Detecting Attractors in Biological Models with Uncertain Parameters

Jiří Barnat, Nikola Beneš(✉), Luboš Brim, Martin Demko, Matej Hajnal, Samuel Pastva, and David Šafránek

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

Abstract. Complex behaviour arising in biological systems is typically characterised by various kinds of attractors. An important problem in this area is to determine these attractors. Biological systems are usually described by highly parametrised dynamical models that can be represented as parametrised graphs typically constructed as discrete abstractions of continuous-time models. In such models, attractors are observed in the form of terminal strongly connected components (tSCCs). In this paper, we introduce a novel method for detecting tSCCs in parametrised graphs. The method is supplied with a parallel algorithm and evaluated on discrete abstractions of several non-linear biological models.

1 Introduction

Biological systems as understood in systems biology are considered to be complex dynamical systems with a large extent of non-linear interactions. Interactions among systems components have the form of negative or positive feedback, the interplay of which can cause hardly predictable or even chaotic behaviour to emerge. In general, long-term systems behaviour may be significantly affected by the coexistence of dozens of complex and concurrent flows of information. For example, the irreversible decision processes observed in cell-cycle [24] or tissue development [18] arise from feedback loops that allow the cell to stabilise in several significantly different states each implying a unique phenotype.

Some of the problems related to the study of systems dynamics, which initially appear extremely complicated, can be greatly simplified if we concentrate on their *long-term behaviour*, i.e. what happens eventually. This idea finds its mathematical expression in the concept of an *attractor*.

Attractors can be seen as a special type of a portrait in the phase space. Points in a phase space represent the value of each of the system's variables at each moment of time. As the system changes over time, the data points make up a trajectory. Trajectories can be arranged into a phase portrait. Certain phase portraits then display attractor(s) as the long-term stable sets of points of the

This work has been supported by the Czech Science Foundation grant GA15-11089S and by the Czech National Infrastructure grant LM2015055.

DOI: 10.1007/978-3-319-67471-1_3

dynamical system, i.e. the locations in the phase portrait towards which the system's dynamics are attracted after transient phenomena have died down.

Attractors can reveal important information about the causal elements operative in a system, e.g. that the variables are non-linearly related to one another and so forth. That is why a quantitative and qualitative study of the geometrical, topological, and other properties of the attractors can yield deep insights into the system's dynamics.

Complex behaviour arising in biological systems is thus typically characterised by various kinds of attractors. An important problem of systems analysis is to determine the number and position of attractors. Biological systems are usually described by highly parametrised differential equations that can be approximated and abstracted by discrete systems [3,11,16]. In discrete systems, the most typical attractors can be observed in the form of terminal strongly connected components (tSCCs) [23]. We use this fact to provide an *efficient parallel* algorithm for automatised detection of such attractors in discrete models and in discrete abstractions of continuous models of dynamical systems. Alternatively, we could use a *general* method based on model checking for identifying non-trivial phase portraits in systems dynamics [4]. That general method needs to employ a hybrid temporal logic for which the algorithm is significantly more computationally demanding. This is a motivation to focus on a specific method targeting attractors.

Our Contribution. We introduce a novel approach for detection of attractors in parametrised systems in which attractors are understood as tSCCs in graphs representing the systems dynamics. We provide a parallel efficient algorithm for detecting tSCCs in parametrised graphs. We supply the method with a set of heuristics improving expected computation times. We evaluate the algorithm on several non-linear biological models. We additionally provide efficient algorithm variants for the simpler problems of tSCC counting and for deciding the question whether the parametrised graph has at least a given number tSCC.

Related Work. The existing solutions to attractor detection in non-linear continuous models are typically based on numerical methods working in two-dimensional systems while higher-dimensional systems remain a challenge [15]. In the case of discrete models, the problem is directly reduced to identification of SCCs which can be done efficiently for higher-dimensional systems in the non-parametrised case [8,9,17]. Owing to the fact that parameter space of a biological system explodes combinatorially with the arity of component influences, parameter uncertainty results in enormously large sets of parameter values. To that end, attractor detection in parametrised models remains to be a grand challenge in general.

On the technical side, our algorithm adapts the known parallel algorithms [1] to the parametrised setting and adds the possibility to accelerate the computation if only the number of tSCCs is requested without the need to enumerate the attractors. Moreover, this paper shows that exploiting the projection of parameters to the system dynamics gives an advantage of significantly faster computation than that achievable with a naïve execution of Tarjan's algorithm for SCC decomposition [25] scanning all parameter values one-by-one.

2 Methods

In the following, we will consider parametrised graphs which are directed graphs with self-loops allowed and edges labelled by parameters taken from a given parameter set.

Definition 1. *A* parametrised graph *is a triple $G = (V, E, \mathbb{P})$ where V is a finite set of* vertices*, $\mathbb{P}$ is a set of* parameter valuations *and $E \subseteq V \times \mathbb{P} \times V$ is the set of parametric edges. We write $u \xrightarrow{p} v$ instead of $(u, p, v) \in E$ when E is clear from the context. For a set of parameters $P \subseteq \mathbb{P}$, we also write $u \xrightarrow{P} v$ to denote that $P = \{p \in \mathbb{P} \mid u \xrightarrow{p} v\}$. For every $p \in \mathbb{P}$, the* restriction of G on p *is the graph $G_p = (V, E_p)$ where $E_p = \{(u, v) \mid (u, p, v) \in E\}$.*

Note that the $\xrightarrow{P}$ notation allows us to see a parametrised graph as an edge-labelled graph whose edges are labelled by sets of parameter valuations. The sets of parameter valuations can be possibly encoded in a symbolic way (say, using interval representation or formulae of a suitable logic).

In order to define our main problem, we now define the attractors of a graph. In general dynamical systems theory an attractor [20] is the smallest set of states (points in the phase space) invariant under the dynamics. Here we consider a discrete abstraction of a dynamical system in the form of a parametrised graph in which the dynamics is represented using paths. The respective abstraction of the notion of an attractor coincides with the notion of a terminal strongly connected component (tSCC) of a graph. We will thus use the notions of an attractor and a tSCC interchangeably.

