Time Series Dependent Analysis of Unparametrized Thomas Networks

Hannes Klarner, Heike Siebert, and Alexander Bockmayr

Abstract—This paper is concerned with the analysis of labeled Thomas networks using discrete time series. It focuses on refining the given edge labels and on assessing the data quality. The results are aimed at being exploitable for experimental design and include the prediction of new activatory or inhibitory effects of given interactions and yet unobserved oscillations of specific components in between specific sampling intervals. On the formal side, we generalize the concept of edge labels and introduce a discrete time series interpretation. This interpretation features two original concepts: 1) Incomplete measurements are admissible, and 2) it allows qualitative assumptions about the changes in gene expression by means of monotonicity. On the computational side, we provide a Python script, *erda.py*, that automates the suggested workflow by model checking and constraint satisfaction. We illustrate the workflow by investigating the yeast network IRMA.

Index Terms—Time series analysis, model checking, temporal logic, biology and genetics, constraint satisfaction.

1 Introduction

In molecular biology, a regulatory network is a description of interactions between components. By assigning activity levels to the components and allowing interacting components to influence their activities depending on parameter values, such networks can be used to describe the system's dynamics in a state space. Since a full set of kinetic parameters is often not available, discrete modeling frameworks with finite parameter space have been suggested as an alternative to systems of differential equations.

Formal methods can help in determining suitable values for discrete parameters, translating available data into constraints on the set of all possible parameter choices, see e.g., Batt et al. [2] or Corblin et al. [8]. In this paper, we employ similar ideas to test assumptions about component interplay for consistency. In case of inconsistencies, new hypotheses are systematically derived that then can be investigated experimentally. We focus in particular on exploiting discrete time series data. Our method is specifically tailored to be able to cope with incomplete measurements resulting from experimental difficulties or low data quality. In further contrast to related work, we additionally use our method to evaluate the given experimental data by analyzing time series for potential ranges of poor sampling, making use of the notion of monotone transitions.

The paper is organized as follows: In Section 2 we recall the logical framework for regulatory networks and temporal logic. In Section 3 we introduce the notion of discrete time series as an ordered sequence of partial states. Section 4 elaborates a method of incorporating specific assumptions about monotonicity in between partial states. These are related to potential unobserved oscillations and can be used to evaluate the sufficiency of the provided data. In Section 5 we suggest a modeling workflow utilizing our method, assessing the modeling assumptions as well as the quality of a given time series in terms of its temporal resolution. Scalability and computational issues are discussed in Section 6. Here, we also introduce a constraint satisfaction preprocessing to model checking and discuss efficiency. Section 7 is a description of the available Python script that automates the workflow. We illustrate the procedure using an application example in Section 8, and conclude the paper discussing perspectives and future work. This paper is an extended version of the conference paper [13].

• *The authors are with the DFG Research Center Matheon, Freie Universität Berlin, Arnimallee 6, 14195 Berlin, Germany. E-mail: {Hannes.Klarner, Heike.Siebert, Alexander.Bockmayr}@FU-Berlin.de.*

Manuscript received 29 Nov. 2011; revised 10 Mar. 2012; accepted 28 Mar. 2012; published online 16 Apr. 2012.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBBSI-2011-11-0313.
Digital Object Identifier no. 10.1109/TCBB.2012.61.

2 Preliminaries

This section formally introduces the concepts needed for our method. These are based on the mathematical framework of Thomas presented in [19]. Throughout, discrete intervals will be denoted by

$$[a, b] := \{k \in \mathbb{N} \mid a \leq k \leq b\}, \quad \text{for } a, b \in \mathbb{N}.$$

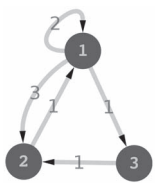
The in- and out-degrees of a vertex of a graph are denoted by $d_-(v)$ and $d_+(v)$, and its pre- and successor sets by $V_-(v)$ and $V_+(v)$, respectively.

2.1 Regulatory Networks

The discrete framework for modeling regulatory systems as introduced by Thomas in [19] consists of an edge-labeled digraph called regulatory network and a set of integer parameters.

Definition 2.1 (Regulatory Network). *A regulatory network $G = (V, E, t)$ is a directed graph with vertices $V := [1, n]$ for some fixed $n \in \mathbb{N}$, edges $E \subseteq V \times V$, maximal activity levels*

$$p : V \to [0, max(1, d_+(v))],$$



$$\begin{array}{ll} K_1(\{\}) = 0, & K_1(\{1\}) = 2 \\ K_1(\{2\}) = 3, & K_1(\{1,2\}) = 1 \\ K_2(\{\}) = 0, & K_2(\{1\}) = 0 \\ K_2(\{3\}) = 0, & K_2(\{1,3\}) = 1 \\ K_3(\{\}) = 0, & K_3(\{1\}) = 1 \end{array}$$

Fig. 1. Example network and parameter set.

and a function

$$t : E \to \mathbb{N}, \text{ with } t(u,v) \in [0, p(u)],$$

that assigns thresholds to the edges $e \in E$. Nodes are called components and edges are called interactions. For a component $v \in V$, a predecessor $w \in V_-(v)$ is called a regulator of v, and a subset of regulators $R \subseteq V_-(v)$ is called a regulatory context of v.

The vertices of the graph can be interpreted as variables taking values in the respective activity level interval $[0, p(v)]$. In the simplest case all variables are boolean. The edge labels are integers that represent thresholds above which regulatory interactions become effective.

Definition 2.2 (Parameter Set). *Given a regulatory network (V, E, t), a parameter set $K = \{K_v \mid v \in V\}$ is a set of functions*

$$K_v : 2^{V_-(v)} \to [0, p(v)].$$

K_v is also called v-parameter subset of K.

The network and parameter set in Fig. 1 will serve as a running example throughout the paper. Here, we choose the maximal activity levels $p(v) = d_+(v)$ for all vertices. Any collection of parameter sets of a regulatory network is called a parameter pool. In particular, we define:

Definition 2.3 (Parameter Space). *The collection of all parameter sets of a regulatory network (V, E, t) is denoted by*

$$\mathcal{K}(V, E, t) := \{K \mid K \text{ is a parameter set of } (V, E, t)\},$$

and called the parameter space of (V, E, t).

The number of sets in the parameter space depends on the maximal activity levels of the network components and their in-degrees:

$$|\mathcal{K}(V, E, t)| = \prod_{v \in V} (p(v) + 1)^{2^{d_-(v)}}.$$

The size of the parameter space of our running example is $|\mathcal{K}(V, E, t)| = 4^4 \cdot 2^4 \cdot 2^2 = 16{,}384$.

The dynamics of a regulatory network (V, E, t) with parameters K is represented by a directed graph, called the state transition graph. It can be thought of as the discrete analogue to all possible trajectories in the phase space of an ODE model. The nodes of this graph represent the discrete states of the system.

Definition 2.4 (State Space). *Given a regulatory network (V, E, t), the state space X is given by*

$$X = \prod_{v \in V} [0, p(v)].$$

To define the transitions between states it is convenient to turn the parameter set K into a function F on the state space X, where

$$F : X \to X, \quad x \mapsto F(x) = (f_1(x), \ldots, f_n(x)).$$

The image of x under component function f_v is defined to be a particular parameter $K_v(R)$. To choose this parameter we define the present regulators of v in a state x.

Definition 2.5 (Present Regulators). *Given a regulatory network (V, E, t) with parameters K and its associated state space X, the present regulators $R_v(x)$ of a component $v \in V$ in state $x \in X$ are*

$$R_v(x) := \{w \in V \mid (w, v) \in E \wedge x_w \geq t(w, v)\}.$$

The present regulators of v in state x are components w that regulate v and whose activity level in state x is above the threshold $t(w, v)$. This definition is the one given by Chaouiya et al. in [6]. With this notation the image of x under F is now defined to be

$$F(x) := (K_1(R_1(x)), \ldots, K_n(R_n(x))).$$

The present regulators of the components of the running example in state $x = (1, 1, 0)$ are $R_1(x) = \{2\}$, $R_2(x) = \emptyset$, and $R_3(x) = \{1\}$. Thus, $F(x) = (2, 0, 1)$ according to the table given in Fig. 1.

There are several strategies for obtaining transitions using F. Most common are synchronous, asynchronous, or priority strategies. The method described in this paper is designed to work with unitary asynchronous transition rules. A simple notation for the transitions of the unitary asynchronous state transition graph is achieved with the tendencies f'_v of the component functions f_v of $F = (f_1, \ldots, f_n)$, as suggested by Richard in [17].

Definition 2.6 (Tendencies). *The tendency f' of a component function $f_v : X \to \mathbb{N}$ is defined to be*

$$f'_v(x) = \begin{cases} 1: & f_v(x) - x_v > 0 \\ 0: & f_v(x) - x_v = 0 \\ -1: & f_v(x) - x_v < 0. \end{cases}$$

The tendency of component 2 in state $(0, 1, 0)$ of the example parameter set is $f'_2(0, 1, 0) = -1$.

