Parameter Synthesis by Parallel Coloured CTL Model Checking*

Luboš Brim, Milan Češka, Martin Demko, Samuel Pastva and David Šafránek

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

Abstract. We propose a new distributed-memory parallel algorithm for parameter synthesis from CTL hypotheses. The algorithm colours the state space transitions by different parametrisations and extends CTL model checking to identify the maximal set of parameters that guarantee the satisfaction of the given CTL property. We experimentally confirm good scalability of our approach and demonstrate its applicability in the case study of a genetic switch controlling decisions in the cell cycle.

1 Introduction

Constructing computational models that describe dynamics of biochemical processes is a key step towards understanding of existing and even yet undiscovered behavioural and physiological phenotypes occurring in biology. Model-based prediction and analysis make cornerstones of systems biology. While the structure of dynamical models of some biochemical processes is already available at the qualitative level represented by known entities and interactions, most of the quantitative aspects of the systems dynamics, such as reaction rates or initial concentration values, cannot be easily determined. Such quantitative attributes are usually reflected in the model as *parameters*. In order to obtain reliable models, parameters need to be specified exactly. For a typical model, a fraction of the parameter values can be determined from the literature or experimental data, leaving many parameter values uncertain or completely unknown. The reason is, that many parameters are hard to measure *in vitro*/*in vivo*.

The *algorithmic discovery* of unknown parameter values (also referred to as *parameter estimation*, *parameter identification*, the *inverse problem*, or *model calibration*) remains thus one of the main challenges in computational systems biology. Besides the traditional approaches to tackle the inverse problem (e.g., [15–17, 24]), there have recently appeared alternative techniques grounded in formal verification [2, 4, 21]. These methods typically focus on identifying reliable subsets of parameter space instead of finding singular parameter values.

Hypotheses mined from biological literature as well as time-series experiments from wet-labs can be considered as dynamical constraints restricting the admissible set of model parameter values. Apart from a concrete kind of dynamical models, these constraints can be sufficiently captured in terms of temporal

* This work has been supported by the Czech Science Foundation grant 11089S and the Czech Ministry of Education, Youth, and Sport project No. CZ.1.07/2.3.00/30.0009.

logic formulae (for review of approaches see, e.g,. [7]). A common computational method that decides the question whether for a given parametrisation the model meets the temporal constraints is *model checking*. The inverse problem is then generalised to *parameter synthesis* [2, 12] – to find the maximal set of parameter values from the given set, such that they meet the stated dynamical constraints.

The general advantage of temporal specification for parameter synthesis is its ability to focus on certain qualitative aspects of observed behaviour [23] (e.g., temporal ordering of events qualitatively characterising important moments in the systems dynamics). In particular, temporal properties can be viewed as *global properties* independent of particular setting of initial conditions (initial values of the state variables). The global view provides biologists a tool which, for a given model and a given property, computes the maximal set of parameter values and initial conditions for which the model entirely fulfils the property. Such an approach is complementary to traditional approaches based on monitoring a numerical simulation [11, 25] or local sensitivity analysis [13].

To capture biologically-relevant temporal hypotheses both branching-time operators and linear-time operators are needed [6]. In this paper we focus on *branching logic* CTL. The reason is that many relevant questions in systems biology need branching operators to express them properly. For instance, switching mechanisms and multi-stability are present in genetic regulatory networks and drive many key biological phenotypes such as, e.g., irreversible decisions in cell division, cell differentiation or programmed cell death. However, it is difficult (or often impossible) to express relevant properties in linear temporal logics. Other reason for usage of CTL is related to the particular procedure for model checking. This procedure allows to effectively identify all system states where the given property is satisfied. Thus CTL procedure leads inherently to global analysis of systems dynamics as opposed to LTL procedure, which requires a single initial state (or iterates over a given set of initial states).

Contribution of the Paper. Several methods for parameter synthesis based on model checking have been proposed recently, targeting different kinds of models and different temporal logics (e.g., [2, 5, 11, 12, 19]). In [2] we proposed a parameter synthesis method for LTL hypotheses established on our automata-based *colored* LTL *model checking* algorithm.

In this paper we extend that work in several directions. First, we consider CTL hypotheses. Second, we propose a distributed-memory parallel colored CTL *model checking* algorithm, keeping thus both the advantage of having an explicit representation and the effectiveness of parallel solution in distributed-memory. Third, we propose a novel *heuristics for partitioning the state space* that effectively uses specifics of rectangularly abstracted ODE models (the abstraction is described in [10]). We have experimentally confirmed good scalability of our approach and demonstrated its applicability in a case study of a genetic switch employing rectangular abstraction [4, 10] of an already existing ODE model [26].

2 Parallel Parameter Synthesis Algorithm

In this paper we propose a formal framework for parameter synthesis of biochemical models from branching time temporal logic formulae. Here, the term parameter refers to both the initial conditions of the model and to dynamical parameters. The method presumes a finite state space. For discrete models such as boolean networks, this can be ensured directly by the definition. For continuous models, like ODE models, a finite discrete abstraction of the state space is necessary. The existing abstractions typically lead to over- or under-approximation (or a mixture of both) of the dynamics of the original system [10]. This has generally some consequences regarding the interpretation of computed results. We will discuss this issue later. The method also presumes a finite parameter space. In the case of continuous parameter spaces an appropriate finite abstraction, like an interval abstraction in the case of ODE models, must be used.

It is important to note that there are two 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 the parameter space grows exponentially with the number of unknown parameters. However, in reality the number of parameters to be considered should be small. A model with too many parameters is hard to falsify - it can fit almost any data. 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.

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 [1, 7]. 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 [5, 14] whereas enumerative parallel techniques have proved to be fruitful for quantitative models [1, 3].

Coloured CTL Model Checking

We start by introducing the notion of a parametrised Kripke structure that encapsulates a family of Kripke structures built over the same model but with different valuations of individual parameters.

Let AP be a set of atomic propositions. A *parametrised Kripke structure* (over AP) is a tuple $\mathcal{K} = (\mathcal{P}, S, I, \rightarrow, L)$, where $\mathcal{P}$ denotes the finite *set of parameter values (parametrisations)*, i.e., all the possible valuations of the parameters, S the finite set of states, $I \subseteq S$ the set of initial states, $L : S \rightarrow 2^{\mathrm{AP}}$ is a labelling of states by atomic propositions, $\rightarrow \subseteq S \times \mathcal{P} \times S$ is a transition relation labelled by parameter valuations (not required to be total). We write

$s \xrightarrow{p} s'$ instead of $(s, p, s') \in \rightarrow$. Fixing a parametrisation $p \in \mathcal{P}$ reduces the parametrised Kripke structure $\mathcal{K}$ to the standard (non-parametrised) Kripke structure $\mathcal{K}(p) = (S, I, \xrightarrow{p}, L)$.

To express properties (hypotheses) about the dynamics of systems, we consider formulae of CTL defined by the following abstract syntax:

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

where Q ranges over *atomic propositions* taken from a set AP. Let φ be a CTL formula. We denote by $cl(\varphi)$ the set of all subformulae of φ and by $tcl(\varphi)$ the set of all (temporal) subformulae of φ of the form $\mathbf{EX}\varphi$, $\mathbf{E}(\varphi_1 \,\mathbf{U}\, \varphi_2)$, $\mathbf{AX}\varphi$ or $\mathbf{A}(\varphi_1 \,\mathbf{U}\, \varphi_2)$. We use the standard abbreviations like $\mathbf{EF}\varphi$ which stands for $\mathbf{E}(true \,\mathbf{U}\, \varphi)$ or $\mathbf{AG}\varphi$ which stands for $\neg\mathbf{E}\,\mathbf{F}\,\neg\varphi$. Examples of some typical CTL formulae are [14]:

- $\mathbf{EF}\ \varphi$ expresses a reachability of a state where the condition φ holds,
- $\mathbf{AG}\ \varphi$ expresses a stabilisation with φ being continually true,
- $\mathbf{EFAG}\varphi_1 \wedge \mathbf{EFAG}\varphi_2$ expresses a bistable switch (two different stable situations φ_1, φ_2 can be reached).

