level l_A and l_B, respectively.

- 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 [3], [32]. Example of the transition system is given in Fig. 2 (right).

For model parameterization we consider parameters $K_{i,r}$ denoting the target regulatory levels of the genes. As it has been discussed in [18], finding the appropriate values of these parameters is a critical task. Formally, we define the *parameterized boolean model* $\mathcal{B}_\chi$ as a tuple $\langle \chi, G, \sigma, \theta, \rho, L\rangle$ where $\chi \subseteq L$ is a set of unknown parameters. Denote by $I(\chi)$ the set of parameter indices, $I(\chi) = \{\langle i, R\rangle \mid K_{i,R} \in \chi\}$. The *parameter space* for $\mathcal{B}_\chi$ (the set of valuations of χ) is defined as $\mathcal{P}_\chi = \prod_{\langle i,R\rangle \in I(\chi)} \{0, ..., \rho(g_i)\}$. Note that once the target level $K_{i,R}$ is supposed to be unknown then all its possible values have the range of all g_i expression levels.

2.2 Quantitative Modeling Approach

By employing the law of mass action [2] that describes quantitatively the population (deterministic) view of the system behaviour, dynamics of any biochemical reaction network can be quantitatively modelled in terms of a system of coupled non-linear ordinary differential equations (ODE system). The regulatory interactions are usually modelled by means of rational polynomial functions (Hill kinetics) appearing in right-hand side of equations [33]. State variables are evaluated as species concentration in continuous domain provided that there is one differential equation per each variable describing the rate of change of the state variable in an infinitesimal interval of time.

Regarding the need for a finite discrete abstraction of the continuous state space due to computational reasons stated in Section 1, we consider a class of ODE systems for which such an abstraction exists [23]. In particular, we consider piece-wise multi-affine ODE systems [14]. Under this class, it is possible to express biochemical systems with chemical reactions in which the stoichiometric coefficients of all reactants are at most 1. Moreover, the (non-linear) rational polynomial functions driving the regulatory kinetics can be expressed under this class by means of piece-wise linear approximation. More specifically, regulatory interactions are modelled as a suitable composition of piece-wise linear *ramp-functions* defined in the following way:

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

where $\theta_i, \theta_i' \in \mathbb{R}^+$, $\theta_i < \theta_i' \le max_i$. The negative ramp-function is defined $r^-(x_i, \theta_i, \theta_i') = 1 - r^+(x_i, \theta_i, \theta_i')$.

The *piece-wise multi-affine ODE model (PMA model)* $\mathcal{M}$ is given by an ODE system in the form $\dot{x} = f(x)$ where $x = (x_1, ..., x_n)$ is a vector of *state variables* and $f = (f_1, ..., f_n) : \mathbb{R}^n \to \mathbb{R}^n$ is a vector of piece-wise multi-affine functions, and a *set of thresholds* $\theta_j^i \in \mathbb{R}$ satisfying $min_i = \theta_1^i < \theta_2^i < \ldots < \theta_{\zeta_i}^i = max_i$ for each variable x_i. The number of thresholds on x_i is denoted by ζ_i, $\zeta_i \ge 2$. By Ω we denote the *partition* of $\mathcal{M}$, $\Omega = \prod_{i=1}^n \{1, \ldots, \zeta_i - 1\}$. A function $g : \mathbb{R}^n \to \mathbb{R}^+$ is piece-wise multi-affine if it is multi-affine on each n-dimensional interval $(\theta_{j_1}^1, \theta_{j_1+1}^1) \times ... \times (\theta_{j_n}^n, \theta_{j_n+1}^n)$ where $(j_1, ..., j_n) \in \Omega$ and $\forall i, 1 \le i \le n.\, j_i < max_i$.

We consider n-dimensional PMA models in which $f = (f_1, ..., f_n)$ has the form

$$\forall i \in \{1, ..., n\}.\, f_i(x) = \sum_{j \in I^+} \kappa_i^j \varrho_i^j(x) - \sum_{j \in I^-} \gamma_i^j \varrho_i^j(x)$$

where I^+ and I^- are finite index sets such that $I^+ \cap I^- = \emptyset$, $0 \ne \kappa_i^j, \gamma_i^j \in \mathbb{R}^+$ are kinetic parameters, and ϱ_i^j is a kinetic function.

Kinetic function $\varrho(x)$ is defined inductively as follows:

- $\varrho(x) = 1$, $\varrho(x) = x_k$ or $\varrho(x) = r^*(x_k, \theta_l^k, \theta_{l+1}^k)$ are kinetic functions for any $k \in \{1, ..., n\}$, $* \in \{+, -\}$, $l \in \{1, ..., \zeta_k - 1\}$.
- If ϱ_1, ϱ_2 are kinetic functions such that $dep(\varrho_1) \cap dep(\varrho_2) = \emptyset$ then $\varrho = 1 - \varrho_1$ and $\varrho = \varrho_1 \varrho_2$ are kinetic functions, where $dep(\varrho)$ denotes the set of variables on which ϱ depends (this ensures multi-affinity).

An example of a PMA model is given in Fig. 3 (left).

In [14], [23] it is shown that the partition Ω constitutes a finite discrete abstraction of the continuous state space. Rules that introduce transitions between neighbouring rectangles are also stated there. We define the *rectangular abstraction transition system* for the model $\mathcal{M}$, written $RATS(\mathcal{M})$, as a triple (S, T, S_0) where $S \subseteq \Omega$ is the *set of states*, $S_0 \subseteq S$ the *set of initial states*, and $T \subseteq S \times S$ the *transition relation*. Precise definition of T relevant to our setting is given in [34]. An example of RATS is shown in Fig. 3 (right), the green arrows represent the transitions between rectangles. The abstraction has an overapproximative character w.r.t. the continuous ODE

$$\frac{d[A]}{dt} = \kappa_a r^-(A,2,3) - \gamma_a[A]$$
$$\frac{d[B]}{dt} = \kappa_b r^+(A,2,3) - \gamma_b[B]$$
$$\kappa_a = 1, \gamma_a = 0.1$$
$$\kappa_b = 1.2, \gamma_b = 0.2$$

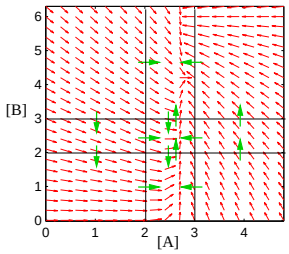


Fig. 3. Example of a PMA model which is a refinement of the boolean model from Fig. 2. The vector field and the corresponding state transition system (right) were obtained under the given settings of kinetic parameters (left).

model: Every trajectory in $\mathcal{M}$ is represented by some path in $RATS(\mathcal{M})$ with the only exception of trajectories the set of which is of measure zero [14].

We consider parameterisation of PMA models which is determined by a set χ of (unknown) kinetic parameters such that in each equation of the underlying ODE system there is at most one parameter considered unknown. An unknown parameter in ith equation is denoted μ_i and $I(\chi) = \{i \mid \mu_i \in \chi\}$ denotes the ordered set of indices of unknown parameters. For each $i \in I(\chi)$ we consider the minimal and maximal parameter value min_{μ_i} and max_{μ_i} and we define the *parameter space* (the set of valuations of χ) as $\mathcal{P}_\chi = \prod_{i \in I(\chi)} [min_{\mu_i}, max_{\mu_i}]$.