Definition 2.7 (State Transition Graph). *Given a regulatory network (V, E, t) with parameter set K, the (unitary) asynchronous state transition graph is a directed graph (X, T), where the nodes X are the elements of the state space associated with the regulatory network and the edges T are transitions between states. We have $(x, y) \in T$ iff either $y = x = F(x)$ or*

$$\exists v : f'_v(x) \neq 0 \wedge y = x + e_v f'_v(x_v),$$

where e_v is the vth unit vector.

The behavior represented in such a state transition graph is nondeterministic. In a given state x there may be several $v \in V$ with $f'_v(x) \neq 0$, and therefore several y with $(x, y) \in T$. An example of this construction is given in Fig. 2.

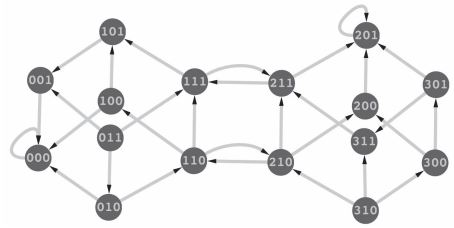


Fig. 2. The state transition graph of the example parameter set.

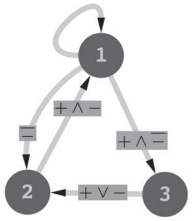


Fig. 3. Edge constraints, for example network, which lead to $|\mathcal{K}(V,E,t,s)| = 432$.

2.2 Edge Constraints

Now we consider information about the interaction type in the form of edge constraints. For example, interactions may be characterized as activating or inhibiting. Formally, edge constraints are additional edge labels that constrain the pool of feasible parameter sets for a regulatory network. Definitions for such edge constraints exist and are based on the observability or monotonicity of an interaction sign, usually either $\mathrm{a}+$ or $\mathrm{a}-$. We use a more extensive set of edge constraints to allow for a more precise characterization of individual interactions. Similar ideas can be found in [8].

The more general form of edge constraint is based on the observation that for a parameter set K, we can note for each interaction (w,v) if there is a regulatory context $R \subseteq V_-(v)$ such that adding w to R increases or decreases the value of K (as in [3]).

Definition 2.8 (Increase and Decrease). *Given a v-parameter subset K_v of a regulatory network (V,E,t), we define the boolean propositions $+$ and $-$ on the set of edges $(w,v) \in E$ by*

$$+(w,v) := \exists R \subseteq V_-(v) : K_v(R) < K_v(R \cup \{w\})$$
$$-(w,v) := \exists R \subseteq V_-(v) : K_v(R) > K_v(R \cup \{w\}).$$

It has been remarked by Richard in the context of deriving global interaction graphs from dynamics (see [17]) that such a comparison of parameter values with and without a regulator w is too weak to guarantee an effect observable in the state transition graph. For stronger results, a slightly more technical definition of increase and decrease could be introduced here. For boolean networks and for components v without self-regulation, i.e., $(v,v) \notin E$, the two definitions coincide.

For the parameter set of the running example, the values of $+$ and $-$ for each edge are the following:

$$\begin{array}{lll} +(1,1)=1, & +(3,2)=1, & +(1,3)=1, \\ +(2,1)=1, & +(1,2)=1, & -(1,3)=0, \\ -(1,1)=1, & -(3,2)=0, & \\ -(2,1)=1, & -(1,2)=0. & \end{array}$$

Instead of $\neg +$ and $\neg -$ we write $\overline{+}$ and $\overline{-}$. Simple logical expressions of these propositions are used to select parameter sets, by defining the following constraints:

Definition 2.9 (Edge Constraints). *A labeling function*

$$s : E' \subseteq E \to \{+, \overline{+}, -, \overline{-}, + \wedge -, + \vee -, \overline{+} \wedge -, + \wedge \overline{-}\},$$

on a subset $E' \subseteq E$ of the edge set of a regulatory network (V,E,t) is called edge constraint. A parameter set K satisfies the edge constraint s, if $s(w,v)$ is true for all $(w,v) \in E'$. In particular $\mathcal{K}(V,E,t,s)$ denotes all $K \in \mathcal{K}(V,E,t)$ that satisfy the edge constraint s.

If an edge is not labeled by s, then no constraints are placed on the respective parameter values. The different labels can be interpreted as follows: $+$ and $-$ signify that an activating or inhibiting effect has been experimentally observed. It is not precluded that the respective opposite effect may also occur, depending on specific cofactors. In contrast, $+ \wedge \overline{-}$ and $\overline{+} \wedge -$ are used if the target is strictly activated or inhibited. $\overline{+}$ and $\overline{-}$ allow for the possibility that there is no interaction at all, but if so it is not activating, respectively, inhibiting. If the character of an interaction is not known or questionable but some effect is assumed, e.g., based on binding site properties, $+ \vee -$ is used. Finally, $+ \wedge -$ applies when the target is activated in some context and inhibited in another, reflecting the importance of cofactors. In a drawing of a regulatory network, we place edge constraints on top of the corresponding edges, as in Fig. 3.

Other logical combinations or types of edge constraints could be considered, for example, labeling the components by max or min:

$$max(v) := \exists R \subseteq V_-(v) : K_v(R) = d_+(v)\ ,$$
$$min(v) := \exists R \subseteq V_-(v) : K_v(R) = 0\ .$$

However, the increase and decrease edge constraints already allow for a detailed description of interactions and suffice to illustrate the underlying method.

2.3 Model Checking

In this section, model checking is introduced as a means to analyze the state transition graph associated with a regulatory network. This has been proposed by various groups, see e.g., [3], [5], [8], [11], [15]. A Kripke structure or transition system is a state transition graph together with a labeling function that assigns atomic formulas to each node of the graph, which are defined to be true in this node. Computation Tree Logic (CTL) is a language that extends boolean propositions by temporal operators (see [12]). Boolean propositions can be evaluated at a node and so can CTL formulas. But, the temporal operators allow making statements about atoms that belong to other states, if there is a directed path in the transition graph from the first to the latter. Symbolic model checking is a fast method for finding the states in which a given CTL formula is true. We now will shortly review how to label the states of a transition graph, define the syntax of CTL and describe the semantics of CTL formulas.

A state transition graph (X,T) can naturally be interpreted as a Kripke structure. Each state $x = (x_1,\ldots,x_n)$ has n labels of the form $v_i \doteq x_i$. Here, we write $\doteq$ to distinguish syntactic from semantic equality. This labeling is extended to make the formula constructions in Section 4 possible.

Definition 2.10 (State Transition System). *Given a state transition graph (X, T) with variables $V := \{v_i \mid i \in [1, n]\}$, the set of atomic formulas consists of equalities*

$$\mathcal{P} := \left\{ \sum_{1 \le i \le n} k_i v_i \doteq k \mid v_i \in V, k \in [-N, N], k_i \in \{-1, 0, 1\} \right\},$$

where $N := \sum_{v \in V} p(v)$. Then, (X, T, L), with $L : X \to 2^{\mathcal{P}}$ and

$$L(x) = \left\{ \sum_{1 \le i \le n} k_i v_i \doteq k \mid k_i \in \{-1, 0, 1\}, k = \sum_{1 \le i \le n} k_i x_i \right\},$$

is the Kripke structure associated with the state transition graph (X, T).

A label $\sum_{1 \le i \le n} k_i v_i \doteq k$ captures simple expressions in the variables v_i that are true in state x. Model checking software like NuSMV (see [7]) can handle such expressions. The number N is included in the definition to emphasize that each node is only labeled with finitely many atoms. Here, are a few atoms of the state $x = (0, 2, 11)$: $v_1 \doteq 0$, $v_1 + v_2 \doteq 2$, $-v_1 - v_2 + v_3 \doteq 9$.

The following definition of the syntax of CTL formulas is restricted to the temporal operators **EF** and **E**[**U**] that are needed for the method described here.

Definition 2.11 (Syntax of CTL Fragment). *A CTL formula ϕ is defined inductively using the Backus Naur form. Let p be an element of the set of atomic formulas $\mathcal{P}$. Then,*

$$\phi ::= p \mid \phi \wedge \phi \mid \mathbf{EF}\phi \mid \mathbf{E}[\phi \mathbf{U} \phi].$$

Definition 2.12 (Semantics of CTL). *Given a Kripke structure (X, T, L), a state $x \in X$ and a CTL formula ϕ, the following rules determine whether ϕ is true in x:*

- *An atomic formula $p \in \mathcal{P}$ is true in x, if p is a label of x, i.e., $p \in L(x)$.*
- *$\phi \wedge \phi'$ is true in x, if ϕ is true in x and ϕ' is true in x.*
- *$\mathbf{EF}\phi$ is true in x, if ϕ is true in x or if there is a path $(x, x^1, \ldots, x^n)$ in (X, T) with $n \ge 1$ and ϕ is true in x^n.*
- *$\mathbf{E}[\phi \ \mathbf{U} \ \phi']$ is true in x, if ϕ' is true in x or if there is a path $(x, x^1, \ldots, x^n)$ in (X, T) with $n \ge 1$ such that ϕ is true in x and x^i for $1 \le i \le n - 1$ and ϕ' is true in x^n.*

In the following sections, CTL formulas will be used to select parameter sets from given parameter pools. The selection is based on the existence of a state satisfying the formula.