Most frequent types of temporal properties investigated for biochemical models have been collected in [23]. There are two important fragments of CTL relevant for biological models. A formula is said to be *positive* if it does not contain any negations. We say that a formula is *existential* (or in ECTL) if it is positive and only contains existential temporal operators. We say that a formula is *universal* (or in ACTL) if it is positive and only contains universal temporal operators.

It is important to note, that model abstraction based on over-approximation preserves truth of universally-quantified CTL properties (ACTL), i.e. if an ACTL property holds in the abstract model, it is guaranteed to hold in the concrete one. Dually, under-approximation preserves falsity of ACTL. The situation is reversed for existentially-quantified CTL properties (ECTL): over-approximation preserves falsity while under-approximation preserves truth.

The parameter synthesis problem is defined in the following way. Suppose we are given a parametrised Kripke structure $\mathcal{K}$ and a CTL formula Ψ. For each state $s \in S$ let $P_s = \{p \in \mathcal{P} \mid s \models_{\mathcal{K}(p)} \Psi\}$, where $s \models_{\mathcal{K}(p)} \Psi$ denotes that Ψ is satisfied in the state s of $\mathcal{K}(p)$. The *parameter synthesis problem* requires to compute the function $\mathcal{F}^{\mathcal{K}}_{\Psi} : S \to 2^P$ such that $\mathcal{F}^{\mathcal{K}}_{\Psi}(s) = P_s$. Often we are especially interested in computing the set $\cap_{s \in I} \mathcal{F}^{\mathcal{K}}_{\Psi}(s)$.

The algorithm for computing $\mathcal{F}^{\mathcal{K}}_{\Psi}$ is a modification of the (explicit) labelling CTL model checking algorithm [9]. It labels states with "coloured" subformulae of Ψ that are satisfied in the state of the Kripke structure $\mathcal{K}(p)$ for the "colour" $p \in \mathcal{P}$. Typically the structures $\mathcal{K}(p)$ have similar transition relations, thus leading to a significant acceleration of the parameter synthesis.

The algorithm operates recursively on the structure of Ψ starting from atomic propositions. Its basic idea is described by the Algorithm 1. The recursive computation of the satisfaction sets $ColSat(\Psi) = \{(p, s) \in \mathcal{P} \times S \mid s \models_{\mathcal{K}(p)} \Psi\}$ follows the parse tree of the formula Ψ.

Algorithm 1 Compute parameters

Require: parametrised KS $\mathcal{K}$ and CTL formula Ψ
Ensure: $\mathcal{F}_\Psi^\mathcal{K}$
  for all $i \leq |\Psi|$ **do** ▷ compute the sets $ColSat(\Phi) = \{(p,s) \in \mathcal{P} \times S \mid s \models_{\mathcal{K}(p)} \Phi\}$
    for all $\Phi \in cl(\Psi)$ *with* $|\Phi| = i$ **do**
      compute $ColSat(\Phi)$ from $ColSat(\Phi')$ ▷ for maximal genuine $\Phi' \in cl(\Phi)$
  return $\{(p,s) \in \mathcal{P} \times S \mid (p,s) \in ColSat(\Psi)\}$

Kripke Fragments

Our aim is to perform the parameter synthesis algorithm as a distributed-memory algorithm on a cluster of n nodes (workstations) in order to enlarge the available memory to accommodate larger models. To this end we use a *partition function* $f : S \to \{1, \ldots, n\}$ to partition the state space among n nodes. After partitioning, each node owns a part of the original state space. Concrete techniques for the state space partitioning are discussed in the next subsection.

We adapt the assumption based distributed CTL model checking paradigm [8] as the basis of our work. We represent the state space owned by one node using a parametrised Kripke structure with *border states* (also called a *fragment*). Intuitively, border states, that are added to the states assigned by f, are states that in fact belong to other station and represent the missing parts of the state space (placed in the memory of other nodes and not directly accessible). For structure $\mathcal{K}$, the set of its border states is defined as $border(\mathcal{K}) = \{s \in S \mid \neg\exists(p, s').s \xrightarrow{p} s'\}$. A *fragment* $\mathcal{K}_i$ of $\mathcal{K}$ is a substructure of $\mathcal{K}$ 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 $\mathcal{K}$. Partitioning the given structure $\mathcal{K}$ results in a finite set $\mathcal{K}_1, \ldots, \mathcal{K}_n$ of fragments each handled by one node. A border state is thus stored several times: as original one on the node that owns it and as duplicates on nodes they own its predecessors.

To define the semantics of CTL formulae over fragments we need to adapt the standard semantic definition. We define the notion of the truth under assumptions associated with border states. An *assumption function* for a parametrised 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 parametrisation p, $\mathcal{A}(p, s, \varphi) = \texttt{ff}$ if we can assume that φ does not hold in the state s under parametrisation p, and $\mathcal{A}(p, s, \varphi) = \perp$ if we cannot assume anything. Let us denote by $AS_\mathcal{K}^\psi$ the set of all assumption functions for a formula ψ and a parametrised Kripke structure $\mathcal{K}$

We consider a new semantic function $\mathcal{C}_\mathcal{K}^\psi : AS_\mathcal{K}^\psi \to AS_\mathcal{K}^\psi$ that takes an input assumption function $\mathcal{A}_{in}$ and returns a new assumption function $\mathcal{A}$. If $s \in border(\mathcal{K})$ and $\varphi \in |tcl|(\psi)$ then $\mathcal{A}(p, s, \varphi) = \mathcal{A}_{in}(p, s, \varphi)$. If $s \notin border(\mathcal{K})$ and $\varphi \in |tcl|(\psi)$ then $\mathcal{A}(p, s, \varphi)$ is defined recursively. We provide here only the

definition for the most complicated case of $\mathbf{A}(\varphi_1 \,\mathbf{U}\, \varphi_2)$ (the full definition is in Appendix A). $\mathcal{A}(p, s, \mathbf{A}(\varphi_1 \,\mathbf{U}\, \varphi_2)) =$

$$\begin{cases} \mathtt{tt} & \text{if for all p-paths } \pi = s_0 s_1 s_2 \ldots \text{ with } s = s_0 \text{ there exists an index} \\ & x < |\pi| \text{ such that: either } \mathcal{A}(p, s_x, \varphi_2) = \mathtt{tt} \text{ or } [s_x \in \mathit{border}(\mathcal{K}) \text{ and} \\ & \mathcal{A}(p, s_x, \mathbf{A}(\varphi_1 \,\mathbf{U}\, \varphi_2)) = \mathtt{tt})], \text{ and } \forall y : 0 \leq y < x : \mathcal{A}(p, s_y, \varphi_1) = \mathtt{tt} \\ \mathtt{ff} & \text{if there exist a p-path } \pi = s_0 s_1 s_2 \ldots \text{ with } s = s_0 \text{ and an index} \\ & x < |\pi| \text{ such that: } [\mathcal{A}(p, s_x, \varphi_1) = \mathtt{ff} \text{ and } \forall y \leq x : \mathcal{A}(p, s_y, \varphi_2) = \mathtt{ff}] \\ & \text{or } \forall x < |\pi| : [\mathcal{A}(p, s_x, \varphi_2) = \mathtt{ff} \text{ and } (|\pi| = \infty \text{ or } (s_{|\pi|-1} \in \mathit{border}(\mathcal{K}) \\ & \text{and } \mathcal{A}(p, s_{|\pi|-1}, \mathbf{A}(\varphi_1 \,\mathbf{U}\, \varphi_2)) = \mathtt{ff}))] \\ \bot & \text{otherwise} \end{cases}$$

Here a p-path π from a state s_0 is a sequence $\pi = s_0 s_1 \ldots$ such that $\forall i \geq 0 : s_i \in S$ and $s_i \xrightarrow{p} s_{i+1}$. The truth of a formula is relative to given assumptions $\mathcal{A}_{in}$ and it is defined as $\mathcal{C}^{\psi}_{\mathcal{K}}(\mathcal{A}_{in})(p, s, \psi)$. The value of an assumption function $\mathcal{A}_{in}(p, s, \varphi)$ for a state $s \notin \mathit{border}(\mathcal{K})$ does not influence the value $\mathcal{C}^{\psi}_{\mathcal{K}}(\mathcal{A}_{in})$. Hence, for any *total* parametrised Kripke structure $\mathcal{K}$ (i.e. $\mathit{border}(\mathcal{K}) = \emptyset$), CTL formula ψ and an arbitrary assumption function $\mathcal{A} \in AS^{\psi}_{\mathcal{K}}$, we have that $s \models_{\mathcal{K}(p)} \psi \Leftrightarrow \mathcal{C}^{\psi}_{\mathcal{K}}(\mathcal{A})(p, s, \psi) = \mathtt{tt}$. In particular, $\mathit{ColSat}(\psi) = \{(p, s) \in \mathcal{P} \times S \mid \mathcal{C}^{\psi}_{\mathcal{K}}(\mathcal{A})(p, s, \psi) = \mathtt{tt}\}$ and thus we can solve the parameter synthesis problem by computing the assumption function $\mathcal{C}_{\mathcal{K}}(\mathcal{A}_{\bot})$.

