Parallel SMT-Based Parameter Synthesis with Application to Piecewise Multi-affine Systems

Nikola Beneš, Luboš Brim, Martin Demko,
Samuel Pastva, and David Šafránek(✉)

Systems Biology Laboratory, Faculty of Informatics, Masaryk University,
Botanická 68a, 602 00 Brno, Czech Republic
{xbenes3,brim,xdemko,xpastva,xsafran1}@fi.muni.cz

Abstract. We propose a novel scalable parallel algorithm for synthesis of interdependent parameters from CTL specifications for non-linear dynamical systems. The method employs a symbolic representation of sets of parameter valuations in terms of the first-order theory of the reals. To demonstrate its practicability, we apply the method to a class of piecewise multi-affine dynamical systems representing dynamics of biological systems with complex non-linear behaviour.

1 Introduction

Complex dynamical phenomena arising in real-world systems such as biological, biophysical processes, or networks involving economic and social interactions are typically formalised by means of *dynamical systems* employing the framework of non-linear ordinary differential equations that are highly parameterised. In most cases, the model complexity and the number of *unknown parameters* do not allow to analyse the systems analytically. Computer-aided analysis of complex dynamical systems and their models is a necessary precursor for design of reliable cyber-physical and cyber-biological systems such as synthetic design and control of living cells [21,32] or safe medical treatment [1].

Phenomena occurring in the time domain of systems dynamics can be encoded in *temporal logics* (TL). TL have the advantage of rigorous and abstract representation of sequences (or even branching structures) of desired observable events in systems dynamics including quantitative bounds on time and variable values [8,10,31] and can be also combined with frequency-domain analysis [19].

In this paper, we target the problem of *global parameter synthesis* (extended with *static constraints* over parameter space). To solve the problem means to *identify parameter valuations* that *satisfy* a given set of *TL properties* universally (regardless of specific initial conditions) provided that the specified *static constraints* are also satisfied. Static constraints include *a priori* known restrictions, dependencies and correlations of individual parameter valuations (e.g., restrictions on production/degradation parameters ratio [36]).

This work has been supported by the Czech Science Foundation grant GA15-11089S.

C. Artho et al. (Eds.): ATVA 2016, LNCS 9938, pp. 192–208, 2016.
DOI: 10.1007/978-3-319-46520-3_13

In general, computationally efficient (scalable) global parameter synthesis under large uncertainty of a number of unknown parameters and unrestricted initial conditions with respect to satisfaction of a given TL specification remains a challenge. Existing techniques do not sufficiently target non-quantitative branching-time properties that can efficiently cope with *decision* events and *multi-stability* arising in complex real-world systems (e.g., existence and characteristics of unstable states in chemical or electric power systems [14,28], or reachability of multiple stable states in a biological switch [24,37]). The situation is even worse if the parameters are *interdependent*.

We introduce a novel approach to global parameter synthesis based on distributed CTL model checking. In particular, parameter synthesis for a given CTL specification and the given parameter space is solved by the coloured model checking technique [3,11] extended with symbolic encoding of parameter valuations and constraints. The main principle of our new technique relies on symbolic representation of parameters. The parameter encoding relies on the first-order theory of reals for which the satisfiability can be algorithmically solved [6]. In particular, we employ *Satisfiability Modulo Theories* (SMT) as a subprocedure wrapped inside the enumerative distributed CTL model checking algorithm. This allows for every state to synthesise a first-order formula that encodes the parameter valuations for which the CTL specification is guaranteed to be satisfied in that state. A significant advantage of employing *enumerative* CTL model checking for parameter synthesis is its capability of computing integrated information in a single parallel run. In particular, the parameter valuations are synthesised for every state and every subformula of the given CTL property. This allows to compute the parameter synthesis for a set of CTL formulae at once.

The distributed algorithm is based on assumption-based CTL model checking we have introduced in [13]. Its extension to parameter synthesis for interval-representation of parameter sets has been considered in [11]. The main drawback of that approach has been the restriction to synthesis of algebraically independent parameters. By using SMT, we significantly generalise the method to parameterisations including interdependent parameters. The new algorithm retains good scaling with increasing number of computing nodes. Since the number of calls to the SMT solver is proportional to the size of the state space, distribution of the state space and related computing tasks realise efficiently the divide&conquer paradigm while minimising the number of SMT calls and parallelising their computation on independent computing nodes.

The typical application domains for our method are highly parameterised systems appearing in systems biology (e.g., dynamics of gene regulatory networks represented by Boolean networks or non-linear continuous systems [3]) or control and verification of hybrid systems [18].

Summary of Our Contribution. The main result of this paper is a new parallel algorithm for parameter synthesis from CTL specifications for dynamical systems with *interdependent* parameters. Our method is unique in combining enumerative model checking with SMT solvers for parameter synthesis. It is distinctive in the following aspects:

1. universality – the method works on a large family of finite-state discrete dynamical systems or finite-state qualitative abstractions of continuous systems in which parameterisations can be encoded in a first-order logic over reals,
2. user feedback – the resulting parameter sets are sampled from the SMT representation and further post-processed by third-party tools such as Symba [29],
3. high-performance – the method is supplied with a parallel distributed-memory algorithm that allows good scalability in a distributed environment.

In order to evaluate our approach, we apply the method to piecewise multi-affine dynamical systems where the systems dynamics is a linear function of the parameters. In the case study we use a model of a gene regulatory network.

Related Work. Monitoring-based synthesis techniques have been developed for continuous-time and discrete-time dynamical systems [4,10,17,34,35] and linear-time TL. These techniques rely on numerical solvers which are well-developed for systems with fixed parameters or small parameter spaces (perturbations). An advantage of these techniques is that they consider the function defining the systems dynamics as a black box provided that there is basically no limitation on the form of parameterisation of the system. The main drawback is the need to sample the parameter space and initial states while losing robust guarantees for the results. This drawback can be overcome by replacing numerical solvers with Satisfiability Modulo Theories (SMT) solvers that can cope with non-linear functions and real domains up to required precision [23]. However, these techniques are limited to reachability analysis [30] and their extension to work with general TL specifications is a non-trivial task yet to be explored. The method in [16] targets reachability analysis and combines guided random exploration of the state space together with sensitivity analysis.

Existing techniques for global parameter synthesis from TL specification are either based on model checking performed directly on a qualitative finite quotient of systems dynamics [3,7,8,11] or on techniques from hybrid systems [9]. Typical limitation of these methods is determined by restrictions on the form of allowed parameterisations. By employing SMT, we obtain support for all parameterisations and constraints that can be encoded in a first-order logic over reals. This is a significant improvement over our previous work [11] that has been limited to algebraically independent parameters only. In [8,26] parameter sets are encoded symbolically in terms of polytopes allowing linear dependencies only. In [25], the authors employ symbolic bounded model checking with SMT to parameter synthesis of discrete synchronous models of weighted genetic regulatory networks. To the best of our knowledge, none of these methods have been parallelised.

In [20], the authors provide a parameter synthesis algorithm for polynomial dynamical systems based on the Bernstein polynomial representation. The approach targets discrete time dynamical systems.