The *parameterised PMA model* $\mathcal{M}_\chi$ is defined for any χ where for any $i \in I(\chi)$ the respective multi-affine function $f_i(x)$ of $\mathcal{M}$ is parameterised to $f_i(x, \mu_i)$. For any valuation $p \in \mathcal{P}_\chi$ we use $RATS(\mathcal{M}_p)$ to denote the respective state transition system. Note that the multi-dimensional function f is parameterised to $f(x, \mu)$ where $\mu = \prod_{i \in I(\chi)} \mu_i$. From the format of each f_i it directly follows that f_i is piece-wise affine in μ_i and thus $f(x, \mu)$ is piece-wise affine in μ. This property is crucial, since it assures that the parameter space can be always partitioned into a finite number of equivalence classes where for any two valuations $p, q \in \mathcal{P}_\chi$ the equivalence $p \sim q$ means that the respective state transition systems collide, i.e., $RATS(\mathcal{M}_p) = RATS(\mathcal{M}_q)$.

2.3 Model Checking

In general, model checking [17] refers to the problem of automatically testing whether a simplified model M of a system meets a given specification ϕ, written $M \models \phi$. Specification is stated by means of a temporal logic formulae. Models are considered as finite state automata (or generators) on infinite strings – the so-called Kripke structures (KS).

When considering a biochemical system, traditional temporal logics provide a formalism which can formally express qualitative properties of reaction kinetics such as all kinds of stability (including non-hyperbolic equilibria as well as multi-stability), oscillations, temporal ordering of certain concentration levels, causality, etc. [20]. We employ linear temporal logic (LTL) interpreted on infinite paths generated by finite state transition systems obtained for the model w.r.t. Section 2.1 or Section 2.2 (formally considered as Kripke structures which additionally require only to add self-transitions for states that form sink nodes of the state transition graph).

A (non-deterministic) *Büchi automaton* (BA) is a tuple $\mathcal{A} = (\Sigma, S, s_0, \delta, F)$, where Σ is the input alphabet, S is a finite non-empty set of states, $s_0 \in S$ is a distinguished initial state, $\delta \subseteq S \times \Sigma \times S$ is a transition relation, and $F \subseteq S$ is the set of accepting states.

Given any LTL formula φ over the set of atomic propositions AP, there can be constructed a Büchi automaton $\mathcal{A}_\varphi$ over the alphabet 2^{AP} accepting exactly the words satisfying φ. The methods for this are common and many tools exist [17].

The traditional approach to model checking [17] a Kripke structure $\mathcal{K}$ against an LTL formula φ is to

1) construct a BA over 2^{AP} such that it accepts all words satisfying $\neg\varphi$,
2) convert $\mathcal{K}$ to a Büchi automaton
3) compute the synchronous product of the two automata, and
4) decide whether the resulting automaton accepts an empty language.

In the negative case, a word which belongs to the non-empty language generated by the product automaton is returned as a counterexample.

2.4 Parameter Synthesis Problem

We start by introducing the notion of a parameterized Kripke structure that encapsulates a family of Kripke structures built over the same model but with different valuations of belonging parameters. This notion allows us to formulate the parameter synthesis problem in the framework of ω-regular languages and automata-based model checking.

We define *parameterized Kripke structure* (PKS) to be a tuple $\mathcal{K} = (\mathcal{P}, S, S_0, \rightarrow, L, F)$, where $\mathcal{P}$ denotes the finite *set of parameter values*, i.e., all the possible valuations of the parameters, S the finite set of states, $S_0 \subseteq S$ the set of initial states. $L : S \rightarrow 2^{\mathrm{AP}}$ is the labeling of states by atomic propositions, $\rightarrow \subseteq S \times \mathcal{P} \times S$ is a transition relation labeled by parameter valuations (not required to be total) and $F \subseteq S$ is a set of accepting states. We write $s \xrightarrow{p} s'$ instead of $(s, p, s') \in \rightarrow$. We write $s \rightarrow s'$ if $s \xrightarrow{p} s'$ for some $p \in \mathcal{P}$. Fixing a valuation $p \in \mathcal{P}$ reduces the parametrized Kripke structure $\mathcal{K}$ to the concrete (non-parameterized) Kripke structure $\mathcal{K}(p) = (S, S_0, \xrightarrow{p}, L)$.

We use the notation $\mathcal{P}(s, s') = \{p \in \mathcal{P} \mid s \xrightarrow{p} s'\}$ where $\mathcal{P}(s, s')$ is a set of parameter values for which a transition from $s \in S$ to $s' \in S$ is allowed.

Preliminary to model checking, a formal step is required, in particular, a PKS has to be transformed into a BA. The difference between the classical and our

approach is that we define the alphabet as a combination of atomic propositions and parameters that allow the transition from one state to another. Formally: PKS $\mathcal{K} = (\mathcal{P}, S, S_0, \rightarrow, L, F)$ can be transformed into BA $\mathcal{A} = (\mathcal{P} \times \Sigma, S \cup \{\imath\}, \imath, \delta, F)$ where $\mathcal{P} \times \Sigma$ with $\Sigma = 2^{\mathrm{AP}}$ is an alphabet, $S \cup \{\imath\}$ is the set of states, $\imath$ is a new initial state, δ is a transition function and $F \subseteq S \cup \{\imath\}$ are accepting states. The transition $(s, (p, \alpha), s') \in \delta$ for $s, s' \in S$, $\alpha \in \Sigma$ if and only if $s \xrightarrow{p} s' \in \rightarrow$ and $\alpha \in L(s')$. We also have to provide a new initial state to comply the specification formally imposed on Büchi automata. In order to retain the behaviour of PKS, we have to extend δ with transitions $(\imath, (p, \alpha), s)$ for all $s \in S_0, p \in \mathcal{P}$ and $\alpha \in \Sigma$ such that $\alpha \in L(s)$. We extend the notation $s \xrightarrow{p} s'$ homomorphically to the transition relation of the product automaton, in particular, $s \xrightarrow{p} s'$ denotes the fact that there exist $p \in \mathcal{P}$ and $\alpha \in \Sigma$ such that $(s, (p, \alpha), s') \in \delta$.

Now we define the central notion of LTL model checking, in particular, the synchronous product of two BAs. We use the same procedure as stated in [17] but augmented with parameters. The *synchronous product of BA* $\mathcal{A}_1 = (\mathcal{P} \times \Sigma, S_1, \imath_1, \delta_1, F_1)$ *and BA* $\mathcal{A}_2 = (\mathcal{P} \times \Sigma, S_2, \imath_2, \delta_2, F_2)$ is a BA $\mathcal{A}_1 \cap \mathcal{A}_2 = (\mathcal{P} \times \Sigma, S', \imath', \delta', F')$ where

- $S' = S_1 \times S_2 \times \{0, 1, 2\}$,
- $\imath' = \imath_1 \times \imath_2 \times \{0\}$,
- $F' = F_1 \times F_2 \times \{2\}$,
- $((s, r, x), (p, \alpha), (s', r', y)) \in \delta$ if and only if $(s, \alpha, s') \in \delta_1$ and $(r, (p, \alpha), r') \in \delta_2$.The rules for the third component are:
  - if $x = 0$ and $s' \in F_1$ then $y = 1$,
  - if $x = 1$ and $r' \in F_2$ then $y = 2$,
  - if $x = 2$ then $y = 0$,
  - otherwise, $y = x$.