Distributed Algorithm

We are now ready to describe the algorithm for distributed parameter synthesis. In order to compute $\mathcal{C}_{\mathcal{K}}(\mathcal{A}_{\bot})$ in a 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 f. Each node performs Algorithm 1 modified in such way, that it is also able to cope with "undefined values". Moreover, it computes both the positive and negative results. This means that if a state s has a successor for which φ is true for parametrisation p, it can be concluded both that s satisfies $\mathbf{EX}\varphi$ and that s does not satisfy $\mathbf{AX}\neg\varphi$ under p, even when the validity of φ in other successors of s is undefined (unknown) yet.

The main idea of the entire distributed computation, summarized in Algorithm 2, 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 initializes the assumption function $\mathcal{A}_i$ to the undefined assumption function $\mathcal{A}_{\bot}$. After initialization, it computes the semantic function $\mathcal{C}_{\mathcal{K}_i}(\mathcal{A}_i)$ using the node algorithm (the algorithm is given in Appendix B). If new assumptions have been computed for some border states, this result is sent directly to appropriate processes. Similarly, if such information is received from another process, the assumption function is modified to reflect these new results. This procedure is repeated until all running processes are "deadlocked", i.e. until no new information (value of an assumption function) can be computed using the node algorithm or by exchanging assumptions among processes. We say that the

Algorithm 2 Main Idea of the Distributed Algorithm

Require: parametrised KS $\mathcal{K}$, CTL formula Ψ, function f
Ensure: $\mathcal{F}_{\Psi}^{\mathcal{K}}$
  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
      Compute the semantic function using the node algorithm;
      Exchange relevant information with other nodes;
      Modify assumption function;
    until all processes reach fixpoint

fixpoint has been reached ("global" stabilization has occurred). In our experimental implementation, the deadlock is detected by additional communication among processes (the code has been skipped for clarity).

After stabilization (reaching the fixpoint) there may still remain a state s and a formula φ, for which $\mathcal{A}_i(s, \varphi) = \perp$. This can happen in the case of the **U** operator. However, if the results for all subformulas of φ have already been computed in all states on all nodes and the fixpoint has been reached then we can conclude that φ does not hold in s.

State Space Partitioning

The key ingredient of distributed model checking algorithms is a suitable state space partitioning that minimizes the communication overhead and equally distributes the workload. In particular, the partitioning should provide 1) a regular *load-balancing* ensuring that each node is responsible for a proportional part of the state space and 2) a good *locality* minimising the number of cross transitions where the source and target states are assigned to two different nodes.

The computation of the optimal partitioning for the given state space typically brings a significant overhead and thus various heuristics are considered. For computer and engineering systems, a hash-based partitioning is usually used, since it does not require any prior knowledge about the structure of the state space. It constructs a hash function mapping each state to a node. This approach usually provides very good load-balancing following from an uniformity of the hash function. However, these heuristics are not able to control the locality and thus they introduce a considerable communication overhead.

In our approach we utilize the regular structure of the state space for biochemical models [20]. We use structural properties of the rectangular abstraction of the given parametric piece-wise multi-affine ODE model [4, 10]. The approximation is formed by an n-dimensional hyper-rectangular state space defined by m state variables and by a set of thresholds for each variable. The partitioning decomposes the state space into n hyper-rectangular subspaces

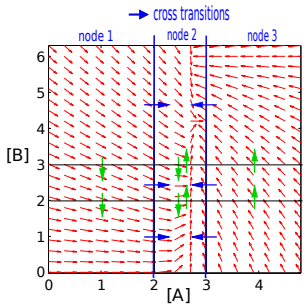


Fig. 1. State space partitioning.

(n is the number of nodes) such that each subspace has similar volume. Figure 1 depicts such partitioning for $m = 2$ and $n = 3$ where the volume for each subspace is 3. Our heuristic usually provides a good load balancing, since the volume reflects the number of states. The construction of the discretised state space further ensures there are only transitions between the adjacent states with respect to the hyper-rectangular structure. Therefore, our partitioning naturally provides almost the minimal number of cross transitions, since only cross transitions between the border states are introduced as illustrated in Figure 1. Comparing to the hash-based partitioning we significantly decrease the communication overhead. Note that, the final load balancing can be negatively affected by the backward connectivity of the state space. However, our experiments demonstrate the connectivity is significantly increased due to the fact that we have to consider all parametrisations of the model. Additional heuristics are used to improve the load balancing by reflecting the atomic propositions in the CTL formula.

3 Experimental Evaluation

We first consider a suitable model that enables us to thoroughly evaluate the scalability of the proposed distributed algorithm. Afterwards, we apply our approach to a relevant and interesting model describing the regulation in a cell cycle transition.

Scalability

The scalability of the algorithm is evaluated on a catalytic reaction model (Appendix C). The model allows to scale the number of intermediate products/variables (N), discretisation thresholds (T) and unknown parameters. For each variable we assume a same number of thresholds and thus the total number of states is $(T-1)^N$. We employ the state space partitioning that reflects the model structure and thus it provides a good load-balancing and locality.

We use homogeneous cluster with 12 nodes each equipped with 16 GB of RAM and a quad-core Intel Xeon 2 GHz processor. In order to provide a fair evaluation we utilize only a single core on each node (although our implementation can effectively utilize multi-core nodes). The reported runtime has been obtained as the arithmetic mean from several experiments.

Fig. 2 illustrates the results for $N = 6$ and $T = 13$ (i.e. almost 3 millions states) and different number of unknown parameters. The figure demonstrates a significant acceleration of the parameter synthesis when more nodes are used. Note that the missing columns indicate that the corresponding experiment run out of memory. The number of unknown parameters changes the structure of the state space and its partitioning. Therefore, in some cases, a higher number of parameters can decrease the runtime.

We have further evaluated the scalability of the algorithm with respect to the increasing number of variables and thresholds. As demonstrated in Appendix C,

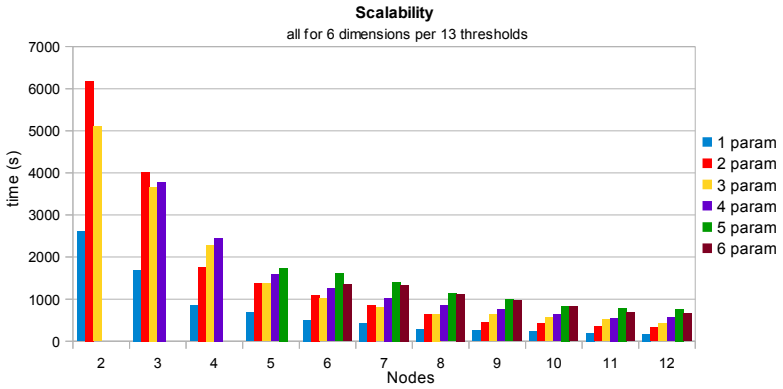
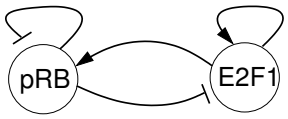


Fig. 2. The scalability with respect to the number of unknown parameters.

the higher number of nodes again considerably accelerates the computation and allows us to synthesize the parameters for larger models. Note that for larger state spaces the increasing number of nodes leads almost to the linear speedup.