Definition 2.13 (ϕ-Acceptable Parameter Sets). *Given a CTL formula ϕ, the collection of parameter sets of a regulatory network (V, E, t) whose associated transition system contains a state in which ϕ is true is denoted by*

$$\mathcal{K}(V, E, t, \phi) := \{K \in \mathcal{K}(V, E, t) \mid \exists x \in X : \phi \text{ is true in } x\}.$$

Sometimes a transition system is said to satisfy a CTL formula ϕ, if ϕ is true in all states. Since we want to query the existence of paths starting in some state of the graph, the above definition is used.

3 Discrete Time Series

A discrete time series for a regulatory network can be obtained by discretizing real-valued experimental data or by qualitative observations about regulatory components. The issue of choosing a suitable discretization method for experimental data is crucial (see, e.g., [10]), but is not the subject of this paper. Under the assumption that the regulation behaves switch like regarding the regulator concentration, one ideally has to estimate the threshold below which the regulator is not effective and above which it becomes effective.

If estimation is not possible, statistical approaches can be used, for example, mean clustering, scan-statistic, or edge gradient methods as described by Shmulevich and Zhang in [18]. There is also a software implementation for the GNU project R called BoolNet by Müssel et al. [16] which automates such discretization. BoolNet will be used in Section 8 to discretize the expression data of the IRMA network [4].

Including qualitative observations in the time series is a strength of discrete modeling as it may be hard to translate such assumptions into quantitative data required for continuous models.

Mathematically, a discrete time series is a matrix where rows are measurements and columns are observations for one component. Data points with questionable discretization results for certain components or observations known to be imprecise may be recorded as uncertain by the sign $\perp$. In practice, this has the advantage of deriving results based on varying levels of certainty.

Definition 3.1 (Time series). *A discrete time series with m measurements of n substances is a matrix $A \in N^{m \times n}$, where the entries of A are elements of $N := \mathbb{N} \cup \{\perp\}$ and additionally $\forall i \in [1, m] : \ \exists j \in [1, n] : \ a_{i,j} \neq \perp$.*

The condition ensures that measurements without supportable entries are not included in the time series. It should also be remarked that throughout we are using matrices merely as a means to presenting data and do not perform any algebraic manipulation on them.

As a discrete time series for the running example, including four measurements and three imprecise observations, we choose

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 2 & \perp & 1 \\ \perp & 1 & 0 \\ 3 & 0 & \perp \end{pmatrix}.$$

A time series will be interpreted as encoding discrete paths. To define these paths, the partial state formulas, one for each measurement, are derived. The definition uses the set of indices whose variables are not equal to -1. Thus, uncertain variables will be excluded from the description of the paths.

Definition 3.2 (Partial states). *Given a time series $A \in N^{m \times n}$, the partial state formula of measurement $i \in [1, m]$ is*

$$\sigma_i := \bigwedge_{j \in M_i} (v_j \doteq a_{i,j}), where$$

$$M_i := \{j \in [1,n] \mid a_{i,j} \neq \perp\}.$$

A partial state formula may be true in a set of states, depending on how many variables are uncertain. The paths encoded in a time series are then all paths that connect the partial states in the given order. A state transition graph that contains at least one such path is said to be able to reproduce the time series.

Definition 3.3 (Reproducing a Time Series). *A state transition graph (X,T) can reproduce a time series $A = (a_{i,j}) \in N^{m\times n}$ if there is a path $(x^1,\ldots,x^k)$ in (X,T) such that the index sequence $(1,\ldots,k)$ has a subsequence $(r_1,\ldots,r_m)$ satisfying for each $1 \leq i \leq m$ that σ_i is true in x^{r_i}.*

We say a parameter set can reproduce a time series, if this holds for the corresponding state transition graph.

The sequence of states $(x^1,\ldots,x^k)$ can be thought of as a simulation of the regulatory network from the initial state x^1. An intuitive CTL formula can be used to check if a parameter set can reproduce a time series. Such a formula is a nested sequence of partial state formulas connected via the predicates **EF**:

$$\sigma_1 \wedge \mathbf{EF}[\sigma_2 \wedge \mathbf{EF}[\ldots \sigma_{m-1} \wedge \mathbf{EF}[\sigma_m]\ldots]].$$

Since a time series is a linear sequence of states, an equivalent LTL encoding is possible. The same is true for the construction in the next section. This may be computationally exploited by using LTL model checkers, or even colored LTL model checking as suggested by Barnat et al. [1].

4 The Monotone Path Formulas

In this section, the paths encoded in a time series are characterized with regard to monotonicity in between successive measurements. The motivation for this is to take into account assumptions about the ratio of time elapsed between measurements on the one hand, and rates of change of components on the other. Intuitively, if for a substance the time elapsed between successive measurements is small compared to its rate of change, then we would expect its concentration to change monotonously, i.e., without oscillations.

To encode these ratios for each variable and at each measurement, we define a matrix to specify exactly which parts of the path should be monotone.

Definition 4.1 (Monotonicity Matrix). *Given a discrete time series $A \in N^{m\times n}$, a monotonicity matrix of A is any matrix $B = (b_{i,j}) \in \{0,1\}^{m-1\times n}$ such that*

$$\forall i,j: \quad b_{i,j} = 1 \Longrightarrow (a_{i,j} \neq \perp \wedge a_{i+1,j} \neq \perp).$$

We say that variable j is specified to be monotone at measurement i, iff $b_{i,j} = 1$.

A time series and a monotonicity matrix define the following partially monotone paths. For technical reasons regarding the CTL construction in Definition 4.5, we require that the path begins in a state representing the first and ends in one representing the last measurement.

Definition 4.2 (A-B-Monotone Paths). *Given a discrete time series $A \in N^{m\times n}$ together with a monotonicity matrix B, and a state transition graph (X,T), a path $(x^1,\ldots,x^r)$ in (X,T) is A-B-monotone, if there is a subsequence $(r_1,\ldots,r_m)$ of $(1,\ldots,r)$ with $r_1 = 1, r_m = r$ such that the following two properties hold. First*

$$a_{i,j} \neq \perp \implies x_j^{r_i} = a_{i,j}.$$

Second, for the variables j specified to be monotone at measurement i

$$\forall t \in [r_i, r_{i+1}-1] : \begin{cases} x_j^t \leq x_j^{t+1} : & \text{if} x_j^{r_i} \leq x_j^{r_{i+1}} \\ x_j^t \geq x_j^{t+1} : & \text{if} x_j^{r_i} > x_j^{r_{i+1}}. \end{cases}$$

A monotonicity matrix for the example time series A is

$$B = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

and an example of an A-B-monotone path is

$$((0,1,0),(1,1,0),(1,1,1),(2,1,1),(2,1,0),(3,1,0),(3,0,0)).$$

Again, a CTL formula is constructed to check the existence of an A-B-monotone path in a transition system. This formula is specifically designed for asynchronous transition graphs. It exploits the observation that for each couple of successive measurements, there is an expression $\sum k_i v_i$ in the marked monotone variables v_i that is increasing along any A-B-monotone path. To determine this expression we need to consider the variables that increase and decrease separately.

Definition 4.3 (Index Sets). *Given a discrete time series $A \in N^{m\times n}$ and a monotonicity matrix $B \in \{0,1\}^{m-1\times n}$, we define for each $i \in [1, m-1]$ the index sets M_i^+ and M_i^- of increasing and decreasing variables, respectively:*

$$\begin{aligned} M_i^+ &:= \{j \in [1,n] \mid b_{i,j} = 1 \wedge \ a_{i,j} \leq a_{i+1,j}\}, \\ M_i^- &:= \{j \in [1,n] \mid b_{i,j} = 1 \wedge \ a_{i,j} > a_{i+1,j}\}. \end{aligned}$$

Now we can construct the increasing expression mentioned before, define its initial value and by how much it has to increase in between measurements.