We say that the product automaton $\mathcal{A}_1 \cap \mathcal{A}_2$ accepts the infinite word w if and only if there is a symbol $\xi \in F'$ that appears infinitely often in w, and $\xi \in F'$. Hence the product automaton $\mathcal{K} \times \mathcal{A}_{\neg\varphi}$ accepts exactly infinite words that violate the formula φ. The parameter synthesis problem w.r.t. specification φ can be now defined as finding the maximal set $P \subseteq \mathcal{P}$ such that for every $p \in P$ the set of infinite words generated by $\mathcal{K} \times \mathcal{A}_{\neg\varphi}$ is empty. From the perspective of the state space of the $\mathcal{A}_1 \cap \mathcal{A}_2$, the automaton accepts if there is a cycle - causing automaton to be able to read the infinite word - that contains a state whose first component is from F_1 and a state with the second component from F_2. Therefore, when the automaton reaches the state with a component from F_1 or F_2, the decision whether to accept or not requires knowledge about preceding states. This knowledge is stored in the third component of the state in the manner described above.

Formally, let $\mathcal{K} = (\mathcal{P}, S, S_0, \rightarrow, L, F)$ be a PKS. For a parameter valuation $p \in \mathcal{P}$ we denote by w_p words satisfying $w_p \in L(\mathcal{K}(p))$. Further let φ be an LTL formula. We define the *parameter synthesis problem* as a problem of finding maximal set of parameter valuations $P \subseteq \mathcal{P}$ such that all words in $\{w_p \in L(\mathcal{K}(p)) \mid p \in P\}$ are accepted by $\mathcal{K} \cap \mathcal{A}_{\neg\varphi}$, formally written $\{w_p \in L(\mathcal{K}(p)) \mid p \in P\} \subseteq L(\mathcal{K} \cap \mathcal{A}_{\neg\varphi})$.

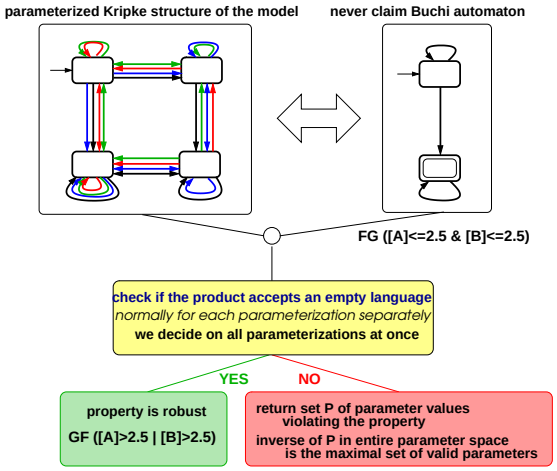


Fig. 4. Intuition behind the paramater synthesis method.

In Fig. 4 there is illustrated the basic idea of solving the parameter synthesis problem by model checking. Parameterized Kripke structure can be considered as an automaton with coloured edges where each colour corresponds to a certain parameter valuation. We expect transitions to be largely shared among individual colors. This allows us to accelerate the property decision.

Since the parameter space cardinality increases exponentially with the number of unknown parameters, our primary goal is to reduce this complexity for the average case. To this end, we limit ourselves to the study of a subclass of models satisfying the following requirements:

1) The model state space and parameter space must be representable in terms of a PKS.
2) The transition-enabling parameter valuation sets $\mathcal{P}(s, s')$ can be computed efficiently from the knowledge of the model and the state s. In particular, we suppose there exists an implicit representation of these sets.
3) A small change in value of a single parameter causes only a local change in the transition relation. This implies that sets $\mathcal{P}(s, s')$ represent significant portions of the parameter space (transition relations of respective non-parameterized Kripke structures are similar).

Note that biological models with parameterizations described in Section 2.1 and Section 2.2 exhibit such properties. Our goal is to provide an algorithm which would in practice perform parameter synthesis on suitable models reasonably fast and which would be effectively parallelizable in order to achieve scalability.

3 Parameter Synthesis Algorithm

In this section we introduce the algorithm that efficiently solves the parameter synthesis problem as defined in the previous section.

Throughout this section we assume that $\mathcal{A} = (\mathcal{P} \times \Sigma, S, \imath, \delta, F)$ is a BA defined for some parameter space $\mathcal{P}$. Suppose that $\mathcal{A}$ is the synchronous product automaton constructed according to Section 2.4.

For any $p \in \mathcal{P}$ we define a p-run as an accepting execution of $\mathcal{A}$ over a word $w_p \in L(\mathcal{A})$. Note that such an execution must be of the form $\imath \xrightarrow{p}{}^{*} \gamma \xrightarrow{p}{}^{+} \gamma$ for some $\gamma \in S$. For any $\alpha, \beta \in S$ we define a p-path from α to β as the execution of $\mathcal{A}$ starting in α and proceeding to β in finite number of steps, $\alpha \xrightarrow{p}{}^{*} \beta$. If a p-path begins and ends in the same state, i.e., $\alpha \xrightarrow{p}{}^{+} \alpha$, we use the term p-cycle instead. To denote p-runs, p-paths or p-cycles for all $p \in P = \{p_1, ..., p_n\} \subseteq \mathcal{P}$ we use the notation P-run, P-cycle and P-path, respectively.

Automaton $\mathcal{A}$ generates a p-run if and only if there exists a state $\gamma \in F$ such that $\imath \xrightarrow{p}{}^{*} \gamma \xrightarrow{p}{}^{+} \gamma$.

We split the parameter synthesis problem into two subtasks:

1) For each $\gamma \in F$ compute the maximal set of parameter values $P \subseteq \mathcal{P}$ such that for all $p \in P$ there is an execution sequence $\imath \xrightarrow{p}{}^{*} \gamma$.
2) For each $\gamma \in F$ and the corresponding parameters $p \in P \subseteq \mathcal{P}$, determine if there is a p-cycle through γ of length at least 1.

We need to compute sets of parameter values (subsets of $\mathcal{P}$, each intuitively represented as a distinct *color*) under which an accepting execution trace is enabled. To this end, colors are accumulated in states provided that each state is assigned all colors under which it is reached. We define *coloring* – a mapping of parameter value sets to states – as a function $c\colon S \to 2^{\mathcal{P}}$. Then $C = \{c(s) \mid s \in S\}$ denotes the set of all colorings, the so-called *pallet*.