Definition 2. *Let $G = (V, E)$ be a directed graph. We say that a vertex $t \in V$* is reachable *from a vertex $s \in V$ if $(s, t) \in E^*$ where E^* denotes the reflexive and transitive closure of E.*

A set of vertices $C \subseteq V$ is strongly connected*, if for any two vertices $u, v \in C$, we have that v is reachable from u. A* strongly connected component *(SCC) is a* maximal *strongly connected set $C \subseteq V$, i.e. such that no C' with $C \subsetneq C' \subseteq V$ is strongly connected. A strongly connected component C is* trivial *if C is made of a single vertex c and $(c, c) \notin E$, and is* non-trivial *otherwise. Furthermore, C is called* terminal *(tSCC) if $(C \times (V \setminus C)) \cap E = \emptyset$.*

In graph theory, tSCCs are also called knots [7,13], with the minor difference that some authors require a knot to have at least two vertices.

We are now ready to state the algorithmic problem we are interested in.

Problem 1 (tSCCs Detecting Problem). Let $G = (V, E, \mathbb{P})$ be a parametrised graph. Our goal is to enumerate, for every parameter valuation $p \in \mathbb{P}$, all tSCCs in the graph G_p, the restriction of G on p.

We may also sometimes be interested in certain simpler versions of the problem. In the *counting* version, we are only interested in the number of tSCCs, i.e. we want to compute the function $c : \mathbb{P} \to \mathbb{N}$ that assigns to each parameter valuation p the number of tSCCs in G_p. In the *threshold* version, we are given a threshold number of tSCCs and want to partition the set of parameter valuations $\mathbb{P}$ into those for which G_p contains at least the given number of tSCCs and those for which it does not. Finally, in the *existential threshold* version, we only aim to decide whether there exists a parameter valuation p for which G_p contains at least the given number of tSCCs.

2.1 Algorithm

Our goal is to develop a parallel (shared-memory or distributed-memory) algorithm for solving the tSCCs Detecting Problem. A simple *sequential* solution to the problem is to use any reasonable SCC decomposition algorithm (e.g. Tarjan's [25]) and enumerate the terminal components in the residual graph. However, all known optimal sequential SCC decomposition algorithms use the depth-first search algorithm, which is suspected to be non-parallelisable [22]. There are known parallel SCC decomposition algorithms; for a survey we refer to [1]. Our approach here, however, is based on the observation that we do not have to compute all the SCCs in order to enumerate the terminal ones. Furthermore, instead of scanning through all parameter valuations and solving the problem for every one of them separately our approach deals with sets of parameter valuations directly. This makes our algorithm suitable for use in connection with various kinds of symbolic set representations.

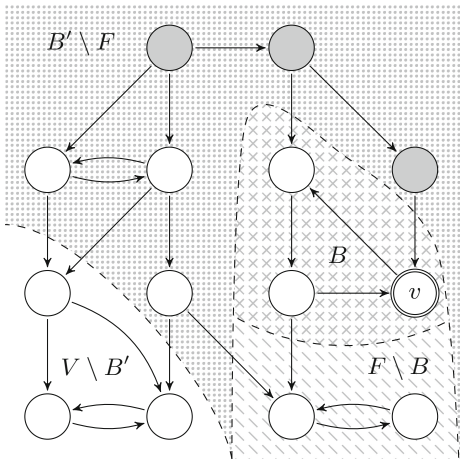


Fig. 1. Illustration of the non-parametrised version of our algorithm.

The main idea of the Terminal Component Detection (TCD) algorithm lies in repeated reachability, which is known to be easily parallelisable. To explain the idea we start with a non-parametrised version of the algorithm first. The following explication is illustrated in Fig. 1. Let us assume a given (non-parametrised) graph $G = (V, E)$. We initialise a tSCC counter to 1: clearly, every graph has at least one tSCC. We choose an arbitrary vertex $v \in V$ (denoted by the double circle in the illustration) and compute all vertices reachable from v; let us call the resulting set of vertices F. We further compute the set of all vertices backwards-reachable from v inside F; we call the resulting set B. Finally, we compute all vertices backwards-reachable from any vertex of F; let us call this set B'.

There are several observations to be made at this point. Clearly, B is an SCC of the graph and moreover, it is a terminal SCC iff $F \setminus B$ is empty. Furthermore, $B' \setminus F$ contains no tSCCs: all vertices in $B' \setminus F$ have a path to a vertex in F. We are thus in one of the following situations:

- $F \setminus B = \emptyset$, $V \setminus B' = \emptyset$: There are no further tSCCs and the algorithm ends.
- $F \setminus B \neq \emptyset$, $V \setminus B' = \emptyset$: We recursively run the algorithm in $F \setminus B$.
- $F \setminus B = \emptyset$, $V \setminus B' \neq \emptyset$: We have found one tSCC (namely, B) and there is at least one tSCC in $V \setminus B'$. We thus increase the counter by one and recursively run the algorithm in $V \setminus B'$.
- $F \setminus B \neq \emptyset$, $V \setminus B' \neq \emptyset$: Observe that no tSCC may intersect both $F \setminus B$ and $V \setminus B'$. We thus increase the counter by one (we know that there is one more tSCC) and recursively run the algorithm twice: in $F \setminus B$ and in $V \setminus B'$.

Note that when running the algorithm recursively twice, the two subgraphs are independent (there is no path from $F \setminus B$ to $V \setminus B'$ or vice versa) and the tasks can thus be run in parallel. The correctness of the algorithm is based on the following invariant. Let t be the number of concurrently running tasks (i.e. invocations of the algorithm), let d be the number of discovered tSCCs (see item 3 above), and let c be the value of the counter. The invariant is $t + d = c \leq$ the number of tSCCs in the graph. Clearly, the algorithm eventually discovers every tSCC: every task is only run on a subgraph known to contain a tSCC and every task ends with a tSCC discovery or a recursive run of another task. Thus, at the end, $d = c =$ the number of tSCCs in the graph.

We also enhance the TCD algorithm with the notion of *trimming* in the manner of [19]. Before every recursive invocation of our algorithm, we iteratively remove all vertices with no incoming edges until a fixed-point is reached. Clearly, such vertices cannot be included in a tSCC (with a minor technical exception, see below) and we thus want to avoid choosing such vertices as starting points. In Fig. 1, the removed vertices are marked in grey; furthermore, the $V \setminus B'$ part of the graph contains one vertex that would be removed in the next recursive run.