Case study: regulation of G_1/S cell cycle transition

To demonstrate applicability of our framework, we investigate a well-known ODE model [26] representing a two-gene regulatory network describing interaction of the tumor suppressor protein pRB and the central transcription factor $E2F1$ (see Fig. 3 (left)). This network represents the crucial mechanism governing the transition from G_1 to S phase in the mammalian cell cycle. In the G_1-phase the cell makes an important decision. In high concentration levels, $E2F1$ activates the G_1/S transition mechanism. In low concentration of $E2F1$, committing to S-phase is refused and that way the cell avoids DNA replication.



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

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

Fig. 3. G_1/S transition regulatory network and its ODE model taken from [26].

The mechanism is an example of a *bistable switch*, an irreversible decision to finally reach some of the two different stable states. In particular, we are interested in the existence of two different stable equilibria on $E2F1$. Activity of pRB is rapidly modulated by phosporylation/dephosporylation turn-over controlled by growth factor signals transferred to cyclin-dependent kinases each acting on a specific subset of pRB phosporylation sites [22]. This control is captured in the model by means of the degradation rate parameter ϕ_{pRB}.

In [26] the authors have provided bifucartion analysis investigating $E2F1$ equilibria depending on ϕ_{pRB}. As shown in Fig. 4(left), by non-trivial elaboration with numerical analysis methods expecting the previous knowledge of the equilibria they constructed equilibrium point curve for $E2F1$ in proportion to ϕ_{pRB} and discovered two saddle-node bifurcation points. For ϕ_{pRB} smaller

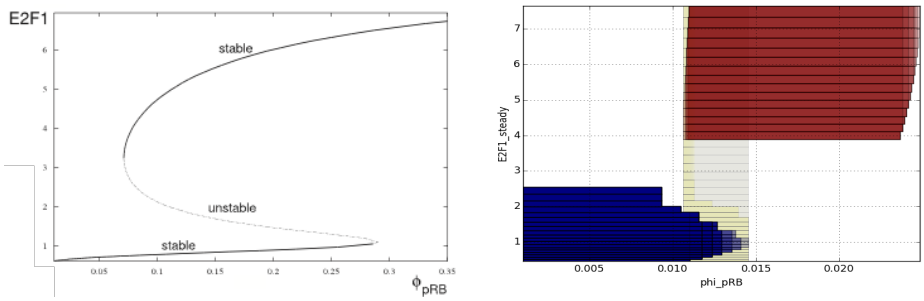


Fig. 4. (left) Equilibrium point curve taken from [26] (we believe there is a typo in the scale of ϕ_{pRB} in the original figure, the range of ϕ_{pRB} should read 0.005-0.035). (right) Model checking results. Red and blue correspond to the high and low stable regions, respectively. Yellow are the states where the *bistable switch* formula φ holds.

then 0.007 the system converges to a single low-concentration stable equilibrium whereas for values higher than 0.027 it converges to a single high-concentration equilibrium. In between the two bifurcation points the system is bistable provided that there always exists an unstable equilibrium for which there is an ϵ-ball that makes a basin of attraction for both stable equilibria.

To employ our framework for this non-linear model, we have first created the piece-wise multi-affine approximation (PMA) of the ODE model [18]. We approximate each non-linear function in the right-hand side of ODEs with an optimal sequence of piece-wise affine ramp functions (in our case we have set the precision to 70 affine segments per each non-linear function). For the resulting PMA we have employed rectangular abstraction [4] to obtain a finite (rectangular) automaton over-approximating the PMA (the intuition is shown in Fig. 1). Finally, we have run the parallel coloured CTL model checking algorithm for the formula $\varphi \equiv EFAG\ \texttt{high} \wedge EFAG\ \texttt{low}$ and the initial parameter space $\phi_{pRB} \in [0.001, 0.025]$. The atomic propositions `low` and `high` characterise the location of expected regions of $E2F1$ stability. Based on the results reported in [26] we define the stable regions as $\texttt{high} \equiv (E2F1 > 4 \wedge E2F1 < 7.5)$ and $\texttt{low} \equiv (E2F1 > 0.5 \wedge E2F1 < 2.5)$ that determine the expected regions of the two stable attractors including (a subset of) their surrounding attracted points. Details of individual steps are described in Appendix D.

Results of the analysis are depicted in Fig. 4(right) in comparison with the equilibrium curve, Fig. 4(left), provided in [26]. The blue region is the place where $AG\ \texttt{low}$ is satisfied, in particular, it says the $E2F1$ low concentration is *guaranteed* to stabilise for the corresponding values of ϕ_{pRB} in the PMA. The guarantee comes from the fact that the abstraction employed is over-approximation [10]. In particular, for each trajectory in the PMA there must exist a corresponding path in the rectangular automaton. For example, the model checking result says that for a fixed parameter value 0.005 there is no path in the rectangular automaton that would exit the concentration bounds $0.5 \leq E2F1 \leq 2.5$ and hence there is no such trajectory in the PMA. However, although there is no red region identified at $\phi_{pRB} = 0.005$ we are not sure this holds also in the PMA since it might be the property introduced by the abstraction. For a given

ACTL formula, the abstraction thus causes the parameter space synthesised by model checking to be under-approximated [4]. For example, with ϕ_{pRB} getting closer to the bistable region the guarantee of low stabilisation becomes limited to a smaller subset of the `low` region until it disappears at $\phi_{pRB} > 0.0145$. The analogous explanation fits the red region obtained for AG `high`, note that in that case the effect of parameter value under-approximation is negligible when compared with equilibrium point curve. For $\phi_{pRB} \in [0.012, 0.0145]$, the system is bistable (there exist two stable regions, i.e., AG `low` $\wedge AG$ `high` is guaranteed).

The yellow region covers points where φ holds. Since an EF-formula might be satisfied within a spurious behaviour introduced by the abstraction, this result does not provide any guarantees but rather estimates parameter values and initial conditions under which both stable regions might be reached. The diagram projects pRB values by means of fill opacity. Grey region reflects the fact there are values of pRB from which the red or the blue region is not reachable. This information is again guaranteed (3D visualisation and further interpretation of results are provided in Appendix D).

4 Conclusions

We have developed a fully automatic method for synthesizing parameters that guarantee the satisfaction of a given CTL hypothesis. The method uses a novel distributed-memory parallel algorithm that extends the CTL model-checking algorithm by colouring the transitions in the underlying state space. We have demonstrated a very good scalability of the algorithm as well as the usefulness of the method on a biological problem of bistable switch. This is an example of a wide range of possible applications. The case study can be compared to numerical bifurcation analysis methods that require good initial estimate of the equilibria and do not scale up with the number of unknown parameters. Our method does not require so detailed initial knowledge about the system and scales well with the number of unknown parameters and system dimensionality.

References

1. Ballarini, P., Guido, R., Mazza, T., Prandi, D.: Taming the complexity of biological pathways through parallel computing. Brief. Bioinform 10(3), 278–288 (2009)
2. Barnat, J., Brim, L., Krejci, A., Streck, A., Safranek, D., Vejnar, M., Vejpustek, T.: On Parameter Synthesis by Parallel Model Checking. IEEE/ACM Transactions on Computational Biology and Bioinformatics 9(3), 693–705 (2012)
3. Barnat, J., Brim, L., Safránek, D.: High-performance analysis of biological systems dynamics with the divine model checker. Brief. Bioinform 11(3), 301–312 (2010)
4. Batt, G., Belta, C., Weiss, R.: Model checking liveness properties of genetic regulatory networks. In: TACAS. LNCS, vol. 4424, pp. 323–338. Springer (2007)
5. 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)
6. Batt, G., Ropers, D., Jong, H.D., Geiselmann, J., Mateescu, R., Schneider, D.: Validation of qualitative models of genetic regulatory networks by model checking: Analysis of the nutritional stress response in escherichia coli. Bioinformatics 21, 19–28 (2005)