To detect a P-cycle through the state $\gamma \in F$, we compute the pallet $\mathrm{Succ}(\gamma, P)$ which assigns to each state $\alpha \in S$ the maximal set of parameter valuations $P' \subseteq P$ such that $\gamma \xrightarrow{p}{}^{+} \alpha$ for all $p \in P'$. Formally, we define $\mathrm{Succ}(\gamma, P)(\alpha) = \{p \in P \mid \gamma \xrightarrow{p}{}^{+} \alpha\}$. The set of valuations $\mathrm{Succ}(\gamma, P)(\gamma)$ includes exactly all parameter valuations p such that there is a non-empty p-cycle through γ.

Algorithm 1 implements the ideas described above. The algorithm iteratively accumulates the parameter value set P and can terminate as soon as $P = \mathcal{P}$, which allows for a fast discovery of runs violating a given property. It is also worth noting that once the existence of a p-run in $\mathcal{A}$ is detected for some $p \in \mathcal{P}$, it is not necessary to search for p-cycles any longer.

Next we give an algorithm for computing $\mathrm{Succ}(s, P)$ for arbitrary $s \in S$, $P \subseteq \mathcal{P}$. Recall the assumption that the system changes slightly with respect to changes in parameter values, from which we conclude that any particular path will likely be shared by several parameter valuations. Our goal is to take advantage of this and detect paths for all valuations at once.

Algorithm 1 Parameter scan

Require: $\mathcal{A} = (\mathcal{P} \times \Sigma, S, \imath, F, \delta)$
Ensure: $p \in P$ iff $\imath \xrightarrow{p}{}^{*} \gamma \xrightarrow{p}{}^{+} \gamma$ for some $\gamma \in F$
1: $P \leftarrow \emptyset$
2: $C \leftarrow \mathrm{Succ}(\imath, \mathcal{P})$
3: **for all** $\gamma \in F$, $c(\gamma) \setminus P \neq \emptyset$ **do**
4: $\quad P \leftarrow P \cup \mathrm{Succ}(\gamma, c(\gamma) \setminus P)(\gamma)$

Algorithm 2 computes $\mathrm{Succ}(s, P)$. It starts with an empty coloring and incrementally updates it. An update is a tuple (α, P), meaning that the set of parameter values $P \subseteq \mathcal{P}$ should be added to the coloring for the state $\alpha \in S$. The set of pending updates is stored in Q. The algorithm terminates as soon as there are no more pending updates.

By $Q(\alpha) = \bigcup\{P \subseteq \mathcal{P} \mid (\alpha, P) \in Q\}$ we denote the set of parameter values that are currently scheduled for addition to $c(\alpha)$. To prevent Q from containing multiple updates for the same state, we use the merge operation $Q \oplus Q'$ defined as $Q \oplus Q' = \{(\alpha, P) \mid P = Q(\alpha) \cup Q'(\alpha) \wedge P \neq \emptyset\}$ to update Q (line 11). To improve readability, the algorithm employs two queues – Q_{new} and Q_{old}. However, for implementation efficiency, a single queue may be used.

Algorithm 2 Compute $\mathrm{Succ}(s, P)$ over the BA $\mathcal{A}$

Require: $\mathcal{A} = (\mathcal{P} \times \Sigma, S, \imath, F, \delta)$; $P \subseteq \mathcal{P}$; $s \in S$
Ensure: $\forall \alpha \in S : c(\alpha) = \mathrm{Succ}(s, P)(\alpha)$
1: **for all** $\alpha \in S$ **do**
2: $\quad c(\alpha) \leftarrow \emptyset$
3: $Q_{new} \leftarrow (s, P)$
4: **while** $Q_{new} \neq \emptyset$ **do**
5: $\quad Q_{old} \leftarrow Q_{new}$
6: $\quad Q_{new} \leftarrow \emptyset$
7: $\quad$ **while** $Q_{old} \neq \emptyset$ **do**
8: $\quad\quad$ remove (α, P) from Q_{old}
9: $\quad\quad$ **if** $P \not\subseteq c(\alpha)$ **then**
10: $\quad\quad\quad c(\alpha) \leftarrow c(\alpha) \cup P$
11: $\quad\quad\quad Q_{new} \leftarrow Q_{new} \oplus \{(\beta, P \cap \mathcal{P}(\alpha, \beta)) \mid \alpha \to \beta, \beta \in S\}$

In the remaining part we gradually argument the algorithm correctness.

Lemma 1: After k iterations of the outer cycle (breath-first search levels), Q_{new} contains updates provided that $Q_{new}(\alpha)$ includes parameter values that enable a path of length (at most) k from the source state to α.

Proof: At the beginning, Q_{new} only contains an update for s. If after k iterations Q_{new} contains updates coming from paths of length k, in the $(k+1)$th iteration, those updates are copied into Q_{old}. The updates are applied and the algorithm tests whether a transition to some succeeding state is enabled under the parameter values considered in each applied update. If so, the respective parameter values are posted into Q_{new} which in the end contains all updates coming from paths of

length $k+1$. □

Lemma 2: The body of the inner cycle is executed at most $|S|^2$ times.

Proof: According to Lemma 1 we know that after $|S|$ iterations of the outer cycle, each state obtained a coloring coming from paths of length at most $|S|$. Since multiple traversals through a given state cannot introduce any new parameter values, there cannot be any update coming from a path longer than $|S|$. Hence, the cycle must be finished. During each iteration of outer cycle, the inner cycle is executed at most $|Q_{old}|$ times. Because updates of individual states are merged, we know that $|Q_{old}| \leq |S|$. Therefore combination of both cycles causes the body to be executed at most $|S|^2$ times. □

Theorem 1: Algorithm 2 computes the specified output in finite time.

Proof: In at most $|S|^2$ steps, every state is colored with all parameter valuations that enabled a path to it from the source state, which is the required output. □

Theorem 2: Algorithm 1 correctly solves tasks 1 and 2 in finite time.

Proof: The iteration is necessarily finished after detecting existence of a cycle for all states in F. Note that $|F|$ is finite. Now it can be seen that the algorithm correctly fulfils its specification. □

Theorem 3: Worst case complexity of the Algorithm 1 belongs to $\mathcal{O}(|F||S|^2\zeta)$ where ζ is the complexity of coloring manipulation plus $\oplus$ operation (line 11).

Proof: The Succ operation is executed $|F+1|$ times in Algorithm 1 and according to Lemma 2, $\oplus$ operation and coloring c are computed at most $|S|^2$ times while computing Succ. □

Both the coloring manipulation and the $\oplus$ operation depend on the model type and number of parameters. Therefore we only capture it by means of an external constant. Since in many cases close parameter values will enable same paths, the expected-case complexity will be significantly lower than the worst case.

4 Case Study

In this section we provide two case studies aimed primarily at demonstrating universality of the approach presented in this paper. In particular, we examine the algorithm on real cases employing the qualitative and the quantitative modeling framework, as introduced in Section 2. As a source, we consider a qualitative and a quantitative model of a single genetic regulatory network. Additionally, we give a case study that demonstrates, by means of a larger biological model, the power of the parallel implementation. The latter is realized in the framework of quantitative modeling approach. Presented experiments are performed on a prototype implementation which is described in detail and evaluated in Section 5.