Note that trimming may eliminate trivial tSCCs, i.e. tSCCs consisting of a single vertex without any outgoing edges. If it is desirable to include trivial tSCCs in the output, we simply modify the trimming algorithm to check whether the eliminated vertices are tSCCs.

2.2 Parametrised Algorithm

We now extend the basic idea with parameter valuations. We first need to modify the reachability procedure to take parameter valuations into account. To be able to formulate the algorithm, we need a notion of parametrised sets of vertices.

Formally, a parametrised set of vertices $\widehat{A}$ is a function $\widehat{A} : V \to 2^{\mathbb{P}}$. We say that v is present in $\widehat{A}$ for $p \in \mathbb{P}$ if $p \in \widehat{A}(v)$. Clearly, whenever $\widehat{A}(v) = \emptyset$, this means that the vertex v is not present in the parametrised set for any parameter valuation. We call a parametrised set $\widehat{A}$ *empty* if $\widehat{A}(v) = \emptyset$ for all vertices v.

To deal with parametrised sets, we use a generalisation of the standard set operations. All the operations are performed element-wise, i.e. the union of parametrised sets $\widehat{A} \cup \widehat{B}$ is defined as the parametrised set $\widehat{C}$ such that $\widehat{C}(v) = \widehat{A}(v) \cup \widehat{B}(v)$ for all v; similarly for intersection and set difference.

To define the forward and backward reachable sets, we first need a notion of p-reachability. We say that v is p-reachable from s in $G|_{\widehat{V}}$ if the restricted graph G_p contains a path from s to v that only includes vertices that are present in $\widehat{V}$ for p. We then define $\mathcal{R}_{\widehat{V}}(s, v) = \{p \mid v \text{ is } p\text{-reachable from } s \text{ in } G|_{\widehat{V}}\}$. The reachability sets are then defined as follows: $\texttt{cfwd}(\widehat{V}, \widehat{X})$ denotes the parametrised set of vertices forward reachable from $\widehat{X}$ inside $\widehat{V}$ and similarly for `cbwd`.

$$\texttt{cfwd}(\widehat{V}, \widehat{X}) = \widehat{A}, \text{ where } \widehat{A}(v) = \widehat{V}(v) \cap \bigcup_{s \in V} \left(\widehat{X}(s) \cap \mathcal{R}_{\widehat{V}}(s, v)\right)$$

$$\texttt{cbwd}(\widehat{V}, \widehat{X}) = \widehat{A}, \text{ where } \widehat{A}(v) = \widehat{V}(v) \cap \bigcup_{s \in V} \left(\widehat{X}(s) \cap \mathcal{R}_{\widehat{V}}(v, s)\right)$$

Both functions can be effectively computed using a fixed-point algorithm.

Algorithm 1 shows the resulting parametrised algorithm. The basic idea is the one described above, extended with parametrised sets. There is, however, one key difference. It is not sufficient to select just one vertex as the basis for the first (forward) reachability. The reason is that as the algorithm proceeds, the investigated parametrised set of vertices may contain vertices with various incomparable sets of associated parameter valuations. Therefore, in the main part of the algorithm we collect several starting vertices with disjoint parameter valuation sets that together cover all parameter valuations that are present in the currently explored parametrised set of vertices.

The problem is illustrated in Fig. 2. We start with a parametrised graph with four vertices and two parameter valuations, depicted by the red (empty) and blue (filled) coloured dots. In the first iteration of the algorithm, $\widehat{V}$ consists of all the vertices with both parameter valuations. We select a starting vertex (depicted by a double circle) and compute $\widehat{F}$, which is in this case equal to $\widehat{B}$ as well as $\widehat{B'}$. The parametrised set $\widehat{V} \setminus \widehat{B'}$ is non-empty, we thus increase the counter for both parameter valuations. Note that the counter also has to be parametrised.

In the next iteration of the algorithm, let us first assume that we would only select a single vertex (see the second row in Fig. 2). Let us thus select t and compute $\widehat{F}$ again. It is again equal to $\widehat{B}$ and $\widehat{B}'$; this means that $\widehat{V} \setminus \widehat{B'}$ is again non-empty. In this case, however, it would be an error to increase the counter for

```
1  procedure init($G = (V, E, \mathbb{P})$)
2      $\mathsf{count}_p \leftarrow 1$ for all $p \in \mathbb{P}$
3      $\widehat{V} \leftarrow [\forall v \in V : v \mapsto \mathbb{P}]$
4      main($\widehat{V}$)
5  procedure main($\widehat{V}$)
6      trim $\widehat{V}$
7      $P \leftarrow \{p \in \mathbb{P} \mid \exists u : p \in \widehat{V}(u)\}$
8      $\widehat{S} \leftarrow [\forall v \in V : v \mapsto \emptyset]$
9      while $P$ is not empty do
10         choose $v$ such that $\widehat{V}(v) \cap P \neq \emptyset$
11         add $v \mapsto \widehat{V}(v) \cap P$ to $\widehat{S}$
12         $P \leftarrow P \setminus \widehat{V}(v)$
13     $\widehat{F} \leftarrow$ cfwd($\widehat{V}, \widehat{S}$)
14     $\widehat{B} \leftarrow$ cbwd($\widehat{F}, \widehat{S}$)
15     run in parallel
16         worker 1
17             $P \leftarrow$ {parameters appearing in $\widehat{B}$ and not appearing in $\widehat{F} \setminus \widehat{B}$}
18             $\widehat{B}$ restricted to $p$ is a tSCC for all $p \in P$
19             main($\widehat{F} \setminus \widehat{B}$) if $\widehat{F} \setminus \widehat{B}$ is not empty
20         worker 2
21             $\widehat{B'} \leftarrow$ cbwd($\widehat{V}, \widehat{F}$)
22             $\mathsf{count}_p \leftarrow \mathsf{count}_p + 1$ for all $p$ occurring in $\widehat{V} \setminus \widehat{B'}$
23             main($\widehat{V} \setminus \widehat{B'}$) if $\widehat{V} \setminus \widehat{B'}$ is not empty
```

Algorithm 1. Parallel algorithm for tSCC counting in parametrised graphs.

the red parameter valuation as there are, in fact, only two tSCCs for each of the parameter valuations. We thus need to keep track of the parameter valuations of the selected vertex and if they do not cover all parameter valuations, we select another one. In the case of the example, it is correct to choose both t and u as starting vertices (see the third row).