Definition 4.4 (Increasing Expression). *The increasing expression $V_i = V_i(v_1,\ldots,v_n)$, the initial value C_i and the distance d_i for $i \in [1, m-1]$ are defined to be*

$$\begin{aligned} V_i &:= \sum_{j\in M_i^+} v_j + \sum_{j\in M_i^-} (a_{i,j} - v_j), \\ C_i &:= \sum_{j\in M_i^+} a_{i,j}, \\ d_i &:= \sum_{j\in M_i^+\cup M_i^-} |a_{i,j} - a_{i+1,j}|. \end{aligned}$$

In a state satisfying the partial state formula σ_i, the atomic formula $V_i \doteq C_i$ is true. The following A-B-monotone path formula asserts that $V_i \doteq C_i$ increases one by one until $V_i \doteq C_i + d_i$ and σ_{i+1} are true. To deal with the nested structure of the formula, it is defined recursively.

Definition 4.5 (A-B-Monotone Path Formula). *The A-B-monotone path formula $\phi_{A,B}$ for a time series $A \in N^{m\times n}$ and monotonicity matrix B is constructed recursively using the formulas $\rho_i, i \in [1,m]$. Let*

$$\rho_1 := \sigma_m,$$

and for $i \in [1, m-1]$

$$\rho_{i+1} := \begin{cases} \sigma_{m-i} \wedge \mathbf{EF}[\rho_i] & if\ M^+_{m-i} \cup M^-_{m-i} = \emptyset \\ \sigma_{m-i} \wedge \gamma^{m-i}_{d_{m-i}+1} & if\ M^+_{m-i} \cup M^-_{m-i} \neq \emptyset. \end{cases}$$

Here

$$\gamma^{m-i}_1 := \mathbf{E}[(V_{m-i} \doteq C_{m-i} + d_{m-i})\ \mathbf{U} \rho_i],$$

and if $d_{m-i} \geq 1$ *then*

$$\gamma^{m-i}_{t+1} := \mathbf{E}[(V_{m-i} \doteq C_{m-i} + d_{m-i} - t)\ \mathbf{U}\ \gamma^{m-i}_t],$$

for $t \in [1, d_{m-i}]$. *Finally, define* $\phi_{A,B} := \rho_m$.

From the above definition, a pseudocode algorithm for the construction of an A-B-monotone path formula is derived:

```
ρ_1 := σ_m
for i = 1 to m − 1 do
  if M^+_{m−i} ∪ M^−_{m−i} = ∅ then
    ρ_{i+1} := σ_{m−i} ∧ EF[ρ_i]
  else
    γ^{m−i}_1 := E[(V_{m−i} ≐ C_{m−i} + d_{m−i})U ρ_i]
    if d_{m−i} ≥ 1 then
      for t = 1 to d_{m−i} do
        γ^{m−i}_{t+1} := E[(V_{m−i} ≐ C_{m−i} + d_{m−i} − t)U γ^{m−i}_t]
    end for
  end if
  ρ_{i+1} := σ_{m−i} ∧ γ^{m−i}_{d_{m−i}+1}
end if
end for
```

Next we show that this formula characterizes the existence of an A-B-monotone path.

Theorem 4.6 (Correctness). *Given an asynchronous state transition graph* (X, T), *its associated state transition system* (X, T, L), *and a discrete time series* $A \in N^{m \times n}$ *together with a monotonicity matrix* $B \in \{0, 1\}^{m-1 \times n}$, *the A-B-monotone path formula is true in* (X, T, L) *if and only if there is an A-B-monotone path in* (X, T).

Proof. By the recursive structure of ρ_m it is sufficient to consider a matrix A with just two rows. For further simplicity assume there are only increasing variables ($M^-_1 = \emptyset$). The mixed case follows the same reasoning, because every $j \in M^-_1$ appears as $v'_j := x^1_j - v_j$ in V_1 and v'_j increases, if v_j decreases.

First, we want to show that the existence of an A-B-monotone path $(x^1, \ldots, x^r)$ in (X, T) implies that ρ_2 is true in x^1. For each $t \in [1, r-1]$ we have $V_1(x^t) \leq V_1(x^{t+1})$, because V_1 is the sum of variables that increase along that path. The difference $V_1(x^{t+1}) - V_1(x^t)$ is at most 1 since T contains only unitary asynchronous transitions. So there must be a partition of $[1, r]$ into $d_1 + 1$ intervals, where $d_1 := \sum_{j \in M^+_1}(x^r_j - x^1_j)$, such that V_1 is constant on each interval and increases by 1 from one interval to the next. On all states x of the first interval the formula $V_1(x) \doteq C_1$ is true and on all states x of the last interval the formula $V_1(x) \doteq C_1 + d_1$ is true. Therefore γ^1_t for $t \in [1, d_1 + 1]$ is true on the tth interval, counted from right to left and hence ρ_2 is true in x^1.

Second, we want to show that ρ_2 is true in $x \in X$ implies that there is an A-B-monotone path in (X, T). Since ρ_2 is true in $x^1 := x$ there is a path $(x^1, \ldots, x^r)$ in (X, T) such that σ_1 is true in x^1 and $\rho_1 \doteq \sigma_2$ is true in x^r, which is the first property of an A-B-monotone path. Furthermore, $[1, r]$ can be partitioned into $d_1 + 1$ intervals such that γ^1_t is true in the tth interval counted from right to left. Therefore, V_1 increases by 1 from one interval to the next. Since T contains only unitary asynchronous transitions, there is exactly one variable $j \in M^+_1$ that increases by 1 from one interval to the next. Therefore, $x^k_j \leq x^{k+1}_j$ for all $k \in [1, r-1]$ and $j \in M^+_1$ which is the second property of an A-B-monotone path. So the path $(x^1, \ldots, x^r)$ is A-B-monotone. □

5 Workflow

In this section, we introduce a methodology to analyze compatibility of a regulatory network and a given time series.

Let us consider a regulatory network, possibly including edge constraints, a time series, and a monotonicity matrix (consisting only of zero entries in case no monotonicity assumptions are made). As a first step, we check whether there are parameter sets that reproduce the time series, i.e., we compute the parameter pool $\mathcal{K}(V, E, t, s, \phi_{A,B})$. This parameter pool, computed for our running example, consists of eight parameter sets. If the model checking procedure returns a unique parameter set, we can proceed with the analysis of the model. However, this case will only occur very rarely. More commonly, the procedure either returns a large pool of parameter sets or no set at all. In the following, we look at both cases more closely.

5.1 Characterizing Model Pools

If the parameter pool contains many parameter sets, the information encoded in the network and the time series was not sufficient to determine a unique specified model. One possibility to deal with this difficulty is to choose a model from the pool using meaningful criteria, e.g., some notion of minimality as in [9]. A different approach is to characterize the parameter pool in order to derive information about the system strongly supported by the available data. We propose ideas in line with the second approach. One characteristic of a model pool are parameter values that are identical across all parameter sets. Such values may allow for new insights into how a component behaves under the influence of several regulators, clarifying synergies and redundancies in the network.

Definition 5.1 (Determined Parameter Values). *Given a parameter pool* $\mathcal{K}$, *the value of a component* v *in a regulatory context* $R \subseteq V_-(v)$ *is determined if there is a* $p \in [0, p(v)]$ *such that*

$$\forall K \in \mathcal{K} : K_v(R) = p.$$

This idea can be extended to finding the range of values for each component and regulatory context.

For our running example, Fig. 4 displays these ranges.

R	$K_1(R)$	R	$K_2(R)$	R	$K_3(R)$
$\emptyset$	0	$\emptyset$	1	$\emptyset$	0
$\{1\}$	3	$\{1\}$	1	$\{1\}$	1
$\{2\}$	2,3	$\{3\}$	0		
$\{1,2\}$	0,1	$\{1,3\}$	0,1		

Fig. 4. Value ranges of $\mathcal{K}(V, E, t, s, \phi_{A,B})$ of running example.

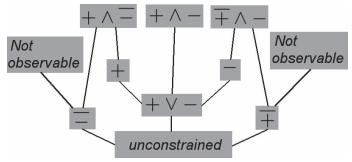


Fig. 5. Hierarchy of edge constraints with stricter labels above weaker ones. "Not observable" is used to emphasize the meaning of $\overline{+}\wedge=$.

Even if the parameters for a given component are not completely determined, we can still try to extract further information. To get an idea about the different behaviors that a component can have in a parameter pool, we count the v-local parameter sets in $\mathcal{K}$.

Definition 5.2 (Behaviors). *Given a parameter pool $\mathcal{K}$ of a regulatory network (V, E, t), the behaviors $\mathcal{K}_v$ of component $v \in V$ are the set of v-local parameter sets in $\mathcal{K}$,*

$$\mathcal{K}_v := \{K_v \mid K \in \mathcal{K}\}.$$

This information can be used to study how components are tuned to work together in reproducing a time series. If any combination of component behaviors is a parameter set in the pool, then the components are said to be independent.

Definition 5.3 (Independence). *A parameter pool $\mathcal{K}$ consists of independent components, if*

$$\prod_{v \in V} |\mathcal{K}_v| = |\mathcal{K}|.$$

In case dependencies exist, the behavior of one component may rule out or enforce a certain behavior of another component. In most cases, time series will create dependent parameter pools, as in the application example in Section 8.