7. Brim, L., Češka, M., Šafránek, D.: Model checking of biological system. In: SFM. LNCS, vol. 7938, pp. 63–112 (2013)
8. Brim, L., Yorav, K., Zidkova, J.: Assumption-based distribution of CTL model checking. STTT 7(1), 61–73 (2005)
9. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. 8, 244–263 (1986)
10. 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)
11. Donaldson, R., Gilbert, D.: A model checking approach to the parameter estimation of biochemical pathways. In: CMSB, LNCS, vol. 5307, pp. 269–287. Springer (2008)
12. Donzé, A., Clermont, G., Langmead, C.J.: Parameter synthesis in nonlinear dynamical systems: Application to systems biology. J. Comput. Biol. 17(3), 325–336 (2010)
13. 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)
14. Fages, F., Soliman, S.: Formal cell biology in biocham. In: SFM. LNCS, vol. 5016, pp. 54–80. Springer (2008)
15. Fröhlich, F., Theis, F., Hasenauer, J.: Uncertainty analysis for non-identifiable dynamical systems: Profile likelihoods, bootstrapping and more. In: CMBS, LNCS, vol. 8859, pp. 61–72. Springer (2014)
16. Gábor, A., Banga, J.R.: Improved parameter estimation in kinetic models: Selection and tuning of regularization methods. In: CMSB, LNCS, vol. 8859, pp. 45–60. Springer (2014)
17. Gilbert, D., Breitling, R., Heiner, M., Donaldson, R.: An introduction to biomodel engineering, illustrated for signal transduction pathways. In: Membrane Computing, LNCS, vol. 5391, pp. 13–28. Springer (2009)
18. Grosu, R., Batt, G., Fenton, F.H., Glimm, J., Guernic, C.L., Smolka, S.A., Bartocci, E.: From cardiac cells to genetic regulatory networks. In: CAV. LNCS, vol. 6806, pp. 396–411 (2011)
19. Jha, S.K., Langmead, C.J.: Synthesis and infeasibility analysis for stochastic models of biochemical systems using statistical model checking and abstraction refinement. Theoretical Computer Science 412(21), 2162 – 2187 (2011)
20. Jha, S., Shyamasundar, R.: Adapting biochemical kripke structures for distributed model checking. In: Transactions on Computational Systems Biology (TCSB) VII, LNBI, vol. 4230, pp. 107–122. Springer (2006)
21. Liu, B., Kong, S., Gao, S., Zuliani, P., Clarke, E.M.: Parameter synthesis for cardiac cell hybrid models using δ-decisions. In: CMSB, LNCS, vol. 8859, pp. 99–113. Springer (2014)
22. Mittnacht, S.: Control of prb phosphorylation. Current Opinion in Genetics & Development 8(1), 21 – 27 (1998)
23. Monteiro, P.T., Ropers, D., Mateescu, R., Freitas, A.T., de Jong, H.: Temporal Logic Patterns for Querying Qualitative Models of Genetic Regulatory Networks. In: ECAI. FAIA, vol. 178, pp. 229–233. IOS Press (2008)
24. Raue, A., Karlsson, J., Saccomani, M.P., Jirstrand, M., Timmer, J.: Comparison of approaches for parameter identifiability analysis of biological systems. Bioinformatics (2014)

25. Rizk, A., Batt, G., Fages, F., Soliman, S.: A general computational method for robustness analysis with applications to synthetic gene networks. Bioinformatics 25(12) (2009)
26. Swat, M., Kel, A., Herzel, H.: Bifurcation analysis of the regulatory modules of the mammalian G1/S transition. Bioinformatics 20(10), 1506–1511 (2004)

Appendix A: Assumption semantics for CTL

A p-path π in a parametrised Kripke structure $\mathcal{K}$ from a state s_0 is a sequence $\pi = s_0 s_1 \ldots$ such that $\forall i \geq 0 : s_i \in S$ and $s_i \xrightarrow{p} s_{i+1}$. A maximal p-path is a p-paths that is either infinite or ends in a border state. For a maximal p-path we denote by $|\pi|$ the length of the path. In case the p-path is infinite we put $|\pi| = \infty$.

Definition 1. *Let $\mathcal{K} = (\mathcal{P}, S, I, \rightarrow, L)$ be a parametrised Kripke structure where $L : AP \rightarrow 2^S$ is* valuation *assigning to each atomic proposition a set of states and ψ be a CTL formula. We define the function $\mathcal{C}^{\psi}_{\mathcal{K}} : AS^{\psi}_{\mathcal{K}} \rightarrow AS^{\psi}_{\mathcal{K}}$. For $\mathcal{A}_{in} \in AS^{\psi}_{\mathcal{K}}$ let $\mathcal{A} = \mathcal{C}^{\psi}_{\mathcal{K}}(\mathcal{A}_{in})$. Then the assumption function $\mathcal{A}$ is defined inductively in the following way (the definition for $\mathcal{A}(p, s, \boldsymbol{A}(\varphi_1 \, \boldsymbol{U} \varphi_2))$ is given in the paper):*

If $s \in \mathit{border}(\mathcal{K})$ and $\varphi \in |tcl|(\psi)$ then $\mathcal{A}(p, s, \varphi) = \mathcal{A}_{in}(p, s, \varphi)$

If $s \notin \mathit{border}(\mathcal{K})$ and $\varphi \in |tcl|(\psi)$ then $\mathcal{A}(p, s, \varphi)$ is defined as follows:

- $\mathcal{A}(p, s, Q) = \begin{cases} \mathtt{tt} & \text{if } s \in L(Q) \\ \mathtt{ff} & \text{otherwise} \end{cases}$
- $\mathcal{A}(p, s, \varphi_1 \wedge \varphi_2) = \begin{cases} \mathtt{tt} & \text{if } \mathcal{A}(p, s, \varphi_1) = \mathtt{tt} \text{ and } \mathcal{A}(p, s, \varphi_2) = \mathtt{tt} \\ \mathtt{ff} & \text{if } \mathcal{A}(p, s, \varphi_1) = \mathtt{ff} \text{ or } \mathcal{A}(p, s, \varphi_2) = \mathtt{ff} \\ \bot & \text{otherwise} \end{cases}$
- $\mathcal{A}(p, s, \neg\varphi_1) = \begin{cases} \mathtt{tt} & \text{if } \mathcal{A}(p, s, \varphi_1) = \mathtt{ff} \\ \mathtt{ff} & \text{if } \mathcal{A}(p, s, \varphi_1) = \mathtt{tt} \\ \bot & \text{otherwise} \end{cases}$
- $\mathcal{A}(p, s, \mathbf{AX}\varphi_1) = \begin{cases} \mathtt{tt} & \text{if } \forall s' \in S : s \xrightarrow{p} s' \Rightarrow \mathcal{A}(p, s', \varphi_1) = \mathtt{tt} \\ \mathtt{ff} & \text{if } \exists s' \in S : s \xrightarrow{p} s' \wedge \mathcal{A}(p, s', \varphi_1) = \mathtt{ff} \\ \bot & \text{otherwise} \end{cases}$
- $\mathcal{A}(p, s, \mathbf{EX}\varphi_1) = \begin{cases} \mathtt{tt} & \text{if } \exists s' \in S : s \xrightarrow{p} s' \wedge \mathcal{A}(p, s', \varphi_1) = \mathtt{tt} \\ \mathtt{ff} & \text{if } \forall s' \in S : s \xrightarrow{p} s' \Rightarrow \mathcal{A}(p, s', \varphi_1) = \mathtt{ff} \\ \bot & \text{otherwise} \end{cases}$
- $\mathcal{A}(p, s, \mathbf{E}(\varphi_1 \, \mathbf{U} \, \varphi_2)) = \begin{cases} \mathtt{tt} & \text{if there exists a p-path } \pi = s_0 s_1 s_2 \ldots \text{ with } s = s_0 \text{ such} \\ & \text{that } \exists x < |\pi| \text{ such that (either } \mathcal{A}(p, s_x, \varphi_2) = \mathtt{tt} \text{ or} \\ & (s_x \in \mathit{border}(\mathcal{K}) \text{ and } \mathcal{A}(p, s_x, \mathbf{E}(\varphi_1 \, \mathbf{U} \, \varphi_2)) = \mathtt{tt})), \\ & \text{and } \forall 0 \leq y < x : \ \mathcal{A}(p, s_y, \varphi_1) = \mathtt{tt} \\ \mathtt{ff} & \text{if for all p-paths } \pi = s_0 s_1 s_2 \ldots \text{ with } s = s_0 \text{ either} \\ & \exists x < |\pi| \text{ such that } (\mathcal{A}(p, s_x, \varphi_1) = \mathtt{ff} \text{ and} \\ & \forall y \leq x : \ \mathcal{A}(p, s_y, \varphi_2) = \mathtt{ff}) \text{or } \forall x < |\pi| : (\mathcal{A}(p, s_x, \varphi_2) = \mathtt{ff} \\ & \text{and } (|\pi| = \infty \text{ or } (s_{|\pi|-1} \in \mathit{border}(\mathcal{K}) \text{ and} \\ & \mathcal{A}(p, s_{|\pi|-1}, \mathbf{E}(\varphi_1 \, \mathbf{U} \, \varphi_2))) = \mathtt{ff})) \\ \bot & \text{otherwise} \end{cases}$

Appendix B: Distributed algorithm

The pseudo-code of the explicit state *node algorithm* is described as Algorithm 3.