The example also illustrates that the choice of the vertex on line 10 may influence the performance of the algorithm. Had we chosen v in the second iteration of the algorithm, no other vertices would be necessary. It might, however, be not always possible to find one vertex that covers all parameter valuations in $\widehat{V}$. Another issue is that a wrong choice of starting vertices may slice the set of parameter valuations into too many small subsets. In Sect. 4.1 we discuss and evaluate two vertex selection heuristics, one based on the cardinality of the parameter valuation set and another that aims to choose vertices close to tSCCs.

It remains to describe how the parametrised TCD algorithm proceeds with trimming and keeping the counter. The parametrised trimming works as follows: For every vertex v in $\widehat{V}$, we compute the set of parameter valuations under which v has no incoming edges. If all the sets are empty, the trimming is done.

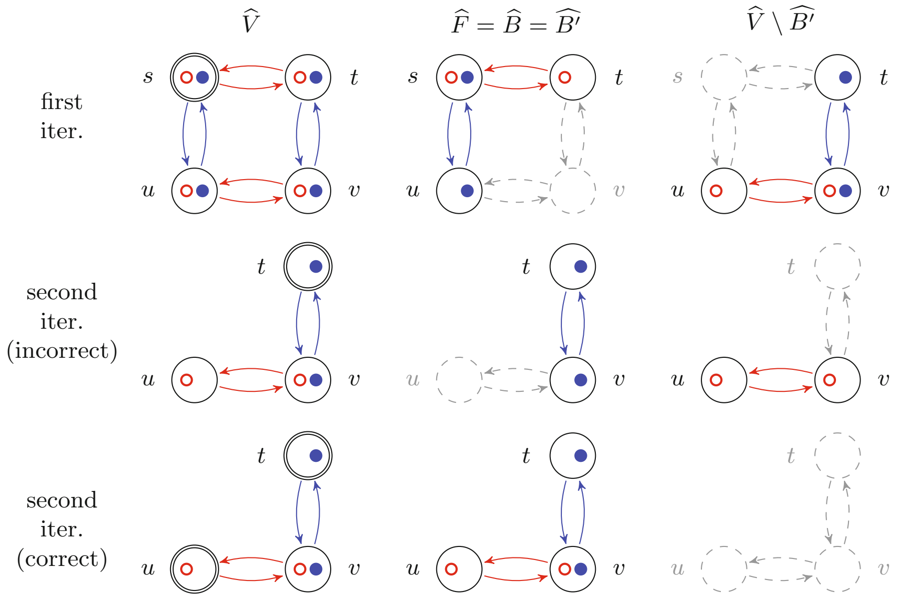


Fig. 2. Illustration of Algorithm 1. (Color Figure Online)

Otherwise, we remove the parameter valuations from $\widehat{V}$ and repeat the process. As for the counter, instead of a single number, we use a mapping $\mathbb{P} \to \mathbb{N}$ that assigns to each parameter valuation the number of tSCCs in its induced graph. The actual implementation of the counter depends on the (symbolic) representation of the parameter valuations and is discussed in Sect. 4.1.

Note that the algorithm as presented in Algorithm 1 solves both the tSCCs Detecting Problem and its counting version. If only the *counting* version is considered, we simply remove lines 17 and 18. Furthermore, if we are only interested in the *threshold* version of the problem, we may stop considering all parameter valuations p for which $\texttt{count}_p$ has already reached the threshold. Moreover, in the *existential threshold* version, we stop the whole algorithm once any parameter valuation has reached the threshold.

3 Applications

We apply the method to several models used in systems biology. Since most of the existing and widely used models are represented by means of ordinary differential equations (ODEs), we employ the piece-wise multi-affine approximation [16] and rectangular abstraction procedures [2,11] to obtain a discrete representation of the systems dynamics in the form of a finite parametrised graph.

3.1 Discretisation of ODE Models

In this section, we briefly describe the format of the ODE models used and the subsequent procedures of approximation and abstraction that allow us to apply the method defined in Sect. 2.

Model. We consider $\mathbb{P} \subseteq \mathbb{R}^m_{\geq 0}$ as the *continuous parameter valuation space* of dimension m. A *biological model* $\mathcal{M}$ is given as 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, $\mu = (\mu_1, \ldots, \mu_m)$ is a vector of parameters such that μ is evaluated in $\mathbb{P}$, and $f = (f_1, \ldots, f_n)$ is a vector where each component is a function constructed as a sum of reaction rates where every sum member is an affine or bi-linear function of x, or a sigmoidal function of x. An important requirement is that each f_i must be affine in μ and there exist no k, l such that $k \neq l$ and μ_k, μ_l both occur in some f_i. Moreover, we assume that every variable x_i has a bound denoted by x_{max_i}. In consequence, we require for all $p \in \mathbb{P}$ that no trajectory can exit the bounds. Formally, $\forall p \in \mathbb{P}, \forall i \in \{1, ..., n\} : (x_i = 0 \Rightarrow f_i(x,p) > 0) \wedge (x_i = max_i \Rightarrow f_i(x,p) < 0)$. Similarly to [2], we assume $\mathbb{P}$ includes *almost all* parameter valuations excluding singular cases for which some trajectory can slide along a threshold plane. In particular, any parameter valuation p for which some component of $f(x,p)$ can be zero on a boundary of some rectangle (as defined below) is not allowed. In consequence, a fixed point can appear only in a rectangle interior.

The restriction imposed on f covers mass action kinetics with stoichiometric coefficients not greater than one and any sigmoidal kinetics such as all significant variants of enzyme or Hill kinetics. Parameters must be independent and cannot appear in an exponent or a denominator of the kinetic function employed.

Approximation. To proceed with discretisation, the model $\dot{x} = f(x, \mu)$ has to satisfy the criterion that every f_i is piecewise multi-affine (PMA) in x. To transform the model into this form, we employ the approach defined in [16]. In particular, each sigmoidal function member in f_i is approximated with an optimal sequence of piecewise affine ramp functions. In this procedure, a finite number of thresholds is introduced for every component of x. The crucial factor of the approximation error is the number of piecewise affine segments. Though there is not yet a method that would somehow propagate the information on approximation error into the trajectories of the resulting PMA model, it has been shown on several case studies that the approximation does affect the system's vector field only negligibly [12,16].