2 Definitions and Problem Statement

The general setting of the parameter synthesis problem is given by the notion of a parameterised Kripke structure [3]. This notion encapsulates a family of Kripke structures with the same state space but with different transitions. The existence of transitions is governed by parameter valuations.

Definition 1. *Let* AP *be a set of atomic propositions. A* parameterised Kripke structure *(PKS) over* AP *is a tuple* $\mathcal{K} = (\mathcal{P}, S, I, \rightarrow, L)$ *where* $\mathcal{P}$ *is a finite set of parameter valuations,* S *is a finite set of states,* $I \subseteq S$ *is the set of initial states,* $L : S \rightarrow 2^{\mathrm{AP}}$ *is a labelling of the states and* $\rightarrow \subseteq S \times P \times S$ *is a transition relation labelled with the parameter valuations. We write* $s \xrightarrow{p} t$ *instead of* $(s, p, t) \in \rightarrow$*. We assume that the PKS is total, i.e. for all* s*,* p *there exists at least one* t *such that* $s \xrightarrow{p} t$*.*

Fixing a concrete parameter valuation $p \in \mathcal{P}$ reduces the parameterised Kripke structure $\mathcal{K}$ to a standard Kripke structure $\mathcal{K}_p = (S, I, \xrightarrow{p}, L)$. We use the notation $\mathcal{P}(s,t) = \{p \in \mathcal{P} \mid s \xrightarrow{p} t\}$ to denote the set of all parameter valuations that enable the transition from s to t. A parameterised Kripke structure can be seen as a Kripke structure with labelled transitions, where the transition labels are the sets $\mathcal{P}(s,t)$.

In the following, we assume that parameter valuations of the PKS are represented symbolically. We thus assume that we are given a (first-order) theory that is interpreted over the parameter valuations; every $\mathcal{P}(s,t)$ is then described via a formula $\Phi_{s,t}$ such that $\mathcal{P}(s,t) = \{p \in \mathcal{P} \mid p \models \Phi_{s,t}\}$. The symbolic representation of a PKS can be thus seen as a Kripke structure with labelled transitions, where the transition labels are the formulae $\Phi_{s,t}$.

To express properties of interest, we employ the standard branching time logic CTL. The formulae of CTL are defined by the following abstract syntax:

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

where q ranges over the atomic propositions from the set AP. We use the standard abbreviations such as $\mathbf{EF}\,\varphi \equiv \mathbf{E}(\mathtt{tt}\,\mathbf{U}\,\varphi)$ and $\mathbf{AG}\,\varphi \equiv \neg\mathbf{EF}\,\neg\varphi$.

Note that there are two sets of formulae we use here: the CTL formulae that consider the states of the PKS and the formulae that are used to symbolically describe the parameter sets in the PKS. To easily distinguish between these two kinds of formulae, we shall adopt the convention to denote CTL formulae by lower-case Greek letters φ, ψ, etc., and the parameter formulae by upper-case Greek letters Φ, Ψ, etc.

The Problem Formulation. Let $\mathcal{K} = (\mathcal{P}, S, I, \rightarrow, L)$ be a parameterised Kripke structure over AP with symbolic description as explained above and let Φ_I be an initial parameter constraint, described using the same theory as the one used in the symbolic description. Let further φ be a CTL formula over AP. The *parametric synthesis problem* is, given $\mathcal{K}$, Φ_I, and φ, to find the function $\mathcal{F}$ that assigns to every state of the Kripke structure the set of parameters that

ensure the satisfaction of the CTL formula. Formally, the function is described as follows:

$$\mathcal{F}(s) = \{p \in \mathcal{P} \mid p \models \Phi_I, s \models_{\mathcal{K}_p} \varphi\}. \quad (1)$$

We extend the basic parametric synthesis problem with the possibility of an optimisation criterion, given as an objective function $f : \mathcal{P} \to \mathbb{R}$ that assigns a real value to every parameter valuation. The *parametric optimisation problem* is, given $\mathcal{K}$, Φ_I, φ, and f, to find the maximal value of f over the set $\mathcal{F}(s)$ for every state s, i.e. to find the function m satisfying $m(s) = \max\{f(p) \mid p \in \mathcal{F}(s)\}$. We are also interested in the parameter valuations that realise this maximum.

3 Parallel Algorithm

We are now going to describe the distributed-memory semi-symbolic parameter synthesis algorithm that solves the parameter synthesis problem described above, i.e. finding the function $\mathcal{F}$. The parametric optimisation problem is then solved using the result of this algorithm as an input to further tools that provide SMT optimisation, such as Symba [29]. We assume that the symbolic description of the parameters is given in a decidable first-order theory.

We adapt the assumption-based distributed CTL model checking algorithm [11,13] as the basis for our work. In this approach, the algorithm is run on a cluster of n computational nodes (workstations). Each workstation owns a part of the original PKS as defined by a partition function. This part is extended with the so called *border states*. Intuitively, border states are states that in fact belong to another computational node and represent the missing parts of the state space. They serve as a proxy between two parts.

More precisely, we define a *PKS fragment* $\mathcal{K}_i$ to be a substructure of the PKS $\mathcal{K}$ satisfying the property that every state in $\mathcal{K}_i$ has either no successor in $\mathcal{K}_i$ or it has exactly the same successors as in $\mathcal{K}$. The states without any successors in $\mathcal{K}_i$ are called the *border states* of $\mathcal{K}_i$. A partition of the PKS $\mathcal{K}$ is a finite set of PKS fragments $\mathcal{K}_1, \ldots, \mathcal{K}_n$ such that every state of $\mathcal{K}$ is present in exactly one $\mathcal{K}_i$ as a non-border state; it may be present in several other $\mathcal{K}_j$ as a border state. In fact, every border state is stored several times: as original one on the node that owns it and as duplicates on nodes that own its predecessors.

To define the semantics of CTL formulae over fragments we need to adapt the standard semantic definition. To that end, we define the notion of the truth under assumptions associated with border states. We start by recalling the notion of an assumption function of [11], itself an extension of the original assumption functions of [13]. However, as we want to deal with the parameters in a symbolic way, we then adapt the notions to our semi-symbolic setting.

For a CTL formula ψ, let $cl(\psi)$ denote the set of all subformulae of ψ and let $tcl(\psi)$ denote the set of all temporal subformulae of ψ. An *assumption function* for a parameterised Kripke structure $\mathcal{K}$ and a CTL formula ψ is defined as a partial function of type $\mathcal{A} : \mathcal{P} \times S \times cl(\psi) \to Bool$. The values $\mathcal{A}(p, s, \varphi)$ are called *assumptions*. We use the notation $\mathcal{A}(p, s, \varphi) = \perp$ to say that the value of $\mathcal{A}(p, s, \varphi)$ is undefined. By $\mathcal{A}_\perp$ we denote the assumption function which is

undefined for all inputs. Intuitively, $\mathcal{A}(p, s, \varphi) = \texttt{tt}$ if we can assume that φ holds in the state s under parameter valuation p, $\mathcal{A}(p, s, \varphi) = \texttt{ff}$ if we can assume that φ does not hold in the state s under parameter valuation p, and $\mathcal{A}(p, s, \varphi) = \bot$ if we cannot assume anything.