For our running example, there are four behaviors for component 1, two behaviors for component 2, and one behavior for component 3. This pool is therefore independent, since $8 = 4 \cdot 2 \cdot 1$.

Further characterization of the parameter pool could study which behaviors do not appear together and try to identify components and regulatory contexts, which, if determined, would lead to the steepest reduction in feasible parameter sets. Identifying such contexts could be used to design experiments that reduce the number of feasible parameter sets in the fastest possible way.

Characterization of the parameter pool can also focus on the edge labels. They can be arranged into a logical implication hierarchy. For example, $+\wedge=$ implies $=$ and we thus place $+\wedge=$ above $=$ in the hierarchy diagram in Fig. 5. For each unlabeled edge of the regulatory network and edges carrying one of the constraints that may be strengthened $(+, -, +\vee-, =, \overline{+})$, we determine the strictest label that is true for all parameter sets. This may lead to determining an effect of a regulator on its target that was formerly not known. An edge may, for example, be included in a network, because the source component is known to bind to the target component's promoter, but without any knowledge of the effect this binding has (i.e., with label $+\vee-$). With a time series this label may be sharpened to $+$ and thus hypothesize an activation.

For the running example there are two edge refinements: the previously unconstrained self-interaction of component 1 is hypothesized to be activating and inhibiting, i.e., $+\wedge-$. The interaction between components 2 and 3, which we assumed to be observable $(+\vee-)$, is sharpened to be inhibiting only, i.e., $-\wedge\overline{+}$.

5.2 Evaluating the Time Series

An ideal sampling frequency would result in a discrete time series capturing all value changes of the components, but usually data points are rather sparse. In order to understand the underlying system, we need to know whether the sampling was sufficient to capture its essential behavior. Here, we focus on determining potential oscillatory behavior not inferable from the time series due to coarse sampling.

Consider a network, a time series and a monotonicity matrix that are compatible, i.e., the corresponding parameter pool is not empty. We start with the assumption that the time series is sufficient to exclude the possibility of undetected oscillatory behavior. Intuitively, if sufficiently many measurements were made, it can be assumed that all variables are monotone at all measurements.

Definition 5.4 (Best Fit). *Given a regulatory network (V, E, t), a time series A and the monotonicity matrix B, where*

$$b_{i,j} = \begin{cases} 1: & a_{i,j}, a_{i+1,j} \neq \perp \\ 0: & \text{else}, \end{cases}$$

a parameter set that satisfies the A-B-monotone path formula is called a best fit of (V, E, t) to A.

The eight parameter sets in $\mathcal{K}(V, E, t, s, \phi_{A,B})$ of our running example all turn out to be best fits to A.

Recall that the entries or positions (i, j) of B represent the value transition of the jth component from measurement i to measurement $(i+1)$, and that the entry 1 signifies a monotone value change. If no best fits of (V, E, t) to A exist, we can be sure that there is a set of positions of B, such that all parameter sets in the considered pool produce at least one unobserved oscillation in one of the positions. In these positions the temporal resolution of A is too coarse to capture the behavior of the network. A trivial such set is the set of all positions, but there may be a smaller set, ideally with only a single position. Starting with the originally considered monotonicity matrix B, a heuristic approach to finding a nontrivial set is to introduce additional monotonicity constraints position by position. If such an added constraint does not result in a reduction of the parameter pool, we discard the corresponding position, since all models agree with the assumed monotonicity for that position, and we need no extra sampling between the corresponding data points. We

introduce a measure for the impact of an additional monotonicity constraint as follows:

Definition 5.5 (Selectivity). *Given a regulatory network (V, E, t), a time series A and a monotonicity matrix B, we define for each $1 \leq i \leq m, 1 \leq j \leq n$ such that*

$$b_{i,j} = 0 \text{ and } a_{i,j}, a_{i+1,j} \neq \perp,$$

the monotonicity matrix B' by

$$b'_{i',j'} := \begin{cases} 1: & i' = i, j' = j \\ b_{i,j}: & \text{else,} \end{cases}$$

and the selectivity of position (i, j) by

$$S(i,j) := 1 - \frac{|\mathcal{K}(V, E, t, \phi_{A,B'})|}{|\mathcal{K}(V, E, t, \phi_{A,B})|}.$$

All positions that have selectivity 1 hypothesize obligatory oscillations of component j in between measurements i and $i+1$, which indicates the need for additional data points between the measurements. If no such positions exist, we choose the set $\{(i,j) \mid S(i,j) > 0\}$ as places of interest for new measurements.

Since there are best fits for our running example, we can conclude without model checking that there is no position with selectivity 1. In fact, the matrices A and B allow deriving only one monotonicity matrix B'. It places the additional monotonicity condition on position $(3, 1)$, i.e., $B_{3,1} = 1$. The selectivity of this position is $S(3, 1) = 0$, which means that all parameter sets in the pool support a path that is monotone for component 2 in between measurements 3 and 4.

To illustrate the concept of selectivity for a more interesting case, we change the matrix A slightly to A':

$$A' = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 0 & 1 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}.$$

The corresponding parameter pool $K(V, E, t, \phi_{A',B})$ contains only four parameter sets. Now, there are six positions for which we can compute the respective selectivities. In matrix notation, they are

$$S(B) = \begin{pmatrix} \perp & 0 & \perp \\ 1 & 1 & \perp \\ 0 & 0 & 0 \end{pmatrix}.$$

These results are exploitable for experimental design, as they predict yet unobserved oscillations at positions with selectivity 1, i.e., of the components 1 and 2 each in between the measurements 2 and 3.

5.3 Reviewing Structure and Data

So far we have considered the case that we have no contradictions in our modeling assumptions and data, resulting in viable choices of parameter sets. If a network is not compatible with a time series and the possibly additionally provided monotonicity matrix, i.e., the corresponding parameter pool is empty, there are two possible lines of investigation, depending on whether the correctness of the network structure or of the data is questioned. In both cases, the idea is to check what minimal changes can lead to compatibility.

Regarding the structure, we may in a first step relax the constraints on the interactions and instead label every edge with the observability label $+ \vee -$. Thus, we include no assumptions on the character of an interaction, but only require it to be observable. We then test if the weakened assumptions result in a nonempty parameter pool. Regarding the data, we proceed similarly by first lifting monotonicity constraints in B (if there are any) and then replacing particular values in A to be imprecise.

6 Efficiency and Scalability

6.1 Scalability

The computational steps in the workflow are 1) to exhaustively generate all parameter sets satisfying the edge constraints (not the whole parameter space), 2) to translate a parameter set into a model checker input file, and 3) to pass it to a model checker, together with the A-B-monotone path formula. For model checking we use NuSMV ([7], see also [3] and [2]). For computation of the parameter sets, we apply a backtracking algorithm with failure on edge label violation.

Regarding scalability and computation times, we first note that the state space is exponential in the number of components, which places a strong limit on the possible number. Second, we compute a large part of the parameter space, depending on how restricting the edge labels are. Efficient algorithms considering partial parametrizations only have been introduced for piecewise linear ODE models (see [2]). Similar approaches would be desirable for the Thomas formalism.

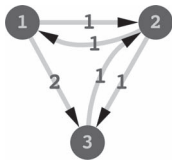
As standing, analysis is limited to structures of about 30,000 states, e.g., 15 binary components or nine ternary components. For such models the time per model check is impacted considerably by the nesting depth of a given CTL formula, which in our case increases linearly with the length of a time series and monotonicity constraints. Model checking a 30,000 state model and a time series of five measurements takes about 1 second on a 2.27 GHz Laptop.

Given these restrictions imposed by the time per model check, the computation time for the parameter sets is negligible. However, it should be noted that even with the most restrictive edge labels ($+ \wedge =$ and $- \wedge \overline{+}$) on edges targeting a binary component, there are already 6,894 local parameter sets for only five regulators. For a ternary component, the number of such regulators is limited to 4, resulting in 7,008 local parameter sets.

6.2 Constraint Satisfaction and Monotone Paths

As discussed, the computationally most expensive step in the workflow is executing a model check. It is therefore desirable to run a preprocessing that marks at least some parameter sets correctly as either acceptable or rejectable with respect to a given CTL specification. This is useful, if the time the preprocessing needs to mark parameter sets is less than the time required for a full model check.

In this section, we introduce such a method by deriving from the matrices A and B and the regulatory network (V, E, t) a necessary condition for a parameter set to be able to reproduce an A-B monotone path. This condition is



$$A := \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$B := \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

Fig. 6. Example network for constraint preprocessing.

formulated as a constraint satisfaction problem on the parameter values.