Abstraction. We employ the rectangular abstraction [3,16]. We assume that we are 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 numbers: $R(j_1, \ldots, j_n) = [\theta^1_{j_1}, \theta^1_{j_1+1}] \times \cdots \times [\theta^n_{j_n}, \theta^n_{j_n+1}]$, 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 $R(j_1, \ldots, j_n)$.

The abstraction results in a symbolic description of a parametrised graph, $G = (V, E, \mathbb{P})$ where $V = \{(j_1, \ldots, j_n) \mid \forall i : 1 \le j_i < n_i\}$ such that each $v \in V$ represents the rectangle $R(v)$. The relation $u \xrightarrow{P} v$ is defined for a parameter valuations set $P \subseteq \mathbb{P}$ between any two nodes $u, v \in V$, $u \neq v$, for which $R(u) \cap R(v)$ forms an $(n-1)$-dimensional (hyper)rectangle ($R(u), R(v)$ are neighbouring in one dimension) and for which one of the following conditions holds:

- $\exists! j. v_j = u_j + 1$, $\forall i, i \neq j : v_i = u_i$ and for each $p \in P$ there exists $\hat{x} \in VR(u) \cap VR(v)$ satisfying $f_j(\hat{x}, p) > 0$;
- $\exists! j. v_j = u_j - 1$, $\forall i, i \neq j : v_i = u_i$ and for each $p \in P$ there exists $\hat{x} \in VR(u) \cap VR(v)$ satisfying $f_j(\hat{x}, p) < 0$.

Additionally, there is a self-loop defined for any $u \in V$ and a parameter valuations set $P \subseteq \mathbb{P}$ such that $\forall p \in P : \mathbf{0} \in hull\{f(\hat{x}, p) | \hat{x} \in VR(u)\}$.

Every edge is associated with a subset $P \subseteq \mathbb{P}$ of parameter values under which it is enabled. Finite number of thresholds implies finite number of distinct parameter sets that can appear on transitions in the model. Total number of parameter sets for an abstraction of model $\mathcal{M}$, denoted $|\mathbb{P}_{|\mathcal{M}}|$, is thus finite.

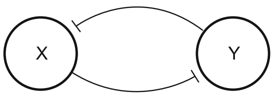
The rectangular abstraction approximates the existence of a fixed point in a rectangle. This is achieved conservatively by introducing reflexivity for every rectangle such that there is a zero vector included in the convex hull of all vertices of the rectangle. In other words, this is a necessary condition for the existence of a point where the derivatives in all coordinates are zero. In this setting, it has been shown that rectangular abstraction is conservative (overapproximation) with respect to almost all trajectories of the approximated (PMA) model [2].

The conservativeness of the abstraction and the consideration of only those parameter values for which the dynamics is bounded (cannot exit the interval $[\theta^i_{j_1}, \theta^i_{j_{n_i}}]$ for any $i \le n$) together imply that *every tSCCs in the abstraction covers an attractor in the PMA system.* This implies that the number of discovered tSCCs in the abstraction is a lower bound for the number of attractors in the corresponding PMA system. To interpret the results for the original system, local linearisation of non-linear vector field preserves topological equivalence implying preservation of hyperbolic attractors [10]. For complex attractors, we are not aware of any relevant mathematical results leaving it open for future research.

3.2 Case Studies

To demonstrate the applicability and benefits of our approach, it is applied to three biological models. Two of them are motifs in genetic regulatory networks and the third is the main part of the cell cycle control in mammalian cells. Note that all the models in this section are PMA approximated models of the original ODEs. Parameter sets for which the method is able to run, *allowed parameters*, consist of independent parameters and parameters not nested in PMA system.

Bi-stable repressilator. The first model to be presented is the smallest repressilator motif, studied in [6,14]. It includes two nodes which inhibit each other



$$\frac{d[X]}{dt} = k_1 \frac{K_1^{n1}}{K_1^{n1} + [Y]^{n1}} - \phi_X [X]$$

$$\frac{d[Y]}{dt} = k_2 \frac{K_2^{n2}}{K_2^{n2} + [X]^{n2}} - \phi_Y [Y]$$

$$k_1 = k_2 = 1,\ K_1 = K_2 = 5,$$
$$n_1 = n_2 = 5,\ \phi_X = \phi_Y = 0.1$$

k_i	ϕ_I
$(0.1, 10)$	$(0, 1)$

Fig. 3. The bi-stable repressilator regulatory network (left) and its ODE model taken from [6] (middle). The parameters and their corresponding value intervals we have considered for all $i \in \{1, 2\}, I \in \{X, Y\}$ (right).

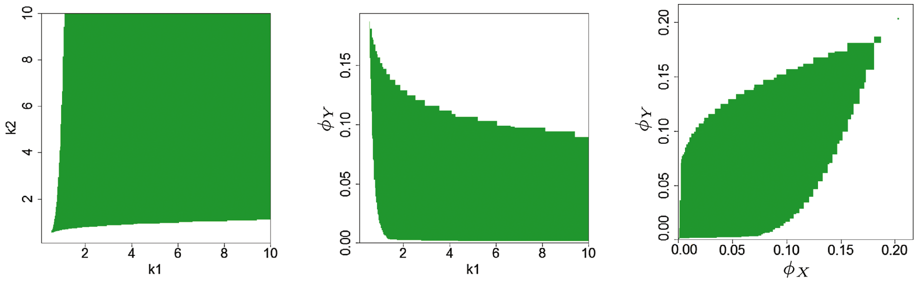


Fig. 4. The parameter space and the corresponding number of terminal components (one in white, two in green). The remaining parameter interval, which is not shown, exhibits one terminal component. Thanks to the symmetry of the model, there are only 3 pairs of allowed parameters (Color figure online).

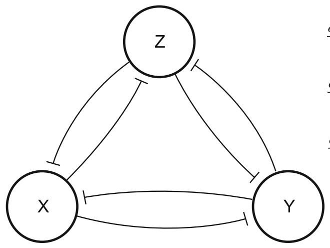
(Fig. 3 left). In biology, this motif is very often present in gene regulatory networks, where X represents the product of $geneX$ which inhibits the production of $geneY$ and vice versa.