Instead of working with the explicit assumption functions as described in [11], we want to deal with the parameters symbolically. We thus replace the assumption functions with *symbolic assumption functions* defined as follows. A symbolic assumption $\widetilde{\mathcal{A}}$ is a function that assigns to each pair (s, φ) a pair of formulae (Φ_t, Φ_f) such that for all $p \in \mathcal{P}$: $p \models \Phi_t$ iff $\mathcal{A}(p, s, \varphi) = \texttt{tt}$ and $p \models \Phi_f$ iff $\mathcal{A}(p, s, \varphi) = \texttt{ff}$. Each such function thus divides the set of all parameter valuations into three sets: those parameters that ensure the satisfaction of φ (Φ_t), those that ensure that φ is not satisfied (Φ_f), and finally those parameter valuations under which the satisfaction of φ is undefined ($\neg\Phi_t \wedge \neg\Phi_f$).

To simplify some of the notation in the algorithms below, we sometimes deal with the two parts (true and false) of the symbolic assumption function separately and use the notation $(\widetilde{\mathcal{A}}^t(s, \varphi), \widetilde{\mathcal{A}}^f(s, \varphi)) = \widetilde{\mathcal{A}}(s, \varphi)$.

The main operation of the distributed algorithm is the iterative computation of the symbolic assumption functions starting from the simplest subformulae of ψ (the atomic propositions) and moving towards ψ. The algorithm takes into account the symbolic assumptions of border states, initially set to $\bot$. The symbolic assumptions for non-temporal subformulae are easily computed as follows:

$$\widetilde{\mathcal{A}}(s, p) = (\texttt{tt}, \texttt{ff}) \text{ if } p \in L(s), (\texttt{ff}, \texttt{tt}) \text{ otherwise}$$
$$\widetilde{\mathcal{A}}(s, \varphi_1 \wedge \varphi_2) = (\widetilde{\mathcal{A}}^t(s, \varphi_1) \wedge \widetilde{\mathcal{A}}^t(s, \varphi_2), \widetilde{\mathcal{A}}^f(s, \varphi_1) \vee \widetilde{\mathcal{A}}^f(s, \varphi_2))$$
$$\widetilde{\mathcal{A}}(s, \neg\varphi) = (\widetilde{\mathcal{A}}^f(s, \varphi), \widetilde{\mathcal{A}}^t(s, \varphi))$$

The symbolic assumptions for temporal subformulae are computed via Algorithms 1, 2, and 3 . Each of these algorithms assumes that all possible assumptions for all subformulae have been already computed (given the current assumptions on border states).

Algorithm 1 computes the assumptions for temporal subformulae of the form $\mathbf{EX}\,\varphi$ (existential next). Initially, the assumption function is set to "false for all parameter valuations". Then, the algorithm iteratively collects assumptions about φ and propagates the information into predecessor states. This propagation extends the set of parameters for which the assumption is true and reduces the set of parameters for which the assumption is false. This ensures that if a state under given parameter valuation has at least one successor that satisfies φ (under the same parameter valuation), this valuation is going to be included in the true assumption formula for that state. Moreover, if all successors of a state under given parameter valuation refute φ, that valuation is going to be included in the false assumption formula for that state. Finally, if a state under given parameter valuation has no successors that satisfy φ and at least one successor whose satisfaction of φ is undefined in the current assumption, this parameter valuation is not going to be included in either the true or false assumption function.

Algorithm 2 computes the assumptions for temporal subformulae of the form $\mathbf{E}(\varphi_1 \,\mathbf{U}\, \varphi_2)$ (existential until). Initially, the assumption function for all non-border states is set to the assumption for φ_2. The propagation of assumptions works similarly to the previous case, with the two differences that (a) assumptions are only changed for states that satisfy φ_1 and (b) once a state's assumptions change, the state is returned to the queue for processing. This ensures that the assumptions propagate as much as possible. To determine whether a state's assumptions have changed, we employ the SMT-solver. The convergence of this procedure is guaranteed due to the monotonicity of the computation. As there is only a finite number of states and a finite number of parameter formulae in the system, the symbolic assumptions $\widetilde{\mathcal{A}}^t(s', \psi)$ and $\widetilde{\mathcal{A}}^f(s', \psi)$, which are build out of these parameter formulae using conjunctions and disjunctions, shall eventually reach a fixed point.

The last Algorithm 3, which computes the assumptions for temporal subformulae of the form $\mathbf{A}(\varphi_1 \,\mathbf{U}\, \varphi_2)$ (universal until), is slightly more complex. Contrary to the $\mathbf{EX}\,\varphi$ and $\mathbf{E}(\varphi_1 \,\mathbf{U}\, \varphi_2)$ cases, which required at least one successor of a state to be valid in order to add assumptions to the true part, the computation of $\mathbf{A}(\varphi_1 \,\mathbf{U}\, \varphi_2)$ needs all successor states (under given parameter valuation) to be valid. In order to ensure this, we need an auxiliary formula $\mathcal{T}(s, s')$ for each pair of states s, s'. One can see this auxiliary formula as a "copy" of the transitions in the PKS. During the propagation phase, the encountered transitions are removed from $\mathcal{T}$ and only as a parameter valuation leaves $\mathcal{T}(s', s)$ for all s, it may be added to the true assumption function. Note that the formula $\widehat{\Phi}_{s'} \wedge \bigwedge_{s' \to s} \neg \mathcal{T}(s', s)$ may be interpreted as a set difference between the set of all outgoing transitions of s' and the set of those outgoing transitions of s' that remain in $\mathcal{T}$.

We are now ready to describe the main algorithm for distributed-memory parameter synthesis. In order to compute the assumption function in the distributed environment, we iteratively compute assumption functions that are defined on fragments of the system $\mathcal{K}$. The algorithm starts by partitioning the given state space of $\mathcal{K}$ among the nodes using a partition function. There are many different partition functions that can be used; one function that is often used is random partitioning.

The main idea of the entire distributed computation, summarised in Algorithm 5, is the following. Each fragment $\mathcal{K}_i$ is managed by a separate process (node) P_i. These processes are running in parallel (simultaneously on each node). Each process P_i initialises the assumption function $\mathcal{A}_i$ to the undefined assumption function $\mathcal{A}_\perp$. After initialisation, it computes the new assumption function from the initial assumption function using the algorithms described above.

Once the algorithm has finished computing the symbolic assumptions, the node exchanges information about border states with other nodes. It sends to each other node the information it has about that node's border states and receives similar information from other nodes. After this exchange is completed, the computation is restarted. These steps are repeated until the whole network reaches a fixpoint, i.e. until no new information is computed by any node.