Algorithm 3 Basic Node Algorithm

Require: parametrised KS $\mathcal{K}$ and CTL formula Ψ
Let $cl(\psi) = \{\varphi_1, \ldots, \varphi_z\}$ such that $\varphi_i \in cl(\varphi_j) \Rightarrow i \leq j$
Ensure: $\mathcal{A}(\Psi)$
for $i = 1$ to z step 1 **do**
switch Ψ **do**
$true$: **forall** $p \in \mathcal{P}, s \in \mathcal{S}$ **do** $\mathcal{A}(p, s, \Psi) :=$ tt **end for**
a : **forall** $p \in \mathcal{P}, s \in \mathcal{S}$ **do**
if $a \in L(s)$ **then** $\mathcal{A}(p, s, \Psi) :=$ tt **else** $\mathcal{A}(p, s, \Psi) :=$ ff **end for**
$\Phi_1 \wedge \Phi_2$: **forall** $p \in \mathcal{P}, s \in \mathcal{S}$ **do**
if $\mathcal{A}(p, s, \Phi_1) =$ tt and $\mathcal{A}(p, s, \Phi_2) =$ tt **then** $\mathcal{A}(p, s, \Psi) :=$ tt
if $\mathcal{A}(p, s, \Phi_1) =$ ff or $\mathcal{A}(p, s, \Phi_2) =$ ff **then** $\mathcal{A}(p, s, \Psi) :=$ ff **end for**
$\neg\Phi$: **forall** $p \in \mathcal{P}, s \in \mathcal{S}$ **do**
if $\mathcal{A}(p, s, \Phi) \neq \perp$ **then** $\mathcal{A}(p, s, \Psi) := \neg\mathcal{A}(p, s, \Phi)$ **end for**
EXΦ : **forall** $p \in \mathcal{P}, s \in \mathcal{S} \setminus border(\mathcal{K})$ **do**
if $\exists s' \in S : s \xrightarrow{p} s' \wedge \mathcal{A}(p, s', \Phi) =$ tt **then** $\mathcal{A}(p, s, \Psi) :=$ tt
if $\forall s' \in S : s \xrightarrow{p} s' \Rightarrow \mathcal{A}(p, s', \Phi) =$ ff **then** $\mathcal{A}(p, s, \Psi) :=$ ff **end for**
AXΦ : **forall** $p \in \mathcal{P}, s \in \mathcal{S} \setminus border(\mathcal{K})$ **do**
if $\exists s' \in S : s \xrightarrow{p} s' \wedge \mathcal{A}(p, s', \Phi) =$ ff **then** $\mathcal{A}(p, s, \Psi) :=$ ff
if $\forall s' \in S : s \xrightarrow{p} s' \Rightarrow \mathcal{A}(p, s', \Phi) =$ tt **then** $\mathcal{A}(p, s, \Psi) :=$ tt **end for**
E$(\Phi_1 \,\mathbf{U}\, \Phi_2)$: **forall** $p \in \mathcal{P}, s \in \mathcal{S}$ **do if** $\mathcal{A}(p, s, \Phi_2) =$ tt **then** $\mathcal{A}(p, s, \Psi) :=$ tt **end for**
while $\exists p \in \mathcal{P} : (\exists s \in S : \mathcal{A}(p, s, \Psi) \neq$ tt $\wedge\, \mathcal{A}(p, s, \Phi_1) =$ tt$\wedge$
$(\exists s' \in S : s \xrightarrow{p} s' \wedge \mathcal{A}(p, s', \Psi) =$ tt$))$ **do** $\mathcal{A}(p, s, \Psi) :=$ tt **end while**
while $\exists p \in \mathcal{P} : (\exists s \in S : \mathcal{A}(p, s, \Psi) = \perp \wedge (\mathcal{A}(p, s, \Phi_1) =$ ff$\vee$
$(\forall s' \in S : s \xrightarrow{p} s' \wedge \mathcal{A}(p, s', \Psi) =$ ff$)))$ **do** $\mathcal{A}(p, s, \Psi) :=$ ff **end while**
A$(\Phi_1 \,\mathbf{U}\, \Phi_2)$: **while** $\exists p \in \mathcal{P}.(\exists s \in S : \mathcal{A}(p, s, \Psi) \neq$ tt $\wedge\, \mathcal{A}(p, s, \Phi_1) =$ tt$\wedge$
$(\forall s' \in S : s \xrightarrow{p} s' \Rightarrow (\mathcal{A}(p, s', \Psi) =$ tt $\vee\, \mathcal{A}(p, s', \Phi_2) =$ tt$)))$ **do**
$\mathcal{A}(p, s, \Psi) :=$ tt **end while**
while $\exists p \in \mathcal{P} : (\exists s \in S : \mathcal{A}(p, s, \Psi) = \perp \wedge (\mathcal{A}(p, s, \Phi_1) =$ ff$\vee$
$(\exists s' \in S : s \xrightarrow{p} s' \wedge \mathcal{A}(p, s', \Psi) =$ ff$)))$ **do** $\mathcal{A}(p, s, \Psi) :=$ ff **end while**
end switch

Appendix C: Scalability

We evaluate the scalability of our approach on a model of reversible catalytic reaction using various number of intermediate enzyme-substrate complexes. The model is given by the following scheme. The first line is simplified chemical equation of the model. The next lines describes differential equations for every species. The last two lines are parameters we used.

$$
\begin{gathered}
\underline{S + E \rightleftharpoons ES_1 \rightleftharpoons \cdots \rightleftharpoons ES_k \rightleftharpoons P + E} \\
\dot{S} = 0.1 \cdot ES_1 - p_1 \cdot E \cdot S \\
\dot{E} = 0.1 \cdot ES_1 - p_2 \cdot E \cdot S + 0.1 \cdot ES_k - p_2 \cdot E \cdot P \\
\dot{ES_1} = 0.01 \cdot E \cdot S - p_3 \cdot ES_1 + 0.05 \cdot ES_2 \\
\vdots \\
\dot{ES_k} = 0.1 \cdot ES_{k-1} - p_k \cdot ES_k + 0.01 \cdot E \cdot P \\
\underline{\dot{P} = 0.1 \cdot ES_k - p_{k+1} \cdot E \cdot P - 0.1 \cdot P} \\
p_1 = 0.01, p_2 = 0.01, p_3 = 0.2, \\
p_k = 0.15, p_{k+1} = 0.01
\end{gathered}
$$

The following tables provide detailed information about the runtime of the distributed algorithm on various variants of the model and the CTL formula **AG** $P \leq 30$. The value *N/A* denotes the algorithm runs out of memory.