According to [21], there is a bistability in the model with parametrised ϕ_X. A bistability region has been discovered for $\phi_X \in (0.022, 0.119) \cup (0.120, 0.138)$ in [6]. Our algorithm has found a bistability region in $(0.014, 0.156)$ for parametrised ϕ_X. This extension of the parameter interval is caused by the presence of a non-trivial terminal component, instead of a *sink* [5].

Additionally, we have managed to analyse this model for all pairs of parameters allowed for the prototype implementation of the method (Fig. 4).

Tri-stable toggle switch. The tri-stable toggle switch is a model of a 3-variable repressilator in which each node inhibits not only one but both of its neighbours (Fig. 5 left). Just one of the two ingoing inhibitions is enough to repress any entity. Therefore the ODE model contains a multiplication of negative Hill functions in the entity regulation (Fig. 5 right).

We have analysed this model for all pairs of parameters allowed for the implementation. As predicted, the model shows tri-stability for specific parameter values (Fig. 6). Additionally, we have managed to analyse this model for a triple of parameters (ϕ_X, ϕ_Y, ϕ_Z) using a reduced state space.



$$\frac{d[X]}{dt} = k_1 \frac{K_{y1}^{n1}}{K_{y1}^{n1}+[Y]^{n1}} \cdot \frac{K_{z1}^{n2}}{K_{z1}^{n2}+[Z]^{n2}} - \phi_X[X]$$

$$\frac{d[Y]}{dt} = k_2 \frac{K_{x2}^{n3}}{K_{x2}^{n3}+[X]^{n3}} \cdot \frac{K_{z2}^{n4}}{K_{z2}^{n4}+[Z]^{n4}} - \phi_Y[Y]$$

$$\frac{d[Z]}{dt} = k_3 \frac{K_{x3}^{n5}}{K_{x3}^{n5}+[X]^{n5}} \cdot \frac{K_{y3}^{n6}}{K_{y3}^{n6}+[Y]^{n6}} - \phi_Z[Z]$$

$$\forall i,j; k_i = 1,\\ K_{x_i} = K_{y_i} = K_{z_i} = 5,\ n_j = 5,\\ \phi_X = \phi_Y = \phi_Z = 0.1$$

k_i	ϕ_I
$(0.1, 10)$	$(0, 1)$

Fig. 5. The tri-stable toggle switch regulatory network (left) and its ODE model (middle). The parameter value intervals we have considered for all $i \in \{1, 2, 3\}, I \in \{X, Y, Z\}$ (right).

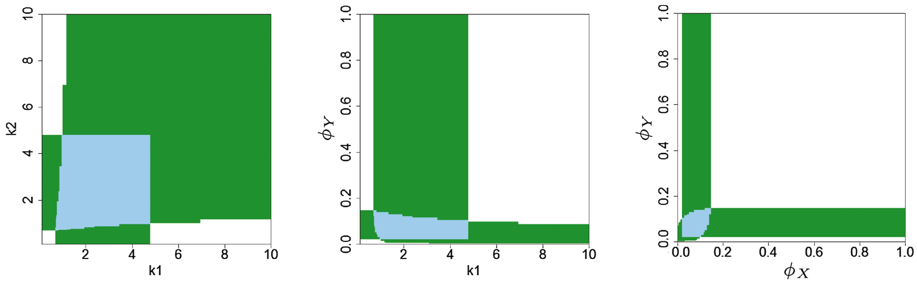
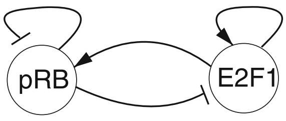


Fig. 6. The parameter space and the corresponding number of terminal components (one in white, two in green, three in blue). Thanks to the symmetry of the model, there are only 3 pairs of allowed parameters (Color figure online).

Regulation of the G_1/S Cell Cycle Transition. As the last case, we have investigated a well-known model representing the central module of the genetic regulatory network governing the G_1/S cell cycle transition in mammalian cells [24]. In particular, the model explains the mechanism behind the irreversible decision for cell division described by a two-gene regulatory network of interactions between the tumour suppressor protein *pRB* and the central transcription factor $E2F1$ (Fig. 7 left). 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. For suitable parameter values, two distinct stable attractors exist. A numerical bifurcation analysis of $E2F1$ stable concentration depending on the degradation parameter of *pRB* (ϕ_{pRB}) has been provided in [24].

A bistability region has been discovered for $\phi_{pRB} \in [0.012, 0.0145]$ in [5]. Our algorithm has found a bistability region in $(0.010, 0.0146)$ for parametrised ϕ_{pRB}. This extension of parameter interval has the same reason as in the first case study—the presence of a non-trivial terminal component, instead of a *sink* [5].

Additionally, we have managed to analyse this model for all pairs of parameters allowed for the implementation (Fig. 8).



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

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

Fig. 7. The G_1/S transition regulatory network (left) and its ODE model taken from [24] (right). The parameter value intervals we have considered are as follows: k_1,(0.1, 10); ϕ_{pRB},(0, 1); k_2,(0.16, 16); k_p,(0.005, 0.5); ϕ_{E2F1},(0, 1).

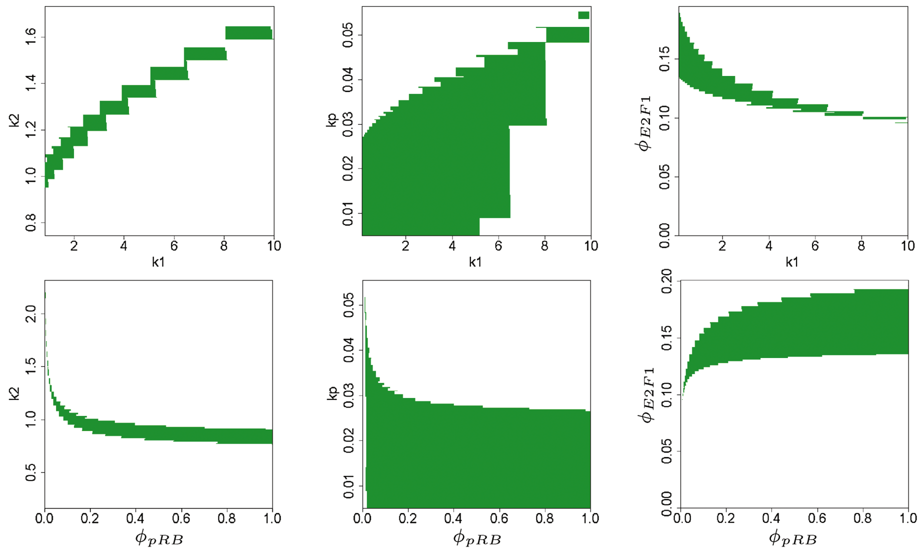


Fig. 8. The parameter space and the corresponding number of terminal components (one in white, two in green). The remaining parameter interval, which is not shown, exhibits one terminal component (Color figure online).