Algorithm 1. Compute symbolic assumptions for $\mathbf{EX}\,\varphi$

Require: PKS fragment $\mathcal{K}$, CTL formula $\psi = \mathbf{EX}\,\varphi$, initial assumptions $\widetilde{\mathcal{A}}_{in}$
Ensure: new symbolic assumptions $\widetilde{\mathcal{A}}$
  $\widetilde{\mathcal{A}} := \widetilde{\mathcal{A}}_{in}$
  set $\widetilde{\mathcal{A}}(s,\psi) := (\mathtt{ff},\mathtt{tt})$ for all non-border states s
  $init := \{(s,\Phi_t,\Phi_f) \mid \widetilde{\mathcal{A}}_{in}(s,\varphi) = (\Phi_t,\Phi_f)\}$
  for (s,Φ_t,Φ_f) **in** $init$ **do**
    for $(s',\Phi_{s',s})$ **in** $pred(s)$ **do**
      $\widetilde{\mathcal{A}}^t(s',\psi) := \widetilde{\mathcal{A}}^t(s',\psi) \vee (\Phi_t \wedge \Phi_{s',s})$
      $\widetilde{\mathcal{A}}^f(s',\psi) := \widetilde{\mathcal{A}}^f(s',\psi) \wedge (\Phi_f \vee \neg\Phi_{s',s}))$

Algorithm 2. Compute symbolic assumptions for $\mathbf{E}(\varphi_1\,\mathbf{U}\,\varphi_2)$

Require: PKS fragment $\mathcal{K}$, CTL formula $\psi = \mathbf{E}(\varphi_1\,\mathbf{U}\,\varphi_2)$, initial assumptions $\widetilde{\mathcal{A}}_{in}$
Ensure: new symbolic assumptions $\widetilde{\mathcal{A}}$
  $\widetilde{\mathcal{A}} := \widetilde{\mathcal{A}}_{in}$
  set $\widetilde{\mathcal{A}}(s,\psi) := \widetilde{\mathcal{A}}_{in}(s,\varphi_2)$ for all non-border states s
  $queue := S$ (all states)
  while $queue$ not empty **do**
    select and remove s from $queue$
    for $(s',\Phi_{s',s})$ **in** $pred(s)$ **do**
      $\widetilde{\mathcal{A}}^t(s',\psi) := \widetilde{\mathcal{A}}^t(s',\psi) \vee \left(\widetilde{\mathcal{A}}^t(s',\varphi_1) \wedge \widetilde{\mathcal{A}}^t(s,\psi) \wedge \Phi_{s',s}\right)$
      $\widetilde{\mathcal{A}}^f(s',\psi) := \widetilde{\mathcal{A}}^f(s'\psi) \wedge \left(\widetilde{\mathcal{A}}^f(s',\varphi_1) \vee \widetilde{\mathcal{A}}^f(s,\psi) \vee \neg\Phi_{s',s}\right)$
      if $\widetilde{\mathcal{A}}(s',\psi)$ was changed and $s' \notin queue$ **then**
        add s' to $queue$

Algorithm 3. Compute symbolic assumptions for $\mathbf{A}(\varphi_1\,\mathbf{U}\,\varphi_2)$

Require: PKS fragment $\mathcal{K}$, CTL formula $\psi = \mathbf{A}(\varphi_1\,\mathbf{U}\,\varphi_2)$, initial assumptions $\widetilde{\mathcal{A}}_{in}$
Ensure: new symbolic assumptions $\widetilde{\mathcal{A}}$
  $\widetilde{\mathcal{A}} := \widetilde{\mathcal{A}}_{in}$
  for all non-border states s **do**
    $\widehat{\Phi}_s := \bigvee_{s\to s'} \Phi_{s,s'}$
    set $\widetilde{\mathcal{A}}^t(s,\psi) := \widetilde{\mathcal{A}}^t(s,\varphi_2)$
    set $\widetilde{\mathcal{A}}^f(s,\psi) := (\widetilde{\mathcal{A}}^f(s,\varphi_1) \vee \neg\widehat{\Phi}_s) \wedge \neg\widetilde{\mathcal{A}}^t(s,\varphi_2)$
  $\mathcal{T}(s,s') := \Phi_{s,s'}$ for all $s \to s'$
  $queue := S$ (all states)
  while $queue$ not empty **do**
    select and remove s from $queue$
    for $(s',\Phi_{s',s})$ **in** $pred(s)$ **do**
      $\mathcal{T}(s',s) := \mathcal{T}(s',s) \wedge \neg\widetilde{\mathcal{A}}^t(s,\psi)$
      $\widetilde{\mathcal{A}}^t(s',\psi) := \widetilde{\mathcal{A}}^t(s',\psi) \vee \left(\widetilde{\mathcal{A}}^t(s',\varphi_1) \wedge \widehat{\Phi}_{s'} \wedge \bigwedge_{s'\to s} \neg\mathcal{T}(s',s)\right)$
      $\widetilde{\mathcal{A}}^f(s',\psi) := \widetilde{\mathcal{A}}^f(s',\psi) \vee \left(\widetilde{\mathcal{A}}^f(s,\psi) \wedge \neg\widetilde{\mathcal{A}}^t(s',\psi) \wedge \Phi_{s',s}\right)$
      if $\widetilde{\mathcal{A}}(s',\psi)$ was changed and $s' \notin queue$ **then**
        add s' to $queue$

Algorithm 4. Solve cycles

Require: PKS fragment $\mathcal{K}$, CTL formula ψ, initial assumptions $\widetilde{\mathcal{A}}_{in}$
Ensure: new symbolic assumptions $\widetilde{\mathcal{A}}$
  $\mathcal{M}_s := \mathtt{tt}$ for all non-border states s
  for $\varphi \in tcl(\psi)$ {sorted from smallest} **do**
    for $s \in S$ **do**
      $\mathcal{U} := \mathcal{M}_s \wedge \neg\widetilde{\mathcal{A}}^t(s,\varphi) \wedge \neg\widetilde{\mathcal{A}}^f(s,\varphi)$
      $\widetilde{\mathcal{A}}^f(s,\varphi) := \widetilde{\mathcal{A}}^f(s,\varphi) \vee \mathcal{U}$
      $\mathcal{M}_s := \mathcal{M}_s \wedge \neg\mathcal{U}$

Algorithm 5. Main Idea of the Distributed Algorithm

Require: parameterised KS $\mathcal{K}$, CTL formula ψ, function f
Ensure: $\mathcal{F}$
  Partition $\mathcal{K}$ into $\mathcal{K}_1, \ldots, \mathcal{K}_n$
  for all $\mathcal{K}_i$ where $i \in \{1, \ldots, n\}$ **do in parallel**
    Take the initial assumption function
    repeat
      repeat
        Compute the new assumptions using the node algorithms (Alg. 1, 2, 3)
        Exchange relevant information with other nodes
      until all processes reach fixpoint
      Modify the assumption function to deal with cycles (Alg. 4)
    until everything is computed

Once the fixpoint is reached, there is additional computation to be made, as there still may be undefined assumptions left. This may happen in the case of the two *until* operators **EU**, **AU**; for more details see [13]. The minimal undefined assumptions are found and set to `ff`, as described in Algorithm 4, and the computation is again restarted. These steps are repeated until a fixpoint is reached and no new assumptions are set in Algorithm 4.