4.1 Regulation of G_1/S Cell Cycle Transition

We investigate a model representing the central module of the genetic regulatory network governing the G_1/S cell cycle transition in mammalian cells [35]. 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. 5 (left)).

This simple model exemplary 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 reaches it) a certain stable state [11]. In particular, in the stable state concentrations of the species does not change, because production and degradation had reached the equilibrium. A stimulus, which is in form of change of some protein concentration caused from outside the system, evokes deflection from the stable state. When the deflection is weak enough, the system returns to the previous stable state afterwords. But when it overruns some threshold, the system approaches to 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.

4.1.1 Qualitative Modeling

First, we consider a boolean model of the network, we formulate the required properties, and we employ the parameter synthesis algorithm to synthesize valuations of unknown parameters (denoted by question marks in Fig. 5).

We define the parameterized boolean model of the given network $\mathcal{B}_\chi$ depicted in Fig. 5. To differentiate between the two outgoing regulations from both E2F1 and pRB, we choose the genes maximal activity levels $\rho(\mathrm{pRB}) = \rho(\mathrm{E2F1}) = 2$. Thresholds can be in most cases revealed from experimental data [36]. In this case we deal with a synthetic model generated by bifurcation analysis and we set the thresholds with respect to data presented in [35].

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

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

$BTS(\mathcal{B})$ may be turned directly into a parameterized Kripke structure $\mathcal{K}_\mathcal{B} = (\mathcal{P}_\chi, S, S_0, \rightarrow, AP, F)$ where $s \xrightarrow{p} s'$ if there is a transition $s \rightarrow s'$ in the model $\mathcal{B}$ with the setting of regulatory logic given by $p \in \mathcal{P}_\chi$. As the initial state set S_0 we choose states satisfying $l_{\mathrm{pRB}} = 0$ and $l_{\mathrm{E2F1}} \in \{0, 1, 2\}$, capturing perturbation in the expression of E2F1.

Now, although the stable modes (values of parameters) are unknown, $K_{\mathrm{pRB},\emptyset}$ and $K_{\mathrm{E2F1},\emptyset}$ may be obtained through static analysis of the model. A significant role for the model behaviour 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 $K_{\mathrm{E2F1},\emptyset} = 2$. To ensure that the model displays non-trivial behaviour, we also set $K_{\mathrm{pRB},\emptyset} = 1$.

To filter certain trivial paths in the previous Kripke structure, the concept of explicitly given accepting states is used. From the above regulatory logic settings follows the selection of accepting states with $l_{\mathrm{pRB}} \geq 1$, denoted F_1. A more restricting filter may be created by choosing states satisfying $l_{\mathrm{pRB}} = 2$, denoted F_2.

In this model, 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 behaviour observed in vitro. In the following we assume $\theta \in \{1, 2\}$:

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

We used the algorithm to synthesize admissible parameter valuations 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 system is considerably robust regarding the choice of θ and the setting of regulatory logic on E2F1 (i.e. $K_{\mathrm{E2F1},R}$ where $R \subseteq \{\mathrm{E2F1}, \mathrm{pRB}\}$). This might be, however, caused by the very high abstraction level of boolean models.

4.1.2 Quantitative Modeling

To analyze the model at the level of quantitative kinetics, we employ the piece-wise multi-affine abstraction (see Section 2) of the non-linear ODE model presented in [35]. The parameters in the original ODE model have been estimated by employing bifurcation analysis [11]. We show how our alternative method can be employed for synthesis of parameter valuations satisfying the required specification. Our PMA abstraction of this system is shown in Fig. 6. Each function $\varrho_i(x)$ is defined as a sum of several ramp-functions that gradually approximate the respective regulatory S-shaped curve by a polyline.

Since we detected bistability by using the qualitative model above, it follows to find how this phenomenon is affected by setting of kinetic parameters. As shown in [35], the steady behaviour of this system is strongly influenced by the degradation coefficient γ_{pRB}. Fig. 7 shows the vector field of the above system for two different values of γ_{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:

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

$RATS(\mathcal{M})$ can be directly turned into a Kripke structure $\mathcal{K}_{\mathcal{M}}$ labeled by AP. That way, LTL is interpreted over infinite paths of $RATS$.

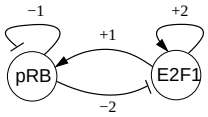
Our goal is to determine the set of parameters in the range $[0.01, 1]$ for which the concentration of $E2F1$ is greater than 8 in the stable state. This phenomenon can be specified in terms of an LTL formula $\varphi = \mathbf{FG}([E2F1] > 8)$.

We executed the algorithm for the model described above and the property φ. 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 φ is satisfied. Owing to the fact that the property φ 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 [34] 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 $E2F1 > 8$. The PKS after the above transformation had 732 states reachable from initial states, the total of 827 states were colored. The computation took approximately 0.8 seconds on a single processor core.

4.2 E. Coli Ammonium Transport Model

As another example, we consider a quantitative model specifying ammonium transport from external environment into the cells of *E. Coli* [37]. The model describes the ammonium transport process that takes effect at very low external ammonium concentration. In such conditions, the transport process complements the deficient ammonium diffusion. The process is driven by a membrane-located ammonium transport protein $AmtB$ that binds external ammonium cations NH_4ex and uses their electrical potential to conduct NH_3 into the cytoplasm. In Fig. 8, the scheme of the transport channel is shown (left) as well as the corresponding biochemical and mathematical model (right). For a sake of simplicity, pH conditions and external ammonium concentration are considered constant.



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

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

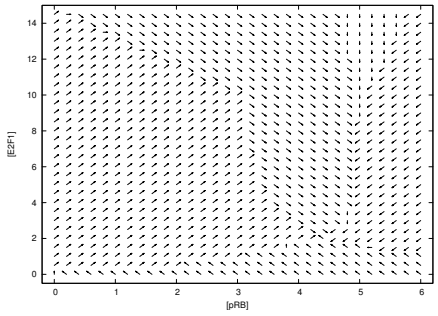
$$\frac{d[pRB]}{dt} = k_1 \frac{[E2F1]}{0.5+[E2F1]} \frac{0.5}{0.5+[pRB]} - \gamma_{pRB}[pRB]$$
$$\frac{d[E2F1]}{dt} = k_p + k_2 \frac{a^2+[E2F1]^2}{16+[E2F1]^2} \frac{5}{5+[pRB]} - \gamma_{E2F1}[E2F1]$$

Fig. 5. (left) Genetic regulatory network controlling the G_1/S transition. (middle) Regulatory logic employed for the qualitative model. (right) The original ODE model system that makes the quantitative model of the network.