4 Evaluation

We evaluate a prototype implementing the method from Sect. 2 in several aspects such as comparison with the naïve approach, scalability in model size, scalability in $|\mathbb{P}_{|\mathcal{M}}|$, different algorithm types and the heuristics for the initial node selection. In this section, we use all biological models from Sect. 3 which are subject to approximation and abstraction described in that section. In addition, we employ a model of a four-stable switch—an extension of the tri-stable toggle switch with four genes where each of the four genes represses the others. It exhibits four different stable states. Moreover, there are the implementation details demanding deeper understanding used in this section which are described next. Note that all the time results in this section are in seconds and represent the average of four runs on a server with two eight-core processors (Intel Xeon X7560 2.26 GHz) and 448 GiB RAM.

Table 1. The second and third row represent the number of states and the number of parameter valuations for that particular model, respectively; *B*/*R* stands for the *Branch*/*Reach* algorithm.

Cores	*Model1*				*Model2*				*Model3*				*Model4*			
	720e3		320e3		1.5e6		750e3		125e3		75e3		390e3		280e3	
	7.1e6		2.8e6		5.5e6		1.8e6		160e6		57e6		255e6		124e6	
	B	*R*	*B*	*R*	*B*	*R*	*B*	*R*	*B*	*R*	*B*	*R*	*B*	*R*	*B*	*R*
2	842	377	651	176	798	886	326	340	1083	871	416	367	1516	1203	799	705
4	735	261	603	122	511	562	256	247	1000	683	391	301	1319	837	722	519
8	690	208	567	107	392	436	192	203	987	544	377	237	1271	723	680	473
12	706	237	595	120	471	477	219	222	1016	497	387	232	1299	718	702	464
16	719	237	590	119	459	467	216	216	1020	476	389	219	1281	704	706	456

Table 2. The second row represents $|\mathbb{P}_{|\mathcal{M}}|$, the number of parameter valuations for that particular model variant; *B*/*R* stands for the *Branch*/*Reach* algorithm.

Cores	*Model1* (1250 states)				*Model2* (1600 states)				*Model3* (8e3 states)				*Model4* (10e3 states)			
	1058e3		159e3		2.7e6		1.1e6		21e6		3.2e6		9.6e6		668e3	
	B	*R*	*B*	*R*	*B*	*R*	*B*	*R*	*B*	*R*	*B*	*R*	*B*	*R*	*B*	*R*
2	483	440	81	83	642	479	114	127	399	318	144	120	625	655	254	248
4	481	424	77	79	626	329	101	90	369	238	132	87	585	383	241	152
8	495	413	79	78	607	293	97	81	358	197	124	74	585	314	235	127
12	481	427	79	79	608	266	96	68	357	184	128	70	583	253	242	107
16	481	416	78	78	623	259	97	73	365	179	127	71	588	249	230	103

4.1 Implementation Details

In this section, we describe two algorithm variants: *Branch* and *Reach*. They differ in the form of the parallelism employed. The *Branch* algorithm runs two parallel workers each time the computation branches, as described in the pseudo-code. The *Reach* algorithm uses a parallel reachability procedure. Here, we describe some important implementation details of both algorithms:

Parameter representation. Due to the restrictions imposed on the model parameters in Sect. 3, we can represent each parameter set as a grid of disjoint hyper-rectangles. Each parameter set maintains its own grid which is refined or simplified as needed to maintain optimal resource usage.

Component counter. The mapping described in Sect. 2 is implemented as a list of disjoint parameter valuation sets. Intuitively, the set on position i contains all the parameter valuations for which i terminal components have been discovered so far.

Partition function. The *Reach* algorithm performs a partitioning of the state space in order to parallelise the reachability computation. To that end, we exploit the regular (rectangular) structure of our models and define a partitioning which splits the model into almost equally sized rectangular blocks. The number of blocks depends on the number of used cores.

Selection heuristics. We also compare three state selection heuristics. *None* is the naïve heuristics which selects the first available state in the set. The *CARD* heuristics tries to ensure that the symbolic parameter representation is well utilised during the computation. To that end, it selects the state with the highest parameter set cardinality as the initial state. Finally, the *CSTR* heuristics is designed to reduce the number of performed reachability computations by selecting states which are part of (or close to) the terminal components. Our observation is that a state is more likely to be a part of a terminal component if there are more transitions entering the state than leaving it. The *CSTR* heuristics therefore pre-computes this parametrised in/out ratio for all states and then uses it to select the best state. If two states agree on the in/out ratio, the parameter set cardinality is used to decide the winner, just as in the *CARD* heuristics.

4.2 Performance Evaluation

Comparison with the naïve approach. As the naïve approach we use Tarjan's SCC decomposition algorithm followed by the counting of terminal components; run once on G_p for each parameter valuation $p \in \mathbb{P}$. This approach was compared with the best results of our prototype for the same models. For the Bi-stable repressilator with 900 states and 866761 parameter valuations the naïve approach took 2520.46 s while our approach took 153.54 s. For the Tri-stable toggle switch with 1000 states and 436921 parameter valuations the naïve approach took 3815.66 s while our approach took 59.71 s.

Problem of the initial state. We analysed all heuristics on several models and for all cases using either *CSTR* or *CARD* was always a better option; sometimes even ten times faster. In some cases *CARD* was more efficient than *CSTR*.

Scalability in model size. These statistics were performed by both algorithm variants on 4 models each for 2 different sizes. Here, by size we mean the size of the state space together with $|\mathbb{P}_{|\mathcal{M}}|$. These cannot be separated because the size of $\mathbb{P}_{|\mathcal{M}}$ in this kind of models depends on the number of states due to the rectangular abstraction. In Table 1 you may observe that for the majority of models the *Reach* algorithm is the better option. For the models used we define abbreviations: *Model1* for the G_1/S switch, *Model2* for the bi-stable repressilator, *Model3* for the tri-stable switch and *Model4* for the four-stable switch.