It remains to explain the way of dealing with the initial parameter constraint Φ_I. The initial parameter constraint is orthogonal to the whole computation and we could, in principle, intersect the symbolic true assumptions with Φ_I after the distributed algorithm is finished. However, to prune the search space and speed up the computation somewhat, we intersect the symbolic assumption functions with Φ_I whenever we pass them to the SMT solver (i.e. whenever we need to know whether a symbolic assumption has changed).

Although the node algorithms have been (for clarity) formulated as recomputing everything in each iteration, this is of course unnecessary and we only recompute the part of assumption functions that have been computed as undefined ($\bot$) in the previous iteration. Formally, we restrict the computation of $\widetilde{\mathcal{A}}(s,\psi)$ to $\neg\widetilde{\mathcal{A}}^t_{in}(s,\psi) \wedge \neg\widetilde{\mathcal{A}}^f_{in}(s,\psi)$.

4 Application to Piecewise Multi-affine ODE Models

Let $\mathbb{P} \subseteq \mathbb{R}^m_{\geq 0}$ denote the *continuous parameter space* of dimension m. A parameterised piecewise multi-affine ODE model (PMA) is given by a system of ODEs of the form $\dot{x} = f(x, \mu)$ where $x = (x_1, \ldots, x_n) \in \mathbb{R}^n_{\geq 0}$ is a vector of variables, $\mu = (\mu_1, \ldots, \mu_m) \in \mathbb{P}$ is a vector of parameters, and $f = (f_1, \ldots, f_n)$ is a vector of functions that satisfy the criterion that every f_i is piecewise multi-affine in x and affine in μ.

To approximate the PMA model with its finite quotient represented in terms of a discrete state-transition system, we employ the rectangular abstraction defined in [8] and further adapted in [3,12,26] (see [15] for overview).

We assume there is given a set of thresholds $\{\theta^i_1, \ldots, \theta^i_{n_i}\}$ for each variable x_i satisfying $\theta^i_1 < \theta^i_2 < \cdots < \theta^i_{n_i}$. Each f_i is assumed to be multi-affine on each n-dimensional interval $[\theta^1_{j_1}, \theta^1_{j_1+1}] \times \cdots \times [\theta^n_{j_n}, \theta^n_{j_n+1}]$. We call these intervals rectangles. Each rectangle is uniquely identified via an n-tuple of indices: $R(j_1, \ldots, j_n) = [\theta^1_{j_1}, \theta^1_{j_1+1}] \times \cdots \times [\theta^n_{j_n}, \theta^n_{j_n+1}]$, where the range of each j_i is $\{1, \ldots, n_i - 1\}$. We also define $VR(j_1, \ldots, j_n)$ to be the set of all vertices of $R(j_1, \ldots, j_n)$.

In order to establish a finite rectangular abstraction of the PMA model, special care has to be given to boundary rectangles. A boundary rectangle is any rectangle $R(j_1, ..., j_n)$ where for some i either $j_i = 1$ or $j_i = n_i - 1$. Any dimension i satisfying that condition is called a boundary dimension of $R(j_1, ..., j_n)$. We restrict ourselves to models where the dynamics is bounded in the range specified by lower and upper thresholds – trajectories cannot exit that range (note that this could occur only in boundary rectangles). Formally, all trajectories determined by the PMA model are required to keep $x_i \in [\theta^i_1, \theta^i_{n_i}]$. We restrict ourselves to parameter spaces where this requirement is satisfied for all parameter valuations. More precisely, for every boundary rectangle $R(j_1, ..., j_n)$ we assume that for all $\mu \in \mathbb{P}, i \in \{1, ..., n\}, x \in R(j_1, ..., j_n)$ it holds that $(j_i = 1 \wedge x_i = \theta^i_1) \Rightarrow f_i(x, \mu) > 0$ and $(j_i = n_i - 1 \wedge x_i = \theta^i_{n_i}) \Rightarrow f_i(x, \mu) < 0$.

In [15] it has been shown that rectangular abstraction is conservative with respect to almost all trajectories of the original (continuous) PMA model. In particular, almost every continuous trajectory in the PMA model is covered by a corresponding sequence of rectangles in its rectangular abstraction. However, there may exist a sequence of rectangles for which there is no corresponding continuous trajectory in the original PMA model.

The rectangular abstraction is encoded as a PKS $\mathcal{K} = (\mathbb{P}, S, I, \rightarrow, L)$ with $S = \{(j_1, \ldots, j_n) \mid \forall i : 1 \leq j_i \leq n_i\}$ where each $\alpha \in S$ represents the rectangle $R(\alpha)$. Let now $\alpha = (j_1, \ldots, j_n) \in S$, $1 \leq i \leq n$ and $d \in \{-1, +1\}$. We define $\alpha^{i,d} = (j_1, \ldots, j_i + d, \ldots, j_n)$ (if $j_i + d$ is in the valid range). Thus $\alpha^{i,d}$ describe all the neighbouring rectangles of α. We further define $v^{i,+1}(\alpha) = VR(\alpha) \cap \{(..., j_i + 1, ...)\}$ and $v^{i,-1}(\alpha) = VR(\alpha) \cap \{(..., j_i, ...)\}$. To define the transition relation $\rightarrow$, every pair of states $\alpha, \alpha^{i,d} \in S$, $1 \leq i \leq n$, $d \in \{-1, 1\}$, is associated with a formula $\Phi_{\alpha,\alpha^{i,d}}$ symbolically encoding the set of parameter valuations $\mu \in \mathbb{P}$ for which the transition $\alpha \rightarrow \alpha^{i,d}$ is valid:

$$\Phi_{\alpha,\alpha^{i,d}} := \bigvee_{v \in v^{i,d}(\alpha)} d \cdot f_i(v,\mu) > 0 \tag{2}$$

Additionally, the rectangular abstraction approximates the potential existence of a fixed point in any rectangle $\alpha \in S$. This is achieved by means of introducing a self-transition $\alpha \to \alpha$ [8]. In particular, a self-transition is valid in a state $\alpha \in S$ for all parameter valuations $\mu \in \mathbb{P}$ satisfying $\mathbf{0} \in hull\{f(v,\mu) \mid v \in VR(\alpha)\}$ (the zero vector included in the convex hull of the rectangle vertices). This is symbolically encoded by the formula $\Phi_{\alpha,\alpha}$ defined in the following way:

$$\Phi_{\alpha,\alpha} := \exists c_1,\ldots,c_k : \left(\bigwedge_{i=1}^{k} c_i \geq 0\right) \wedge \left(\sum_{i=1}^{k} c_i = 1\right) \wedge \left(\sum_{i=1}^{k} c_i \cdot f(v_i,\mu) = 0\right) \tag{3}$$

where $k = |VR(\alpha)|$ is the number of vertices of the rectangle α.

To express properties of rectangular abstraction dynamics, the atomic propositions are set to represent concentration inequalities, $AP = \{x_i \odot \theta_j^i \mid 1 \leq i \leq n, 1 \leq j \leq n_i\}, \odot \in \{\leq, \geq\}\}$. States of the PKS are labelled with the adequate constraints of AP. To partition the state space into PKS fragments, we utilise the regular structure of the state space as described in [27]. Note that the PKS constructed by the rectangular abstraction is always total.