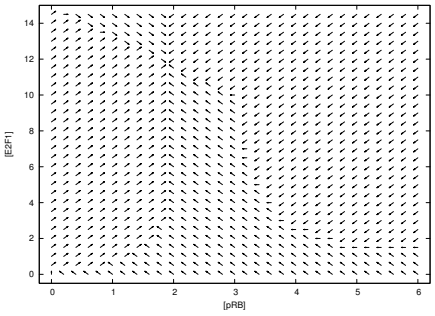
$$\frac{d[pRB]}{dt} = k_1 \varrho_1(pRB, E2F1) - \gamma_{pRB}[pRB]$$
$$\frac{d[E2F1]}{dt} = k_p + k_2 \varrho_2(pRB, E2F1) - \gamma_{E2F1}[E2F1]$$

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

Fig. 6. Piece-wise affine abstraction (PMA model) for the G_1/S transition regulatory network.

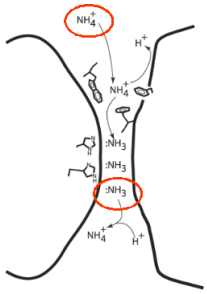


(a) $\gamma_{pRB} = 0.01$, the system stabilizes with $E2F1 < 3$



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

Fig. 7. Vector field of the liveness model.



$$AmtB + NH_4ex \overset{k_2}{\underset{k_1}{\leftrightarrows}} AmtB : NH_4$$
$$AmtB : NH_4 \overset{k_3}{\rightarrow} AmtB : NH_3 + H_{ex}$$
$$AmtB : NH_3 \overset{k_4}{\rightarrow} AmtB + NH_3in$$
$$NH_4in \overset{k_5}{\rightarrow}$$
$$NH_3in + H_{in} \overset{k_7}{\underset{k_6}{\leftrightarrows}} NH_4in$$
$$NH_3in \overset{k_9}{\underset{k_8}{\leftrightarrows}} NH_3ex$$

$$\frac{d[AmtB]}{dt} = -k_1[AmtB][NH_4ex] + k_2[AmtB : NH_4] + k_4[AmtB : NH_3]$$
$$\frac{d[AmtB:NH_3]}{dt} = k_3[AmtB : NH_4] - k_4[AmtB : NH_3]$$
$$\frac{d[AmtB:NH_4]}{dt} = k_1[AmtB][NH_4ex] - k_2[AmtB : NH_4] - k_3[AmtB : NH_4]$$
$$\frac{d[NH_3in]}{dt} = k_4[AmtB : NH_3] - k_6[NH_3in][H_{in}] + k_7[NH_4in] + k_9[NH_3ex]$$
$$\frac{d[NH_4in]}{dt} = k_6[NH_3in][H_{in}] - k_5[NH_4in] - k_7[NH_4in]$$

constant species: $NH_3ex, NH_4ex, H_{in}, H_{ex}$

initial conditions and threshold numbers:

$AmtB \in \langle 0, 1 \cdot 10^{-5} \rangle$ [7], $AmtB : NH_3 \in \langle 0, 1 \cdot 10^{-5} \rangle$ [3], $AmtB : NH_4 \in \langle 0, 1 \cdot 10^{-5} \rangle$ [3],
$NH_3in \in \langle 1 \cdot 1^{-6}, 1.1 \cdot 10^{-6} \rangle$ [6], $NH_4in \in \langle 2 \cdot 10^{-6}, 2.1 \cdot 10^{-6} \rangle$ [4],
$NH_3ex \in \langle 0, 1 \cdot 10^{-5} \rangle$ [4], $NH_4ex \in \langle 0, 1 \cdot 10^{-5} \rangle$ [4]

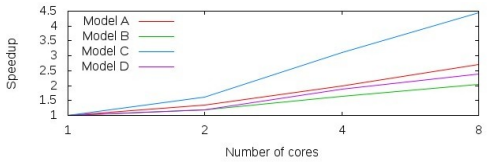
Fig. 8. Ammonium Transport Model

Owing to the membrane location of $AmtB$, *in vitro* measuring of concentration of $AmtB$-based species is impossible and therefore the estimation of kinetic parameters of this model is very difficult. To this end, we use the model to 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 ammonia concentration above the standard values. In terms of LTL model checking, we formulate 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. $NH4_in$) never exceeds the given concentration.

We performed two groups of parameter synthesis tasks. In the first group, each single parameter was considered unknown (remaining parameters were set w.r.t. [37]). In the second group, we considered a set of three unknown parameters. In all experiments, the range for all parameters has been set to $(1 \cdot 10^{-12}, 1 \cdot 10^{12})$. In Table 1, the most interesting results are summarized. The presented data show the scanned parameter set, the analyzed property with the computed valid parameter valuations, states of the PKS reached, and the best computation times. Note that each parameter which is not mentioned led trivially to validity on entire $(1 \cdot 10^{-12}, 1 \cdot 10^{12})$. Results for three parameters show good practicability also for complex parameter synthesis

P	prop.	intervals of validity	# states reached	time
k_4	φ_1	$(1 \cdot 10^{-12}, 2.7 \cdot 10^6)$	124580	30 s
k_5	φ_2	$(1.5 \cdot 10^7, 1 \cdot 10^{12})$	3068	0.40 s
k_6	φ_1	$(5.2 \cdot 10^6, 1 \cdot 10^{12})$	67572	22 s
k_6	φ_2	$\emptyset$	6319	1.8 s
k_7	φ_1	$(1 \cdot 10^{-12}, 3.3 \cdot 10^6)$	126458	33 s
k_7	φ_2	$(1.6 \cdot 10^7, 1 \cdot 10^{12})$	12523	3.5 s
k_9	φ_1	$(1 \cdot 10^{-12}, 2.7 \cdot 10^6)$	97495	20 s
k_9	φ_2	$\emptyset$	5779	1.5 s
$k_{1,6,9}$	φ_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
$k_{1,6,10}$	φ_2	$[k_1 \in (1 \cdot 10^{-12}, 1 \cdot 10') \wedge k_6 \in$ $(1 \cdot 10^{-12}, 1.4 \cdot 10^5) \wedge k_{10} \in (1.18 \cdot 10^6, 1 \cdot 10^{12})] \vee$ $[k_6 \in (1.4 \cdot 10^5, 1.07 \cdot 10^6) \wedge k_{10} \in (1 \cdot 10^{-12}, 1.18 \cdot 10^5)]$	19473	19 min

TABLE 1
Safety Model Checking Experiments



Model	possible valuations	states reached	final states	8-core runtime
A	7.5 billions	756	108	8.33 s
B	112.5 billions	2484	1	16.73 s
C	4.5 billions	1936	1	11.33 s
D	28.125 billions	864	192	8.33 s

Fig. 9. Statistics of computation on qualitative models

problems – in the case of $\{k_1, k_6, k_9\}$, a large number of parameter valuation classes ($7.2 \cdot 10^4$) were tested. This implies the significant increase of time. All experiments have been performed on an 8 core 2.2GHz CPU with 24GB RAM.

We have checked the computed validity intervals against numerical simulations. All the results show good estimation. Of special interest are individual scans of k_6 and k_9 for φ_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 $NH4_in$ 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.

5 Performance Evaluation

In order to show that our approach can be effectively deployed to a high-performance environment, we have implemented a multi-core data-parallel implementation of algorithms presented in Section 3, using the C++ programming language. The parallel implementation effectively reduces the time needed for the program execution. Because our framework has been developed to be independent of the character of input data sets, we adopt the implementation presented in [34].