# of params / # of nodes	1	2	3	4	5	6
1	6456	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
2	2610	6179	5089	*N/A*	*N/A*	*N/A*
3	1696	4022	3661	3784	*N/A*	*N/A*
4	854	1759	2285	2454	*N/A*	*N/A*
5	683	1371	1365	1580	1736	*N/A*
6	499	1095	1019	1254	1609	1350
7	435	861	796	1023	1406	1340
8	292	642	650	853	1134	1118
9	258	439	630	752	983	962
10	232	418	557	637	839	822
11	198	347	516	553	784	679
12	177	329	420	562	759	660

Table 1. The runtime in seconds for the model containing 6 variables, 13 thresholds and different number of unknown parameters. The model has almost 3 millions of states (exactly 12^6).

# of variables (# of states) / # of nodes	4 (10^4)	5 (10^5)	6 (10^6)	7 (10^7)
1	3.3	22	794	*N/A*
2	3.3	12	489	*N/A*
3	3.5	9.5	253	*N/A*
4	3.4	8.2	185	8571
5	2.7	8	112	6608
6	2.4	7.2	101	5291
7	2.7	7.3	77	3024
8	2.3	6.6	64	2630
9	2.3	6.3	55	2366
10	2.5	6.8	52	2081
11	2.7	6.2	47	1999
12	2.2	5.6	41	1828

Table 2. The runtime in seconds for the model with 1 unknown parameter, 11 thresholds per the variable and different number of variables.

# of thresholds (# of states) / # of nodes	10 (9^6)	11 (10^6)	12 (11^6)	13 (12^6)
1	274	794	1805	6456
2	196	489	1252	2610
3	84	253	570	1696
4	62	185	408	854
5	51	112	280	683
6	40	101	226	499
7	33	77	192	435
8	29	64	161	292
9	26	55	145	258
10	24	52	108	232
11	23	47	94	198
12	22	41	96	177

Table 3. The runtime in seconds for the model with 1 unknown parameter, 6 variables and different number of thresholds per variable.

Appendix D: Case Study Details

The model of G_1/S transition in the mammalian cell cycle has been displayed in Fig. 4. Before facilitating the finite abstraction and model checking, we first had to approximate the ODE model by means of a piece-wise multi-affine system. The approximation is based on replacing every non-linear (Hill kinetics) function appearing at the right hand side of the equations with a sum of piece-wise linear functions.

In our implementation, we expect the sigmoidal or Hill kinetics to appear in a normal form in order to be properly detectable. In the model, all non-linear kinetic terms (sigmoidal Hill functions) appear in a normal form with the only exception of one. In particular, the numerator $a^2 + [E2F1]^2$ does not fit the normal form for positive Hill kinetics. Therefore, we first rewrite the equation for $E2F1$ in an equivalent way that fits the normal form, while equation for pRB remains the same. The result after rewriting is the following:

$$\begin{aligned}\tfrac{d[pRB]}{dt} &= k_1 \cdot Hill^+(E2F1, K_{m1}, 1) \cdot Hill^-(pRB, J_{11}, 1) - \phi_{pRB}[pRB] \\ \tfrac{d[E2F1]}{dt} &= k_p + \tfrac{k_2 \cdot a^2}{K_{m2}^2} \cdot Hill^+(E2F1, K_{m2}, 2) \cdot Hill^-(pRB, J_{12}, 1) + \\ &\quad + k_2 \cdot Hill^+(E2F1, K_{m2}, 2) \cdot Hill^-(pRB, J_{12}, 1) - \phi_{E2F1}[E2F1]\end{aligned}$$

where

$$Hill^+(x, K_x, n) = \frac{[x]^n}{K_x^n + [x]^n}, \qquad Hill^-(x, K_x, n) = 1 - Hill^+(x, K_x, n).$$

The approximation technique [18] employes a linear programming algorithm that optimally transforms a given sigmoidal continuous function into a series of affine segments. The method optimally approximates all sigmoids depending on a common shared variable together. In our particular case, we requested 70 segments per each variable which resulted into the points on the respective variables as listed in Table 4 and Table 5.

σ_{E2F1} = [0, 0.0233489, 0.0700467, 0.116744, 0.163442, 0.21014, 0.256838, 0.303536, 0.350233, 0.396931, 0.443629, 0.490327, 0.560374, 0.63042, 0.700467, 0.770514, 0.84056, 0.933956, 1.02735, 1.12075, 1.23749, 1.35424, 1.47098, 1.61107, 1.75117, 1.91461, 2.07805, 2.26484, 2.47498, 2.68512, 2.91861, 3.17545, 3.45564, 3.75917, 4.08606, 4.41294, 4.76318, 5.11341, 5.48699, 5.88392, 6.3042, 6.72448, 7.16811, 7.63509, 8.12542, 8.63909, 9.17612, 9.73649, 10.3202, 10.9506, 11.6044, 12.2815, 13.0053, 13.7525, 14.5464, 15.3869, 16.2742, 17.2081, 18.1888, 19.2161, 20.2902, 21.4343, 22.6251, 23.8859, 25.2168, 26.6177, 28.1121, 29.6998, 31.3576, 33.1321, 35]

Table 4. Array of points σ^i_{E2F1} (i is the index of the ith value) computed by optimal partitioning of $E2F1$ variable domain into 70 segments.

Every Hill function from previous equations is replaced by a sum of corresponding affine segments, according to [18]. Resulting equations employ piecewise affine functions called *ramp functions* and have the following form:

σ_{pRB} = [0, 0.072048, 0.108072, 0.144096, 0.18012, 0.216144, 0.264176, 0.312208, 0.36024, 0.408272, 0.456304, 0.516344, 0.576384, 0.636424, 0.708472, 0.78052, 0.852568, 0.936624, 1.02068, 1.11674, 1.21281, 1.32088, 1.44096, 1.56104, 1.69313, 1.83722, 1.99333, 2.16144, 2.34156, 2.53369, 2.74983, 2.97799, 3.21815, 3.4463, 3.67445, 3.89059, 4.10674, 4.32288, 4.53903, 4.76718, 4.99533, 5.22348, 5.46364, 5.7038, 5.95597, 6.20814, 6.47231, 6.7485, 7.03669, 7.33689, 7.6491, 7.97332, 8.30954, 8.65777, 9.03002, 9.41428, 9.82255, 10.2548, 10.6991, 11.1795, 11.6838, 12.2121, 12.7765, 13.3769, 14.0254, 14.7098, 15.4423, 16.2348, 17.0874, 18]

Table 5. Array of points σ^i_{pRB} (i is the index of the ith value) computed by optimal partitioning of pRB variable domain into 70 segments.

$$\begin{aligned}\frac{d[E2F1]}{dt} &= k_p + \frac{k_2 \cdot a^2}{K^2_{m2}} \cdot [\textstyle\sum_{i=2}^{|\sigma_{E2F1}|} R^-(E2F1, \sigma^{i-1}_{E2F1}, \sigma^i_{E2F1}, Hill^-(\sigma^{i-1}_{E2F1}, K_{m2}, 2), Hill^-(\sigma^i_{E2F1}, K_{m2}, 2))]\cdot \\ &\cdot[\textstyle\sum_{j=2}^{|\sigma_{pRB}|} R^-(pRB, \sigma^{j-1}_{pRB}, \sigma^j_{pRB}, Hill^-(\sigma^{j-1}_{pRB}, J_{12}, 1), Hill^-(\sigma^j_{pRB}, J_{12}, 1))]+ \\ &+k_2 \cdot [\textstyle\sum_{i=2}^{|\sigma_{E2F1}|} R^+(E2F1, \sigma^{i-1}_{E2F1}, \sigma^i_{E2F1}, Hill^+(\sigma^{i-1}_{E2F1}, K_{m2}, 2), Hill^+(\sigma^i_{E2F1}, K_{m2}, 2))]\cdot \\ &\cdot[\textstyle\sum_{j=2}^{|\sigma_{pRB}|} R^-(pRB, \sigma^{j-1}_{pRB}, \sigma^j_{pRB}, Hill^-(\sigma^{j-1}_{pRB}, J_{12}, 1), Hill^-(\sigma^j_{pRB}, J_{12}, 1))] - \phi_{E2F1}[E2F1]\end{aligned}$$