5 Experimental Evaluation

We have implemented the distributed algorithm from Sect. 3 in a prototype tool written in Java using the MPJ Express implementation of MPI [2] and the Z3 SMT solver via its Java API [33]. In this section we report on experiments demonstrating scalability and practicability of our approach on case studies of two well-known biological systems.

In order to minimise computational overhead caused by calling Z3 on first-order SMT formulae with quantifiers constructed during the computation, we employ a simplification of abstraction of piecewise multi-affine systems that has been introduced in [3]. In particular, the non-trivial formula (3) representing the convex hull of vectors in rectangle vertices gives a minimal overapproximation of self-transitions by excluding a zero vector from linear combination of rectangle vertices vectors. This formula is replaced with a quantifier-free formula giving a coarser overapproximation:

$$\begin{aligned}\Phi_{\alpha,\alpha} := \neg \bigvee\nolimits_{1 \leq i \leq n} \big(&(\Phi_{\alpha^{i,-1},\alpha} \wedge \Phi_{\alpha,\alpha^{i,+1}} \wedge \neg\Phi_{\alpha,\alpha^{i,-1}} \wedge \neg\Phi_{\alpha^{i,+1},\alpha}) \\ &\vee (\neg\Phi_{\alpha^{i,-1},\alpha} \wedge \neg\Phi_{\alpha,\alpha^{i,+1}} \wedge \Phi_{\alpha,\alpha^{i,-1}} \wedge \Phi_{\alpha^{i,+1},\alpha})\big)\end{aligned}$$

In particular, we exclude self-transitions only in rectangles where there exists a dimension i in which the flow is guaranteed to be one-directional. More specifically, there is either the pair of transitions $\alpha^{i,-1} \to \alpha \to \alpha^{i,+1}$ or the pair of transitions $\alpha^{i,+1} \to \alpha \to \alpha^{i,-1}$ provided that the respective two transitions are the only transitions allowed in ith dimension through the rectangle α. This situation implies that the zero vector is not included in the convex hull of the rectangle vertices (its ith component must be non-zero). The condition is only necessary thus this simplification increases occurrence of spurious self-loops.

5.1 Case Study I: Repressilator

To demonstrate the scalability of the algorithm, we consider a PMA model of the repressilator [12]. It is an approximation of the original model of a genetic regulatory network representing a set of genes mutually inhibited in a closed circle [22].

On this model, we evaluate the scalability of the algorithm from Sect. 3 on a homogeneous cluster with 16 nodes each equipped with 16 GB of RAM and a quad-core Intel Xeon 2 GHz processor. The analysis is provided according to the number of states in combination with one independent and two interdependent parameters, respectively. The considered property is $\mathbf{AG}\,\varphi$ where φ is an atomic proposition specifying a particular subset of states.

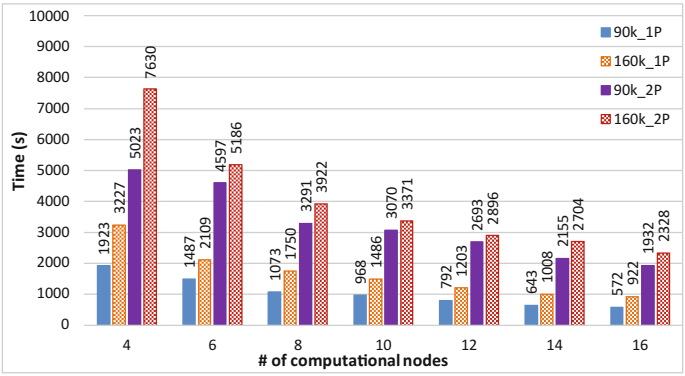
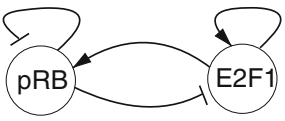


Fig. 1. Scalability achieved for two models approximating the repressilator system: a rough model of the size $90 \cdot 10^3$ states and a refined model of the size $160 \cdot 10^3$ states. Variants 1P represent analyses with a single uncertain parameter whereas variants 2P reflect results achieved for two uncertain mutually dependent parameters. (Color figure online)

5.2 Case Study II: Regulation of G_1/S Cell Cycle Transition

To demonstrate the applicability, we employ our approach on a non-linear ODE model [37] describing a two-gene regulatory network of interactions between the tumour suppressor protein pRB and the central transcription factor $E2F1$ (Fig. 2 (left)). For suitable parameter valuations, two distinct stable attractors may exist (the so-called *bistability*). In [37], the authors have provided numerical bifurcation analysis of $E2F1$ stable concentration depending on the degradation parameter of pRB (ϕ_{pRB}). Note that traditional methods for bifurcation analysis hardly scale to more than a single parameter.

We demonstrate that our algorithm can overcome some of the drawbacks of numerical methods. In particular, we focus on synthesis of values of two interdependent model parameters with respect to satisfaction of the bistability property.



$$\frac{d[pRB]}{dt} = k_1 \frac{[E2F1]}{K_{m1}+[E2F1]} \frac{J_{11}}{J_{11}+[pRB]} - \phi_{pRB}[pRB]$$
$$\frac{d[E2F1]}{dt} = k_p + k_2 \frac{a^2+[E2F1]^2}{K_{m2}^2+[E2F1]^2} \frac{J_{12}}{J_{12}+[pRB]} - \phi_{E2F1}[E2F1]$$
$$a = 0.04,\ k_1 = 1,\ k_2 = 1.6,\ k_p = 0.05,\ \phi_{pRB} = 0.005$$
$$\phi_{E2F1} = 0.1,\ J_{11} = 0.5,\ J_{12} = 5,\ K_{m1} = 0.5,\ K_{m2} = 4$$

Fig. 2. G_1/S transition regulatory network (left) and its ODE model with default values according to [37] (right).

We deal with the degradation parameter ϕ_{pRB} and the production parameter of pRB (k_1). Additionally, we perform post-processing of achieved results by employing additional constraints on the parameter space (i.e., imposing a lower and upper bound on the production/degradation parameter ratio).

The original non-linear model (Fig. 2 (right)) is first converted into a PMA model by employing the approach introduced in [26]. This involves replacement of each non-linear function by an optimal sum of piecewise affine segments (40 segments for pRB and 20 for $E2F1$). Finally, the rectangular abstraction [8] is employed to obtain the PKS for model checking analysis.

The model has been analysed with respect to the properties $\varphi_1 = (\mathbf{AG}\,\mathtt{low})$, $\varphi_2 = (\mathbf{AG}\,\mathtt{high})$ and $\varphi_3 = (\mathbf{EF}\,\mathbf{AG}\,\mathtt{low} \wedge \mathbf{EF}\,\mathbf{AG}\,\mathtt{high})$ where $\mathtt{low} = (E2F1 > 0.5 \wedge E2F1 < 2.5)$ (representing safe cell behaviour) and $\mathtt{high} = (E2F1 > 4 \wedge E2F1 < 7.5)$ (representing excessive cell division). Both properties φ_1 (resp. φ_2) describe the presence and stability of low (resp. high) state and are guaranteed by the rectangular abstraction due to its conservativeness. More specifically, synthesised parameter valuations underapproximate the exact parameter valuation set. Note that φ_1 and φ_2 are subformulae of φ_3, hence all three formulae have been verified in a single run due to the principle of Algorithm 5.