The framework was developed before we started to work with qualitative models, therefore, the parallel execution was optimised primarily for quantitative models. In qualitative models, the framework loses a part of its efficiency. The reasons are the discrete character of qualitative parameters and their narrow domain ranges which decrease the scalability achievable with the prototype implementation. This could be improved by tuning the implementation for qualitative models. However, due to the experimental character of the application, we have been satisfied with the current framework.

5.1 Implementation details

As a storage primitive for state updates, we use a queue, which was introduced in Algorithm 2. The parallelisation is achieved by distributing updates in the queue to multiple threads using a hash function. Each thread possesses its own partition of the state space on which it computes. Results are then posted to the main queue for a next round of the distribution. That way, we can easily and quickly accomplish synchronisation of threads on individual levels of breadth-first search. The synchronisation is mandatory in order to avoid redundant calculations and to achieve good performance.

Both models are optimised at the level of parameter synthesis algorithm. Especially, the computation of the transition-enabling parameter valuation sets $\mathcal{P}(\alpha, \beta)$ is realized differently in both approaches. In order to reach a better computational efficiency, we represent the parameter space in terms of sets containing possible valuations combinations rather than sets of independent valuations. This feature provides strong results with respect to computational times achieved.

The prototype tool is developed in C++ using the Boost libraries and the Hoard memory allocator.

5.2 Evaluation on Qualitative Models

To critically evaluate the performance of our implementation, we created a set of models designated solely for examination of the algorithm scalability.

To obtain most unbiased data as possible we have used a set of random models built on a similar structure. We also provided these models with different Büchi automata which radically change the behaviour of the parameter synthesis algorithm.

For the performance analysis, we have used four random models with six genes in each one. The analysis tasks considered for each individual model were the following:

- A to test reachability of a set of states,
- B to detect a stable state in a specified range,
- C to detect oscillations,
- D to provide reachability analysis under specific constraints.

For each of the models, we executed multiple parameter synthesis tasks using 1, 2, 4 and 8 processor cores. The resulting speedup is presented in Fig. 9. Data for each model were obtained as an average of five independent runs on a 2.2GHz CPU with 24GB RAM.

Although we optimised our algorithm in order to cope with large sets of parameter valuations, we also tested its

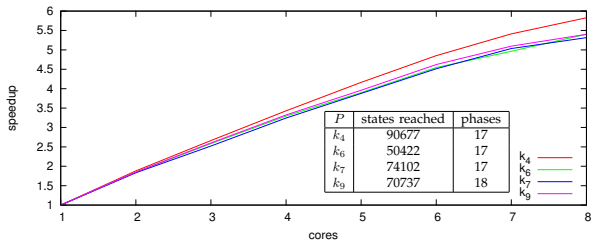


Fig. 10. Statistics of computation on quantitative models

capabilities on small ranges of parameter valuations, but in larger models. To this end, we have created a random model with 15 genes on which we tested a reachability property. On 8 cores, this test was performed in less than a minute and during the execution, the program examined over 300 000 different states.

We also examined limitations of our approach by gradually extending the number of parameters in model B. Before we encountered problem of memory lack, we were able to analyse a model with $54 \cdot 10^{15}$ possible valuations of 30 parameters simultaneously unknown. The test took 2 minutes on 8 cores.

5.3 Evaluation on Quantitative Models

The scalability of the algorithm for quantitative modelling approach has been evaluated by running the algorithm on the PMA model described in Section 4.2. The model was checked against the property $\varphi_1 = \mathbf{G}[NH_3in] < 1.1 \cdot 10^6$ (see Section 4.2). We did not evaluate it against property φ_2 since the individual checks were completed in less than 5 seconds.

Trials consisted of several runs of the parameter synthesis procedure on the model with a single parameter (selected from $\{k_4, k_6, k_7, k_9\}$). The number of cores ranged from 1 to 8. The trials were also run on an 8 core machine with 2.2GHz CPUs and 24GB RAM. From the times of individual runs in each trial, average was considered. Results are shown in Figure 10. Table in the graph shows for each PKS the number of BFS levels and the number of reachable states, i.e., the number of states reached during the initial coloring computation. Note that most of Algorithm 1 running time is spent in the computation of the initial coloring, therefore, the numbers are close to the state counts from Table 1.

Recall that the complexity of Algorithm 2 is $\mathcal{O}(|F||S|^2\zeta)$ and that the factor $|S|$ comes from the upper bound on the number of BFS levels in PKS. Notice that on the ammonium transport model, the number of BFS levels is much smaller than $|S|$.

6 Conclusions

The presented solution to parameter synthesis reduces the problem to model checking of a parameterised family of Kripke structures in which the set of states is shared but the transition relation is variable. At this level, we provide a model checking algorithm that can be in the average case significantly more efficient than a series of many model checking procedures each performed for every individual parameter set.

To enable effective usage of high-performance computing, we have provided a parallel implementation tuned for multi-core platforms. Two biological models have been considered to show the applicability to systems biology. All parameter synthesis tasks realized for a single unknown parameter were computed in acceptable times on common multi-core hardware, parameter synthesis tasks for combinations of unknown parameters were finished in minutes on the qualitative and tens of minutes on the quantitative model. In this setting we were able to synthesise valuations for up-to three simultaneously unknown parameters in multi-dimensional quantitative ODE models (7 dimensions were considered) and up-to thirty simultaneously unknown parameters in boolean models composed from 6 genes.

Acknowledgment

This work has been supported by the Czech Science Foundation grant No. 201/09/1389.