Before we define the constraints, we motivate the idea by making two observations. The first observation is about the existence of a parameter value that is a witness to an observed behavior. The second is about the activity range of a component along an A-B-monotone path. Then, we discuss the effect that a restricted activity of a component has on the witnesses of its targets in the regulatory network. Finally, we proceed to define the constraints.

Assume that a regulatory network, a discrete time series A with two measurements and a monotonicity matrix B are fixed. The first observation is that if the two sampled activities of a component v are precise but not equal, i.e., $a_{1,v}, a_{2,v} \neq \perp$ and $a_{1,v} \neq a_{2,v}$, we deduce that there must be some regulatory context $R \subseteq V_-(v)$ such that $K_v(R)$ lies on the right side of the second sample $a_{2,v}$. By *on the right side* we mean that, if A records an increase in v, we deduce $\exists R : K_v(R) \in [a_{2,v}, p(v)]$, and if a decrease is observed, then $\exists R : K_v(R) \in [0, a_{2,v}]$. A context R that fulfills this condition is called a *witness* to the change observed in v. It is easy to see that each change in A requires a witness.

On its own, this condition is not very strong. A parameter set of a boolean network, e.g., fails this test only when all parameter values are not on the right side. For example, if 0 is not on the right side for v, then the condition is $\forall R : K_v(R) = 0$, which is a single valuation of K_v, and all other valuations remain feasible. We want to sharpen this condition by restricting the set from which the witness R can be drawn by analyzing the corresponding regulators of v.

For each regulator $u \in V_-(v)$ that is marked as monotone by $b_{1,u} = 1$ (which by definition implies precise measurements for u), we can deduce that along an A-B monotone path its activity remains in the interval $[u_{min}, u_{max}]$, where

$$u_{min} := \min(a_{1,u}, a_{2,u}), \text{ and } u_{max} := \max(a_{1,u}, a_{2,u}).$$

The extreme cases for this interval are when $u_{min} = u_{max}$, and the activity of u is constant, and $[u_{min}, u_{max}] = [0, p(u)]$ and no information about the activity can be obtained.

Now we compare the threshold $t(u, v)$ with the lower and upper end points of the interval. Whenever we find that the threshold lies strictly below the lower end point (resp. is equal to or above the upper end point), we deduce that u is an *effective* (resp. *ineffective*) *regulator* of v along an admissible path. Hence, a witness R to the observed change in v must (resp. must not) contain u. Depending on the measurements and network topology, this is a strong restriction, since each such u halves the set of potential witnesses.

As an example, consider the 3-component network, time series, and monotonicity matrix in Fig. 6.

The observed increase in 3 requires a witness $R \subseteq V_-(3)$ such that $K_3(R) \in [1, 1]$, i.e., $K_3(R) = 1$. Since $b_{1,1} = 1$ the activity range of component 1 is restricted to the interval $[0, 1]$. Additionally, the threshold $t(1, 3) = 2$ is strictly above the upper end point of the interval and we deduce that a witness R must not contain the regulator 1, i.e., it is ineffective. So without performing a model check we can discard all parameter sets K where $K_3(R) = 0$ for $R \in \{\{2\}, \emptyset\}$. In the example this means that instead of performing 2,304 model checks, we solve the constraint satisfaction problem and perform 1,728 model checks on the remaining parameter sets.

Let us now define the constraint satisfaction problem.

Definition 6.1 (Some-In-Set). *A parameter set K satisfies the constraint $Some\text{-}In\text{-}Set(v, S, \mathcal{R})$ for a component v, a set $S \subseteq [0, p(v)]$, and contexts $\mathcal{R} \subseteq 2^{V_-(v)}$, iff $K_v(\mathcal{R}) \cap S \neq \emptyset$.*

The suggested preprocessing performs the following tasks. For each pair of consecutive measurements, we 1) select the components that change, 2) compute the activity ranges of each regulator of each changing component, and 3) add one $Some\text{-}In\text{-}Set$ constraint to the list of constraints that an acceptable parameter set has to satisfy.

The reason why model checking cannot be avoided altogether is, that the condition is not sufficient. So far we have not found a compact constraint satisfaction formulation on the parameter values of a parameter set that is equivalent to the existence of an A-B monotone path in its transition system.

6.3 Efficiency of Constraint Satisfaction

In this section, we investigate the efficiency of monotonicity constraints in terms of avoided model checks. Given that the constraint problems arising from the networks that we deem computationally feasible (see Section 6.1) have at most 32 variables, one for each context of the component, and domains of size at most 4, solving a $Some\text{-}In\text{-}Set$ constraint is preferable to executing a model check. In principle, each pair of consecutive rows in A may pose one $Some\text{-}In\text{-}Set(v, S, \mathcal{R})$ constraint for each $v \in V$, which reduces the set $\mathcal{K}_v$. The number of model checks that need to be executed without monotonicity constraints is equal to the number of parameter sets

$$\mathcal{K}(V, E, t, s) = \prod_v \mathcal{K}_v.$$

Because of this product structure, the factor by which we reduce $\mathcal{K}_v$ is equal to the factor by which we reduce the required number of total model checks. So we have to ask by what factor a $Some\text{-}In\text{-}Set(v, S, \mathcal{R})$ constraint reduces $|\mathcal{K}_v|$.

In general this is a hard question, but for the special case of independent parameter values, i.e., unlabeled interactions targeting v, we can give a formula. To simplify notation denote $s := |S|$ and $r := |\mathcal{R}|$. The reduction factor f is then equal to

$$f(v, s, r) := 1 - \left(1 - \frac{s}{p(v) + 1}\right)^r.$$

To better understand this formula, we can interpret it as the probability that at least one context from $\mathcal{R}$ hits the set S, which is the complement of all contexts missing S.

A closed formula for the labeled case would include, as a special case, the size of the set of all boolean monotonic

TABLE 1
Reduction of v-Behaviors, Depending on Effective and Ineffective Regulators in R and Values in S

Effective	Ineffective	S	$\lvert\mathcal{R}\rvert$	$\mathcal{K}_v$	Reduction	$f(v,s,r)$
3	0	0	1	37	0.37	0.50
2	1	0	1	45	0.45	0.50
1	2	0	1	56	0.55	0.50
0	3	0	1	64	0.63	0.50
2	0	0	2	70	0.69	0.75
1	1	0	2	82	0.81	0.75
0	2	0	2	89	0.88	0.75
1	0	0	4	97	0.96	0.94
0	1	0	4	101	1.00	0.94
0	0	0	8	101	1.00	1.00

The interactions targeting v are labeled with $+$.

functions in a fixed number of variables, which is an open problem (*computation of Dedekind number*). In general, the reduction factor f is neither an upper nor a lower bound of the reduction in the edge labeled case, but as the following tables illustrate, it may serve as a good indicator.

To cope with the many unknowns, $p(v)$, $|V_-(v)|$, the edge labels s, the contexts $\mathcal{R}$ and S, we decided to consider three cases that differ in the edge labels, but have the other parameters fixed to give an intuition about the efficiency of the preprocessing. Throughout $p(v) = 1$ and $|V_-(v)| = 3$. To keep the tables concise, using symmetries, the three interactions have the same label in each example. Recall that $\mathcal{R}$ is defined by stating for each regulator of v, if it is effective, ineffective, or undetermined (see Section 6.2). Instead of distinguishing which particular regulator is in which of the three conditions, we group cases by stating how many are in each class. Because the three labels are identical, all sets $\mathcal{R}$ with the same counts have the same reduction factor.

In Table 1, we are considering a component v that has three *activating* regulators (interaction label $+$). For such a component there are 101 behaviors. Additionally, it is symmetric under simultaneously swapping effective with ineffective regulators and $S = \{0\}$ with $S = \{1\}$. Hence, rows with $S = \{1\}$ are omitted. The table is ascending in the reduction factor.

In Table 2, we are considering a component that has three *activating only* regulators (label $+ \wedge =$). For such a component there are only nine behaviors. The same symmetry applies as above, but we have also grouped rows with identical reduction factors by specifying a range of ineffective regulators. The first row, with reduction factor 0.00, shows that the constraint satisfaction approach is suitable for detecting local inconsistencies that prevent global compatibility with a time series.

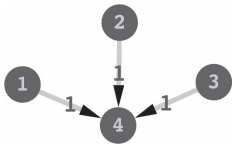


Fig. 7. Vertex-induced subgraph of example network.

To illustrate this, we consider the following successive measurements A and monotonicity matrix B

$$A := \begin{pmatrix} 1 & 1 & 2 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix}, \quad B := \begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix},$$

in a regulatory network whose vertex-induced subgraph of the four respective vertices with interaction labels $+ \wedge =$ is given in Fig. 7.

Independently of how this subgraph is embedded in a larger network, and independently of the values of the remaining measurements, the resulting parameter set pool will be empty. Without the constraint satisfaction preprocessing, a model check for each global parameter set would be necessary to arrive at the same conclusion.