The property φ_3 expresses the possibility of reaching both stable states from a given (initial) state. Such a state thus represents a decision point in the system dynamics. Due to the mixing of existential and universal quantifiers, the property is not preserved by the rectangular abstraction and can thus only be used for estimation (detailed numerical investigation needs to be employed further in the significantly restricted area of the parameter space).

The output of parameter synthesis follows Eq. (1), in particular, we obtain a table of all states satisfying a particular property provided that every state is accompanied with a synthesised constraint on the parameters. Technically, the constraints are given in the SMT-LIB format 2.5 [5]. Consequently, in order to compare and visualise satisfactory parameter valuations in a human-readable form the obtained results have to be further post-processed. The valid area of the parameter space can be visualised by solving the obtained constraints in sampled points. In Fig. 3 (up left), the parameter space with areas constrained by each of the three formulae is depicted (reachability of bistability is shown in *green*; `low` and `high` stable states are shown in *blue* and *red*, respectively).

Additionally, we can employ a static constraint $\Phi_I := \alpha < \frac{k_1}{\phi_{pRB}} < \beta$ to restrict the resulting parameter space to a desired range of the

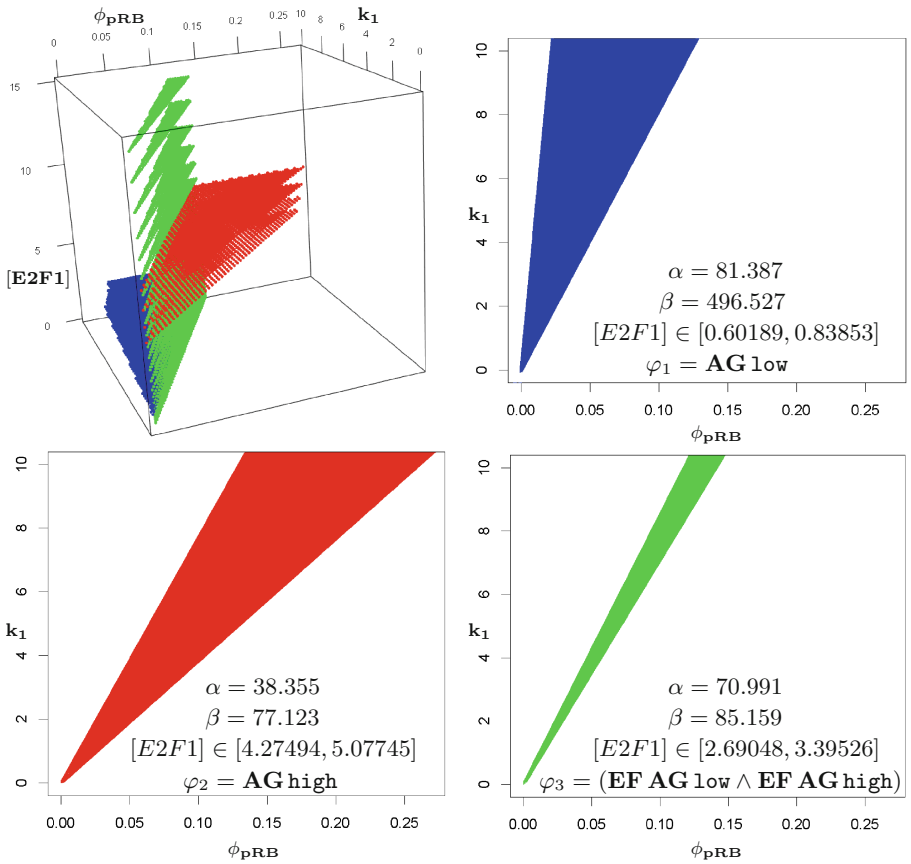


Fig. 3. Parameter space of G_1/S gene regulatory network. Each area meets the respective property: φ_1 (blue), φ_2 (red) and φ_3 (green). (Upper left) Valid parameter spaces sampled for arbitrary initial concentration of $E2F1$ (from 0 to 15 AU). (Other figures) Areas displaying valid ranges of the production/degradation ratio for respective formulae, computed by optimisation. Every figure displays the result for a distinct initial state of $E2F1$. Values of α and β were computed by optimisation. They represent the minimal (α) and maximal (β) ratio $\frac{k1}{\Phi_{pRB}}$ satisfying the particular property.

production/degradation parameters ratio. Moreover, we can use an SMT-based optimisation tool to solve a parametric optimisation problem to find a maximal bound α and a minimal bound β satisfying Φ_I. In our case we employ the tool Symba [29] to compute an over- (resp. under-) approximation of α (resp. β). The achieved ranges of parameters ratio that guarantee the respective formulae are shown in Fig. 3.

6 Conclusion

We have presented a novel parallel algorithm for parameter synthesis on systems with CTL specifications. The method uses a symbolic representation of parameters and employs the satisfiability modulo theories (SMT) approach to deal with first-order formulae that represent sets of parameters. The general description of the algorithm allows it to be used with various families of systems with parameters. In particular, to evaluate the applicability of our algorithm, we have presented a biologically motivated case study.

While evaluating our algorithm we have found the bottleneck to be the large number of calls to the SMT solver. To alleviate this problem somewhat, our implementation uses some optimisation techniques such as query caching and formula simplification. The main simplification relies on the observation that many transition constraints are actually strict subsets of other transition constraints in the model. We plan to explore more of these techniques to reduce both the number and the complexity of the SMT solver calls. We also plan to employ various other SMT solvers, e.g. dReal [23], and compare the efficiency.

References

1. Arney, D., Pajic, M., Goldman, J.M., Lee, I., Mangharam, R., Sokolsky, O.: Toward patient safety in closed-loop medical device systems. In: ICCPS 2010, pp. 139–148. ACM (2010)
2. Baker, M., Carpenter, B., Shafi, A.: MPJ express: towards thread safe Java HPC. In: IEEE Cluster Computing 2006. IEEE Computer Society (2006)
3. Barnat, J., Brim, L., Krejčí, A., Streck, A., Šafránek, D., Vejnár, M., Vejpustek, T.: On parameter synthesis by parallel model checking. IEEE/ACM Trans. Comput. Biol. Bioinf. **9**(3), 693–705 (2012)
4. Barnat, J., Brim, L., Šafránek, D.: High-performance analysis of biological systems dynamics with the divine model checker. Brief. Bioinf. **11**(3), 301–312 (2010)
5. Barrett, C., Fontaine, P., Tinelli, C.: The SMT-LIB standard: version 2.5. Technical report, Department of Computer Science, The University of Iowa (2015)
6. Basu, S., Pollack, R., Roy, M.F.: Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics). Springer-Verlag New York Inc, Secaucus (2006)
7. Batt, G., Page, M., Cantone, I., Gössler, 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)
8. Batt, G., Yordanov, B., Weiss, R., Belta, C.: Robustness analysis and tuning of synthetic gene networks. Bioinformatics **23**(18), 2415–2422 (2007)
9. Bogomolov, S., Schilling, C., Bartocci, E., Batt, G., Kong, H., Grosu, R.: Abstraction-based parameter synthesis for multiaffine systems. In: Piterman, N., et al. (eds.) HVC 2015. LNCS, vol. 9434, pp. 19–35. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26287-1_2
10. Brim, L., Dluhoš, P., Šafránek, D., Vejpustek, T.: STL*: extending signal temporal logic with signal-value freezing operator. Inf. Comput. **236**, 52–67 (2014). Special Issue on Hybrid Systems and Biology