References

[1] H. Kitano, "Systems biology: a brief overview." *Science*, vol. 295, no. 5560, pp. 1662–4, 2002.
[2] F. Horn and R. Jackson, "General mass action kinetics," *Archive for Rational Mechanics and Analysis*, vol. 47, pp. 81–116, 1972.
[3] D. Thieffry and R. Thomas, "Dynamical Behavior of Biological Regulatory Networks," *Bulletin of Mathematical Biology*, vol. 57, no. 2, pp. 277–297, 1995.
[4] H. de Jong etal., "Genetic network analyzer: qualitative simulation of genetic regulatory networks," *Bioinformatics*, vol. 19, no. 3, pp. 336–344, 2003.
[5] Y. Kaznessis, "Models for synthetic biology," *BMC Syst. Biol.*, vol. 47, 2007.
[6] S. Key, *Fundamentals of Statistical Signal Processing Volume I: Estimation Theory*. Prentice Hall, 1993.
[7] M. Rodriguez-Fernandez etal., "A hybrid approach for efficient and robust parameter estimation in biochemical pathways," *Biosystems*, vol. 83, no. 2-3, pp. 248 – 265, 2006.
[8] P. Lindner and B. Hitzmann, "Experimental design for optimal parameter estimation of an enzyme kinetic process based on the analysis of the fisher information matrix," *Journal of Theoretical Biology*, vol. 238, no. 1, pp. 111 – 123, 2006.
[9] D. Battogtokh etal., "Bifurcation analysis of a model of the budding yeast cell cycle," *Chaos*, vol. 14, no. 3, pp. 653–661, 2004.
[10] D. Angeli etal., "Detection of multistability, bifurcations, and hysteresis in a large class of biological positive-feedback systems," *Proc Natl Acad Sci USA*, vol. 101, no. 7, pp. 1822–1827, 2004.
[11] J. Lu, H. W. Engl, R. Machné, and P. Schuster, "Inverse bifurcation analysis of a model for the mammalian g1/s regulatory module," in *Proc. of BIRD'07*. Springer, 2007, pp. 168–184.
[12] T. A. Henzinger and H. Wong-Toi, "Using hytech to synthesize control parameters for a steam boiler," in *Formal Methods for Industrial Applications*, 1995, pp. 265–282.
[13] G. Frehse etal., "A counterexample-guided approach to parameter synthesis for linear hybrid automata," in *HSCC*, 2008, pp. 187–200.
[14] G. Batt etal., "Robustness analysis and tuning of synthetic gene networks," *Bioinformatics*, vol. 23, no. 18, pp. 2415–2422, 2007.
[15] B. Yordanov and C. Belta, "Parameter synthesis for piecewise affine systems from temporal logic specifications," in *Hybrid Systems: Computation and Control*, vol. 4981, 2008, pp. 542–555.
[16] A. Donzé, B. Krogh, and A. Rajhans, "Parameter synthesis for hybrid systems with an application to simulink models," in *HSCC*, ser. LNCS, vol. 5469. Springer, 2009, pp. 165–179.
[17] E.M. Clarke etal., *Model Checking*. MIT Press, 2000.

[18] J.F. Guespin-Michel etal., "Epigenesis and dynamic similarity in two regulatory networks in pseudomonas aeruginosa," *Acta Biotheoretica*, vol. 52, pp. 379–390, 2004.

[19] L. Calzone, F. Fages, and S. Soliman, "BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge," *Bioinformatics*, vol. 22, no. 14, pp. 1805–1807, 2006.

[20] P.T. Monteiro etal., "Temporal logic patterns for querying dynamic models of cellular interaction networks," *Bioinformatics*, vol. 24, no. 16, pp. 227–233, 2008.

[21] A. Rizk etal., "A general computational method for robustness analysis with applications to synthetic gene networks," *Bioinformatics*, vol. 25, no. 12, pp. i169–178, 2009.

[22] R. Donaldson and D. Gilbert, "A model checking approach to the parameter estimation of biochemical pathways," in *Computational Methods in Systems Biology*, ser. LNBI, vol. 5307. Springer, 2008, pp. 269–287.

[23] P. Collins etal., "Abstraction of biochemical reaction systems on polytopes," in *Proc. of 18th IFAC World Congress*. Elsevier, 2011, in press.

[24] G. Batt et.al., "Efficient parameter search for qualitative models of regulatory networks using symbolic model checking," INRIA, Tech. Rep., 2010.

[25] P. Ballarini et.al., "Taming the complexity of biological pathways through parallel computing," *Brief. in Bioinformatics*, vol. 10, no. 3, pp. 278–288, 2009.

[26] J. Barnat, L. Brim, and D. Šafránek, "High-performance analysis of biological systems dynamics with the divine model checker," *Brief. in Bioinformatics*, vol. 11, pp. 301–312, 2010.

[27] A. Donzé etal., "Parameter synthesis in nonlinear dynamical systems: Application to systems biology," in *Proc. of RECOMB 2009*, ser. RECOMB 2'09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 155–169.

[28] Z. Khalis etal., "The smbionet method for discovering models of gene regulatory networks," *Genes, Genomes and Genomics*, vol. 3, no. 1, 2009.

[29] B. Pal etal., "Buspec: A framework for generation of verification aids for standard bus protocol specifications," *Integration, the VLSI Journal*, vol. 40, no. 3, pp. 285 – 304, 2007.

[30] K. Hamaguchi etal., "Formal verification of speed-dependent asynchronous circuits using symbolic model checking of branching time regular temporal logic," in *Computer Aided Verification*, ser. LNCS, vol. 575/1992. Springer, 1991, pp. 410–420.

[31] C. Chaouiya etal., "Logical modelling of regulatory networks with ginsim 2.3," *Biosystems*, vol. 97, pp. 134–139, 5 2009.

[32] G. Bernot etal., "Application of formal methods to biological regulatory networks: Extending thomas' asynchronous logical approach with temporal logic," *Journal of Theoretical Biology*, vol. 229, no. 3, pp. 339–347, 2004.

[33] T. Mestl et.al., "A mathematical framework for describing and analysing gene regulatory networks," *Journal of Theoretical Biology*, vol. 176, no. 2, pp. 291–300, 1995.

[34] J. Barnat etal., "Parameter Scanning by Parallel Model Checking with Applications in Systems Biology," in *Proc. of HiBi/PDMC 2010*. IEEE, 2010, pp. 95–104.

[35] M. Swat, A. Kel, and H. Herzel, "Bifurcation analysis of the regulatory modules of the mammalian G1/S transition," *Bioinformatics*, vol. 20, no. 10, pp. 1506–1511, 2004.

[36] S. Liang, S. Fuhrman, and R. Somogyi, "Reveal, a general reverse engineering algorithm for inference of genetic network architectures," *Pac Symp Biocomput.*, vol. 1998, no. 3, pp. 18–29, 1998.

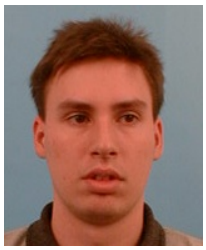
[37] H. Ma, F. Boogerd, and I. Goryanin, "Modelling nitrogen assimilation of escherichia coli at low ammonium concentration," *Journal of Biotechnology*, vol. 144, pp. 175–83, 2009.

J. Barnat obtained MSc degrees in computer science from the Masaryk University (Czech Republic) and completed his PhD thesis in computer science at the same university. Since 2005 he is an Assistant Professor at Faculty of Informatics, Masaryk University.

L. Brim obtained MSc degree in mathematics from the University of Brno (Czech Republic). He completed his PhD thesis in computer science at Czechoslovak Academy of Science in 1986. In 1996 he has become an Associate Professor at the Faculty of Informatics, Masaryk University. Since 2006 he is a full Professor at the same faculty.

A. Krejčí is student of bioinformatics at the Masaryk University (Czech Republic) since 2008. Currently finishing his BSc degrees.

A. Streck obtained BSc degree in computer science from the Masaryk University (Czech Republic) in 2011. Currently student of MSc programme and member of the Sybila laboratory at the Masaryk University.



D. Šafránek obtained MSc degree in computer science from the Masaryk University (Czech Republic) and completed his PhD thesis in computer science at the same university. Since 2009 he is an Assistant Professor at Faculty of Informatics, Masaryk University.

M. Vejnár obtained BSc degree at the Masaryk University (Czech Republic) in 2008. He is currently finishing his MSc degree.

T. Vejpustek currently finishing his BSc degree at the Masaryk University. Since 2010, he is a member of the Sybila laboratory.