Scalability in parameter space. These statistics were performed by both algorithm variants on the four previously mentioned models each for two differently sized parameter spaces with constant size of the state space. In Table 2 you may observe that for the majority of models the *Reach* algorithm is the better option.

5 Conclusion

The novel result of this paper is a parallel algorithm for the detection of terminal SCCs in parametrised graphs. The scalability of the algorithm has been analysed showing a significant speed-up w.r.t. the naïve approach using standard algorithms. We have shown that the algorithm can be sufficiently applied to detect attractors in dynamical systems. The case studies have shown the method can deal with two parameters in a reasonable time and even with three parameters in case of a smaller state space (for the tri-stable toggle switch model). The method provides a fully automated and parallel efficient alternative to traditional bifurcation analysis focused on multi-stability as in [24]. Note that the precision of the results is affected by settings of the approximation and abstraction procedures. Possible imprecisions can be observed as discontinuities in plotted results, see Fig. 4. This can be eliminated by manual fine-tuning of the approximation and abstraction. However, we have been primarily interested in the functionality of the algorithm here. Detailed study of the application aspects is left for future work.

References

1. Barnat, J., Chaloupka, J., Van De Pol, J.: Distributed algorithms for SCC decomposition. J. Logic Comput. **21**(1), 23–44 (2011)
2. Batt, G., Belta, C., Weiss, R.: Model checking genetic regulatory networks with parameter uncertainty. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 61–75. Springer, Heidelberg (2007). doi:10.1007/978-3-540-71493-4_8
3. Batt, G., Yordanov, B., Weiss, R., Belta, C.: Robustness analysis and tuning of synthetic gene networks. Bioinformatics **23**(18), 2415–2422 (2007)
4. Beneš, N., Brim, L., Demko, M., Pastva, S., Šafránek, D.: A model checking approach to discrete bifurcation analysis. In: Fitzgerald, J., Heitmeyer, C., Gnesi, S., Philippou, A. (eds.) FM 2016. LNCS, vol. 9995, pp. 85–101. Springer, Cham (2016). doi:10.1007/978-3-319-48989-6_6
5. 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, Cham (2015). doi:10.1007/978-3-319-23401-4_21
6. Brim, L., Demko, M., Pastva, S., Šafránek, D.: High-performance discrete bifurcation analysis for piecewise-affine dynamical systems. In: Abate, A., Šafránek, D. (eds.) HSB 2015. LNCS, vol. 9271, pp. 58–74. Springer, Cham (2015). doi:10.1007/978-3-319-26916-0_4
7. Chandy, K.M., Misra, J.: Distributed computation on graphs: shortest path algorithms. Commun. ACM **25**(11), 833–837 (1982)
8. Chatain, T., Haar, S., Jezequel, L., Paulevé, L., Schwoon, S.: Characterization of reachable attractors using petri net unfoldings. In: Mendes, P., Dada, J.O., Smallbone, K. (eds.) CMSB 2014. LNCS, vol. 8859, pp. 129–142. Springer, Cham (2014). doi:10.1007/978-3-319-12982-2_10
9. Choo, S.M., Cho, K.H.: An efficient algorithm for identifying primary phenotype attractors of a large-scale boolean network. BMC Syst. Biol. **10**(1), 95 (2016)
10. Coayla-Teran, E.A., Mohammed, S.E.A., Ruffino, P.R.C.: Hartman-grobman theorems along hyperbolic stationary trajectories. Discret. Contin. Dyn. Syst. **17**(2), 281–292 (2007)

11. Collins, P., Habets, L., van Schuppen, J., Černá, I., Fabriková, J., Šafránek, D.: Abstraction of biochemical reaction systems on polytopes. In: IFAC World Congress, pp. 14869–14875. IFAC (2011)
12. Demko, M., Beneš, N., Brim, L., Pastva, S., Šafránek, D.: High-performance symbolic parameter synthesis of biological models: a case study. In: Bartocci, E., Lio, P., Paoletti, N. (eds.) CMSB 2016. LNCS, vol. 9859, pp. 82–97. Springer, Cham (2016). doi:10.1007/978-3-319-45177-0_6
13. Dijkstra, E.W.: In reaction to Ernest Chang's "Deadlock Detection" (1979). http://www.cs.utexas.edu/users/EWD/ewd07xx/EWD702.PDF
14. Dilão, R.: The regulation of gene expression in eukaryotes: bistability and oscillations in repressilator models. J. Theor. Biol. **340**, 199–208 (2014)
15. Dudkowski, D., Jafari, S., Kapitaniak, T., Kuznetsov, N.V., Leonov, G.A., Prasad, A.: Hidden attractors in dynamical systems. Phys. Rep. **637**, 1–50 (2016)
16. 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). doi:10.1007/978-3-642-22110-1_31
17. Guo, W., Yang, G., Wu, W., He, L., Sun, M.: A parallel attractor finding algorithm based on boolean satisfiability for genetic regulatory networks. PLOS ONE **9**(4), 1–10 (2014)
18. MacArthur, B.D., Ma'ayan, A., Lemischka, I.R.: Systems biology of stem cell fate and cellular reprogramming. Nat. Rev. Mol. Cell Biol. **10**(10), 672–681 (2009)
19. McLendon III, W., Hendrickson, B., Plimpton, S.J., Rauchwerger, L.: Finding strongly connected components in distributed graphs. J. Parallel Distrib. Comput. **65**(8), 901–910 (2005)
20. Milnor, J.: On the concept of attractor. Commun. Math. Phys. **99**(2), 177–195 (1985)
21. Müller, S., Hofbauer, J., Endler, L., Flamm, C., Widder, S., Schuster, P.: A generalized model of the repressilator. J. Math. Biol. **53**(6), 905–937 (2006)
22. Reif, J.H.: Depth-first search is inherently sequential. Inf. Process. Lett. **20**(5), 229–234 (1985). https://doi.org/10.1016/0020-0190(85)90024-9
23. Sullivan, D., Williams, R.: On the homology of attractors. Topology **15**(3), 259–262 (1976)
24. Swat, M., Kel, A., Herzel, H.: Bifurcation analysis of the regulatory modules of the mammalian G1/S transition. Bioinformatics **20**(10), 1506–1511 (2004)
25. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972). https://doi.org/10.1137/0201010