11. Brim, L., Češka, M., Demko, M., Pastva, S., Šafránek, D.: Parameter synthesis by parallel coloured CTL model checking. In: Roux, O., Bourdon, J. (eds.) CMSB 2015. LNCS, vol. 9308, pp. 251–263. Springer, Heidelberg (2015)
12. Brim, L., Demko, M., Pastva, S., Šafránek, D.: High-performance discrete bifurcation analysis for piecewise-affine dynamical systems. In: Abate, A., et al. (eds.) HSB 2015. LNCS, vol. 9271, pp. 58–74. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26916-0_4
13. Brim, L., Yorav, K., Žídková, J.: Assumption-based distribution of CTL model checking. STTT **7**(1), 61–73 (2005)
14. Chiang, H.D., Wang, T.: On the number and types of unstable equilibria in nonlinear dynamical systems with uniformly-bounded stability regions. IEEE Trans. Autom. Control **61**(2), 485–490 (2016)
15. Collins, P., Habets, L.C., van Schuppen, J.H., Černá, I., Fabriková, J., Šafránek, D.: Abstraction of biochemical reaction systems on polytopes. In: IFAC World Congress. pp. 14869–14875. IFAC (2011)
16. Dang, T., Donze, A., Maler, O., Shalev, N.: Sensitive state-space exploration. In: IEEE Conference on Decision and Control, pp. 4049–4054 (2008)
17. Donzé, A., Fanchon, E., Gattepaille, L.M., Maler, O., Tracqui, P.: Robustness analysis and behavior discrimination in enzymatic reaction networks. PLoS ONE **6**(9), e24246 (2011)
18. Donzé, A., Krogh, B., Rajhans, A.: Parameter synthesis for hybrid systems with an application to simulink models. In: Majumdar, R., Tabuada, P. (eds.) HSCC 2009. LNCS, vol. 5469, pp. 165–179. Springer, Heidelberg (2009)
19. Donzé, A., Maler, O., Bartocci, E., Nickovic, D., Grosu, R., Smolka, S.: On temporal logic and signal processing. In: Chakraborty, S., Mukund, M. (eds.) ATVA 2012. LNCS, vol. 7561, pp. 92–106. Springer, Heidelberg (2012)
20. Dreossi, T., Dang, T.: Parameter synthesis for polynomial biological models. In: Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control, HSCC 2014, pp. 233–242 (2014)
21. Dvorak, P., Chrast, L., Nikel, P.I., Fedr, R., Soucek, K., Sedlackova, M., Chaloupkova, R., Lorenzo, V., Prokop, Z., Damborsky, J.: Exacerbation of substrate toxicity by IPTG in Escherichia coli BL21(DE3) carrying a synthetic metabolic pathway. Microb. Cell Fact. **14**(1), 1–15 (2015)
22. Elowitz, M.B., Leibler, S.: A synthetic oscillatory network of transcriptional regulators. Nature **403**(6767), 335–338 (2000)
23. Gao, S., Kong, S., Clarke, E.M.: dReal: an SMT solver for nonlinear theories over the reals. In: Bonacina, M.P. (ed.) CADE 2013. LNCS, vol. 7898, pp. 208–214. Springer, Heidelberg (2013)
24. Gardner, T.S., Cantor, C.R., Collins, J.J.: Construction of a genetic toggle switch in Escherichia coli. Nature **403**, 339–342 (1999)
25. Giacobbe, M., Guet, C.C., Gupta, A., Henzinger, T.A., Paixão, T., Petrov, T.: Model checking gene regulatory networks. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 469–483. Springer, Heidelberg (2015)
26. Grosu, R., Batt, G., Fenton, F.H., Glimm, J., Le Guernic, C., Smolka, S.A., Bartocci, E.: From cardiac cells to genetic regulatory networks. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 396–411. Springer, Heidelberg (2011)
27. Jha, S., Shyamasundar, R.K.: Adapting biochemical kripke structures for distributed model checking. In: Priami, C., Ingólfsdóttir, A., Mishra, B., Riis Nielson, H. (eds.) Transactions on Computational Systems Biology VII. LNCS (LNBI), vol. 4230, pp. 107–122. Springer, Heidelberg (2006)

28. Li, W., Zhong, L., He, Y., Meng, J., Yao, F., Guo, Y., Xu, C.: Multiple steady-states analysis and unstable operating point stabilization in homogeneous azeotropic distillation with intermediate entrainer. Ind. Eng. Chem. Res. **54**(31), 7668–7686 (2015)
29. Li, Y., Albarghouthi, A., Kincaid, Z., Gurfinkel, A., Chechik, M.: Symbolic optimization with SMT solvers. In: POPL 2014, pp. 607–618. ACM (2014)
30. Madsen, C., Shmarov, F., Zuliani, P.: BioPSy: an SMT-based tool for guaranteed parameter set synthesis of biological models. In: Roux, O., Bourdon, J. (eds.) CMSB 2015. LNCS, vol. 9308, pp. 182–194. Springer, Heidelberg (2015)
31. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Lakhnech, Y., Yovine, S. (eds.) FORMATS 2004 and FTRTFT 2004. LNCS, vol. 3253, pp. 152–166. Springer, Heidelberg (2004)
32. Milias-Argeitis, A., Engblom, S., Bauer, P., Khammash, M.: Stochastic focusing coupled with negative feedback enables robust regulation in biochemical reaction networks. J. R. Soc. Interface **12**(113), 20150831 (2015)
33. de Moura, L., Bjørner, N.S.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
34. Raman, V., Donzé, A., Sadigh, D., Murray, R.M., Seshia, S.A.: Reactive synthesis from signal temporal logic specifications. In: HSCC 2015, pp. 239–248. ACM (2015)
35. Rizk, A., Batt, G., Fages, F., Soliman, S.: A general computational method for robustness analysis with applications to synthetic gene networks. Bioinformatics **25**(12), i169–i178 (2009)
36. Rosenfeld, N., Alon, U.: Response delays and the structure of transcription networks. J. Mol. Biol. **329**(4), 645–654 (2003)
37. Swat, M., Kel, A., Herzel, H.: Bifurcation analysis of the regulatory modules of the mammalian G1/S transition. Bioinformatics **20**(10), 1506–1511 (2004)