For the third and final example we consider three regulators with *observable* effects on v (label $+ \vee -$). For such a component there are 256 behaviors. This time, the reduction factors are invariant under the sum of effective and ineffective regulators in $\mathcal{R}$. They are also identical for $S = \{0\}$ and $S = \{1\}$. See Table 3 for reduction factors of this example.

In conclusion, under fairly weak local assumptions about the time series and monotonicity matrix, the constraint satisfaction preprocessing strongly reduces the required model checks to determine the compatible parameter set pool. The three tables above give the reduction factors in the setting of a boolean component that is regulated by interactions of identical type (activating, activating only, observable). Similar tables could be given for larger values $p(v)$ and mixed interaction types targeting v.

The question remains how the reduction factors combine, if there are several $Some\text{-}In\text{-}Set$ constraints. In the first case, for component disjunct constraints, the factors simply multiply to give a global reduction factor. In the other case, the product of a selection of disjunct reduction factors may give a good lower bound.

7 Software for Workflow Automation

In this section, we describe a Python script that automates the edge refinement of Section 5.1 and data assessment of Section 5.2 for arbitrary regulatory networks and time series

TABLE 2
Reduction of v-Behaviors, Depending on Effective and Ineffective Regulators in R and Values in S

Effective	Ineffective	S	$\lvert\mathcal{R}\rvert$	$\mathcal{K}_v$	Reduction	$f(v,s,r)$
3	0	0	1	0	0.00	0.5
2	0-1	0	1-2	2	0.22	0.50-0.75
1	0-2	0	1-2	7	0.78	0.50-0.94
0	0-3	0	1-8	9	1.00	0.50-1.00

The interactions targeting v are labeled with $+ \wedge =$.

TABLE 3
Reduction of v-Behaviors, Depending on Sum of Effective and Ineffective Regulators in R

Effective + Ineffective	$\mathcal{K}_v$	$\lvert\mathcal{R}\rvert$	Reduction	$f(v,s,r)$
3	128	1	0.50	0.50
2	192	2	0.75	0.75
1	240	4	0.94	0.94
0	255	8	0.99	0.99

The interactions targeting v are labeled with $+ \vee -$.

that are within the computational limitations mentioned in Section 6.1. It is called *erda.py* (Edge Refinement and Data Assessment) and available from sourceforge.[1] The input to the script is a regulatory network (V, E, s, t), a time series A and a monotonicity matrix B. The script then performs up to seven tasks, depending on the outcome of the previous steps:

1. Computation of the parameter space $\mathcal{K}(V, E, t, s)$
   Output: Size of parameter space
2. Selection of $\phi_{A,B}$ compatible parameter sets using the constraint satisfaction preprocessing (see Section 6.2).
   Output: Preprocessing reduction factor, size of compatible parameter space
3. Computation of the number of component behaviors (see Definition 5.2)
   Output: Number of behaviors of each component
4. Computation of the value ranges (see Definition 5.1)
   Output: Value ranges for each context of each component
5. Refinement of edge labels with respect to the hierarchy in Fig. 5
   Output: The new label of each refined edge
6. Check each position (i, j) in A for a necessary oscillation (see Definition 5.5)
   Output: The component and interval for which oscillations are predicted
7. An existence test for best fits (see Definition 5.4)
   Output: Whether or not best fits exist

The script is a single file of about 1,100 lines of code without dependencies on packages other than the standard library. To perform model checking an installation of NuSMV is required. We will now describe the algorithms in more detail.

The computation of $\mathcal{K}(V, E, t, s)$ is performed by computing for each component v the set of behaviors $\mathcal{K}_v$ that agree with the edge propositions of its regulators. By definition, the parameter space consists of independent components (Definition 5.3), so $\mathcal{K}$ is just the Cartesian product of the component behaviors. The behaviors $\mathcal{K}_v$ are computed by a backtracking algorithm with failure on violation of a relevant edge proposition.

The next step, the computation of $\mathcal{K}(V, E, t, s, \phi_{A,B})$, is performed in two steps. First, the constraint satisfaction, discussed in Section 6.2, is checked for each behavior $K_v \in \mathcal{K}_v$ of each component. Behaviors not satisfying the constraints are discarded, and the resulting total reduction factor is returned. Then, $\phi_{A,B}$ compatibility is tested either by model checking or a local state graph search. As shown by Meiers [14], for the network sizes considered here and everywhere precise time series, a best-first graph search with Hamming distance as its cost heuristic can be faster than model checking. If A is somewhere imprecise though, we must resort to NuSMV instead.

If there is at least one $\phi_{A,B}$ compatible parameter set, edges which are not already labeled with a label at the top of the hierarchy are refined until they reach the top or there are no more parameter sets for refinement. Algorithmically this is achieved for each component v by a loop over all compatible behaviors of v, which breaks if none of the interactions targeting v can be further refined. The refinement of a particular interaction is achieved by a (static) mapping from the set of observed atomic propositions (see Definition 2.8) to the resulting strictest admissible proposition. To illustrate this, assume that there are only two behaviors of a component v that is regulated by a component u in $\mathcal{K}(V, E, t, s, \phi_{A,B})$. For the first parameter set the interaction $(u.v)$ is labeled with $+ \wedge -$ and for the second with $+ \wedge =$. The mapping of this set of atomic observations returns $+$. Then, the possible behaviors of each component are counted and displayed. In the same loop the value ranges of each context are gathered and also displayed.

Next, the time series is assessed by checking the existence of a $\phi_{A,B'}$ compatible parameter set for each B', where B' is a monotonicity matrix with exactly one more monotonicity constraint than B, as suggested in Definition 5.5. If no such parameter set exists the respective position (i, j) has selectivity 1, which implies an unobserved oscillation between measurements i and $i + 1$ of component j.

Finally, if no unobserved oscillations were found, the existence of a best fit, see Definition 5.4, is checked and the results are given.

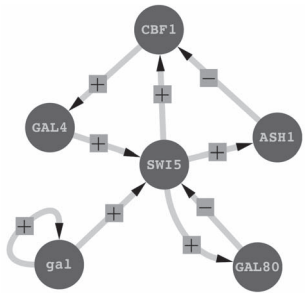


Fig. 8. The IRMA regulatory network.

8 Application: The IRMA Network

We apply the workflow of the previous section to a biological network called IRMA, for which several time series are available. A corresponding search for consistent parameters of a piecewise linear ODE model is described by Batt et al. [2]. All analysis steps can be reproduced with the software from Section 7.

The IRMA regulatory network consists of five genes with gene control and protein-protein interactions, which has been inserted into the genome of *Saccharomyces cerevisiae* (see Cantone et al. [4]) and the environment variable *gal*, see Fig. 8. Several populations of this genetically modified yeast were grown and subjected to perturbations by adding or removing galactose from the growth medium. Altogether 11 real-valued time series are available: five repetitions of the switch-on perturbation (adding galactose) and four repetitions of the switch-off perturbation (removing galactose) plus two averaged time series for each category.

A comprehensive analysis would include all available time series. Since we aim for a clear illustration of our approach, we restrict analysis to the averaged switch-off time series. In addition, we only consider a boolean model.

We binarized the expression data for the galactose removal experiment using the scan-statistic method described in [16]. Additionally, we added values for *gal* based

1. http://sourceforge.net/projects/erda/.

on qualitative observations. The first entry of its profile is left uncertain, because although the cells were washed, we are not sure if galactose was still present in the cytoplasm or not. This resulted in the discrete time series

$$A = \begin{pmatrix} CBF1 & ASH1 & GAL4 & GAL80 & SWI5 & gal \\ 1 & 1 & 1 & 1 & 1 & \perp \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Although the analysis is not affected by removing duplicate rows in A, we have kept duplicates in order to relate the following results to the actual 19 measurements. Matching the binarized data, we chose a boolean representation, i.e., $p(v) = 1$ for every variable v. The state space is then

$$X = [0,1]^6 \text{ with } |X| = 64, \text{ and}$$
$$|\mathcal{K}(V,E,t)| = 2^2 \cdot 2^2 \cdot 2^4 \cdot 2^2 \cdot 2^2 \cdot 2^8 = 2^{20} = 1{,}048{,}576.$$

The network edges and edge-constraints were adopted from [4] and interpreted as $+$ and $-$, i.e., as observable activations or inhibitions. We then computed all parameter sets that satisfy the edge-constraints and reproduce the time series without any monotonicity assumptions:

$$|\mathcal{K}(V,E,t,s)| = 404 \text{ and } |\mathcal{K}(V,E,t,s,\phi_{A,0})| = 73.$$

We proceeded by characterizing the parameter pool $\mathcal{K}(V,E,t,s,\phi_{A,0})$. All parameters of components with a unique regulator, namely $GAL4$, $ASH1$, and $GAL80$, coincide for all parameters sets, i.e., the component behavior is completely determined. The labels of edges targeting these components can be strengthened to $+ \wedge =$, i.e., they are recognized as nonambiguous activating influences. For $SWI5$ one parameter is determined: $K_{SWI5}(\{GAL4\}) = 1$, suggesting that $GAL4$ alone is sufficient to activate $SWI5$, as opposed to galactose which may require $GAL4$ for upregulation of $SWI5$ as the parameter $K_{SWI5}(\{gal\})$ is in the range $[0,1]$.