$$\begin{aligned}\frac{d[pRB]}{dt} &= k_1 \cdot [\textstyle\sum_{i=2}^{|\sigma_{E2F1}|} R^+(E2F1, \sigma^{i-1}_{E2F1}, \sigma^i_{E2F1}, Hill^+(\sigma^{i-1}_{E2F1}, K_{m1}, 1), Hill^+(\sigma^i_{E2F1}, K_{m1}, 1))]\cdot \\ &\cdot[\textstyle\sum_{j=2}^{|\sigma_{pRB}|} R^-(pRB, \sigma^{j-1}_{pRB}, \sigma^j_{pRB}, Hill^-(\sigma^{j-1}_{pRB}, J_{11}, 1), Hill^-(\sigma^j_{pRB}, J_{11}, 1))] - \phi_{pRB}[pRB]\end{aligned}$$

where $|\sigma_x|$ denotes cardinality of the array σ_x and the positive (R^+) and negative (R^-) ramp functions are defined as follows:

$$R^+(x, \sigma^i, \sigma^j, y_i, y_j) = \begin{cases} y_i + \frac{x-\sigma^i}{\sigma^j-\sigma^i} \cdot (y_j - y_i) & \text{if } \sigma^i \le x \le \sigma^j, \\ 0 & \text{otherwise;} \end{cases}$$

$$R^-(x, \sigma^i, \sigma^j, y_i, y_j) = \begin{cases} y_i + \frac{x-\sigma^i}{\sigma^j-\sigma^i} \cdot (y_i - y_j) & \text{if } \sigma^i \le x \le \sigma^j, \\ 0 & \text{otherwise.} \end{cases}$$

We have bounded the variable domains by assuming $E2F1 < 35$, $pRB < 18$. In the piece-wise multi-affine approximation this ensures that the system dynamics never exits the rectangle defined by the lower left corner $[0, 0]$ and the upper right corner $[35, 18]$. The reason for that is that in each of the boundary vertices the vector of both variable derivatives points towards the interior of the rectangle.

As can be seen in Fig. 5, the system dynamics as displayed by the vector fields of both the original and approximated model agree. The comparison has been realised for the same sampling resolution of the variable derivatives.

The points introduced by the piece-wise multi-affine approximation make a rectangular partition of the system phase space provided that on each single rectangle of the partition the system is multi-affine. This allows us to transform the system into a finite state transition system according to [4]. In our case the model contains $70^2 = 4900$ states. The abstraction is guaranteed to be an over-approximation with respect to the continuous piece-wise multi-affine system (see [10] for overview of the results).

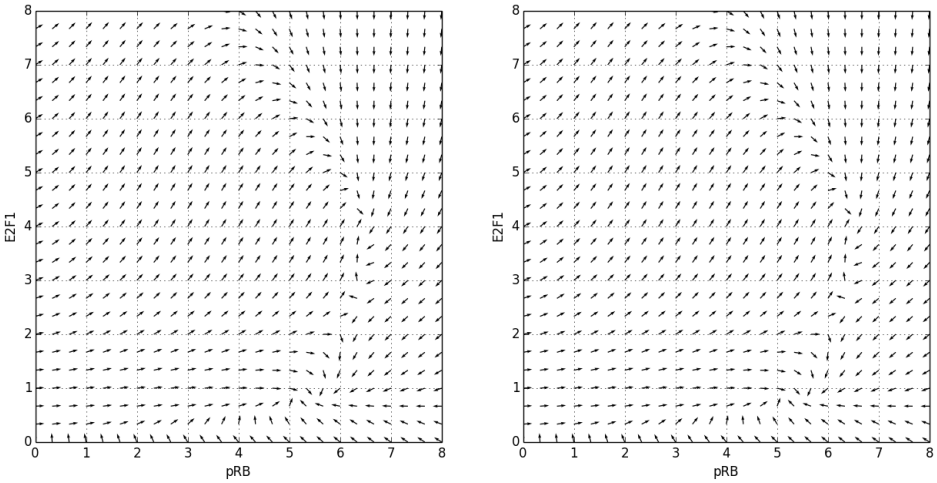


Fig. 5. (left) Vector field sampled on the original non-linear ODE model. (right) Vector field sampled on the piece-wise multi-affine approximation.

Finally, we check the CTL properties defined in Section 3. We have considered the deactivation/degradation parameter ϕ_{pRB} in the range $[0.001, 0.025]$ that is suspected as the hypothesised (slow) time scale of the tumor suppresor protein deactivation processes. In that range we want to identify the property of bistable switch that causes the system to take a possibility to select one of two significantly different asymptotically stable equilibria depending on the initial concentration of both proteins. Fig. 6 gives a 3D visualisation of the states and parameter values satisfying the overall formula $\varphi \equiv EFAG\,\texttt{high} \wedge EFAG\,\texttt{low}$ (yellow) and the particular "stable attractor regions" subformulas $\varphi_1 \equiv AG\texttt{high}$ (red) and $\varphi_2 \equiv AG\,\texttt{low}$ (blue).

As regards the over-approximation, the strongest interpretation goes with ACTL formulae φ_1 and φ_2. In particular, the displayed portion of parameter values (Z-axes) corresponding to the high stable state guarantee the presence of the attractor in that region. The same applies to the low stable state. However, the exact parameter ranges exemplifying stability may be larger since there can exist points out of the identified range where the formula is violated only by a spurious behaviour introduced due to over-approximation.

Interpretation of the results for $EFAG$ formulas is weaker due to presence of spurious reachability. The yellow region can be interpreted as an approximation of the decision points from which both stable states might be reached. However, what is guaranteed is imposibility of reaching both red and blue regions simultaneously from the states that make the complement of the yellow region. This gives us a good estimate for parameter values where the bistable switch is impossible.

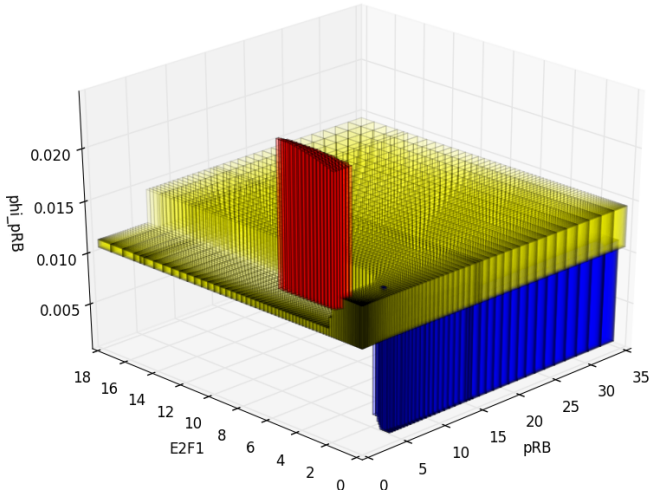


Fig. 6. Coloured model checking results. Red and blue parts correspond to the high and low stable regions, respectively. Yellow part displays the states where the overall *bistable switch* formula φ holds.

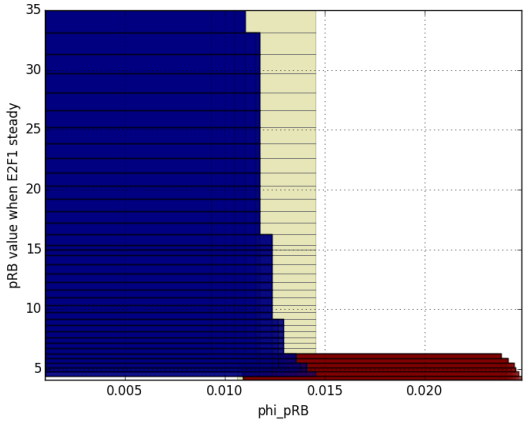


Fig. 7. Coloured model checking results projected onto pRB/ϕ_{pRB} plane. Red and blue parts correspond to the high and low stable regions, respectively. Yellow part displays the states where the overall *bistable switch* formula φ holds.

Fig. 7 shows the projection of the 3D visualisation to the variable pRB and the parameter value. It supplements the $E2F1$ projection shown in Fig. 4(right). In particular, we can see that the low stable state of $E2F1$ traverses almost entire domain of pRB which is not the case for the high stable state. This gives us (guaranteed) information that the low stable state is much more robust with respect to concentration of pRB and the parameter value than the high stable state.