Regarding the behaviors of $CBF1$ and $SWI5$ as defined in Section 5.1, there are four for the latter and 33 for the former. The set $\mathcal{K}(V,E,t,s,\phi_{A,0})$ is not independent, since $4 \cdot 33 = 132$, but there are only 73 sets in the pool. Therefore, not every behavior of $SWI5$ is compatible with every behavior of $CBF1$. Identification of conflicting behaviors can then be utilized for experimental design. Development of strategies that allow one to identify a component and a corresponding behavior whose parameter determination would result in a maximal decrease of the parameter pool is an issue for future work.

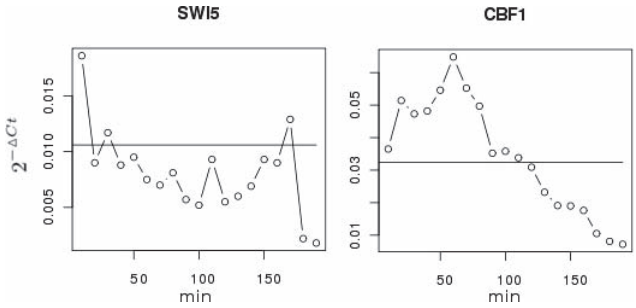


Fig. 9. Real-valued expression profiles of $SWI5$ and $CBF1$. The horizontal lines are the binarization thresholds obtained by the scan-statistic method.

Continuing in the workflow, we assessed the quality of the time series. There are no best fits of the IRMA network to the time series, but computing the selectivity of positions (i,j) in A we found eight positions to have a selectivity of 1 and hypothesize the following oscillations:

Name	Begins oscillation at measurement
$CBF1$	1,9,13
$SWI5$	6,8,9,13,15

The real-valued expression profiles show that $SWI5$ does indeed oscillate, but the oscillations are below the threshold that the binarization method computed (see Fig. 9). In this particular case, the result emphasizes the need of revising the chosen threshold. However, it also illustrates nicely the potential of our method to evaluate sufficiency of measurements, since similar results would be obtained if the data points between 5 and 15 in the $SWI5$ plot were simply missing. Based on our analysis the importance of providing additional measurements for that time span would be highlighted.

For $CBF1$, the expression curve shows a decline with two steady intervals around measurements 10 and 15 (see Fig. 9). Here, the real-valued data shows no oscillation, but rather different plateaus. Our results point out the time points where changes of activity levels result in qualitatively observable effects, and thus indicate the need for a finer representation of activity levels than a simple boolean view. Investigating the relation between the predictions for oscillations generated by our method and the need for an expanded component value range will be an objective of future work.

Since the boolean model for the IRMA network can reproduce the chosen time series, we imposed additional assumptions to illustrate the workflow in case of inconsistencies (Section 5.3). We considered that value changes in $GAL80$ involve transcription processes. Let us assume that the transcription of $GAL80$ is slow, so that it is not expected to significantly change concentration within the sampling rate of 10 minutes, i.e., there will be no oscillations between the sampling points.

The entries of a monotonicity matrix B encoding this assumption are 1 in the column corresponding to the $GAL80$ expression profile. We set all remaining entries of B

to zero, imposing no further monotonicity constraints. The corresponding parameter pool $\mathcal{K}(V, E, t, s, \phi_{A,B})$ is empty. We decided to proceed by revising the structure of the internal components, taking the activating effect of gal on $SWI5$ as given. The IRMA network is structurally compatible with A and B. We now try to derive valid information from the resulting parameter pool. Of the 12,960 parameter sets in the pool where all internal edges of the network, i.e., not (gal, gal) and $(gal, SWI5)$, are relaxed to $+ \vee -$, 144 satisfy $\phi_{A,B}$. Interestingly there are no determined parameter values in this pool, but two interactions are stricter than assumed in every parameter set: $s(ASH1,\ CBF1) = -,\ s(SWI5, CBF1) = +$.

This illustrates how we can recover information from the parameter pool supported by the available data. In summary, we can observe that the reasonable assumption that the switch-off series has captured all oscillations of $GAL80$ validates the original labels targeting $CBF1$.

9 Conclusion

In this paper, we study the compatibility of a model of a regulatory network and its observed behavior in the form of a discretized time series. On the formal level, we slightly extend the usual edge labels (e.g., [3]) with boolean propositions on edges (similar to [8]) and introduce time series that may be partially exact or monotone. On the methodological level, a workflow is suggested that branches in places where given assumptions may or may not be satisfied.

In contrast to related work, we also use our method to assess the quality of the considered time series. In case of consistency of the network structure and the time series, we investigate the temporal resolution of the time series by defining a best fit. For such parameter sets additional measurements would not reveal much further information, because in between measurements all variables approach their target activities without oscillating. However, if no best fits exist, oscillations can be predicted for particular variables in particular time intervals. We have shown the potential of this approach using the IRMA network. In addition, the results hint at the possibility of using the same methods to assess the discretization threshold of individual components, as well as the number of thresholds used for a component. This will be further elucidated in future work.

While we obtain satisfactory results for networks of small and medium size, we certainly have to increase computational efficiency to tackle larger models. A first step in this direction is the discussed constraint satisfaction preprocessing. Future research will focus on developing more powerful implementations of our ideas.

References

[1] J. Barnat, L. Brim, A. Krejci, A. Streck, D. Safranek, M. Vejnar, and T. Vejpustek, "On Parameter Synthesis by Parallel Model Checking," *IEEE/ACM Trans. Computational Biology and Bioinformatics,* vol. 9, no. 3, pp. 693-705, May/June 2012.

[2] G. Batt, M. Page, I. Cantone, G. Goessler, P. Monteiro, and H. de Jong, "Efficient Parameter Search for Qualitative Models of Regulatory Networks Using Symbolic Model Checking," *Bioinformatics,* vol. 26, pp. i603-i610, Sept. 2010.

[3] G. Bernot, J.-P. Comet, A. Richard, and J. Guespin, "Application of Formal Methods to Biological Regulatory Networks: Extending Thomas' Asynchronous Logical Approach with Temporal Logic," *J. Theoretical Biology,* vol. 229, no. 3, pp. 339-347, 2004.

[4] I. Cantone, L. Marucci, F. Iorio, M.A. Ricci, V. Belcastro, M. Bansal, S. Santini, M. Di Bernardo, D. Di Bernardo, and M.P. Cosma, "A Yeast Synthetic Network for in Vivo Assessment of Reverse-Engineering and Modeling Approaches," *Cell,* vol. 137, no. 1, pp. 172-181, 2009.

[5] N. Chabrier-Rivier, M. Chiaverini, V. Danos, F. Fages, and V. Schächter, "Modeling and Querying Biomolecular Interaction Networks," *Theoretical Computer Science,* vol. 325, no. 1, pp. 25-44, 2004.

[6] C. Chaouiya, E. Remy, B. Mossé, and D. Thieffry, "Qualitative Analysis of Regulatory Graphs: A Computational Tool Based on a Discrete Formal Framework," *Proc. First Multidisciplinary Int'l Symp. Positive Systemsd Applications,* L. Benvenuti, A. De Santis, and L. Farina, eds., pp. 830-832, 2003.

[7] A. Cimatti, E.M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "Nusmv 2: An Opensource Tool for Symbolic Model Checking," *Proc. 14th Int'l Conf. Computer Aided Verification (CAV '02),* pp. 359-364, 2002.

[8] F. Corblin, E. Fanchon, and L. Trilling, "Applications of a Formal Approach to Decipher Discrete Genetic Networks," *BMC Bioinformatics,* vol. 11, article 385, July 2010.

[9] E.S. Dimitrova, L.D. Garcia-Puente, F. Hinkelmann, A.S. Jarrah, R.C. Laubenbacher, B. Stigler, M. Stillman, and P. Vera-Licona, "Parameter Estimation for Boolean Models of Biological Networks," *Theoretical Computer Science,* vol. 412, no. 26, pp. 2816-2826, 2011.

[10] E.S. Dimitrova, J.J. McGee, and R.C. Laubenbacher, "Discretization of Time Series Data," *J. Computational Biology,* vol. 17, no. 6, pp. 853-868, 2010.

[11] J. Fromentin, J.-P. Comet, P.L. Gall, and O. Roux, "Analysing Gene Regulatory Networks by Both Constraint Programming and Model-Checking." *Proc. IEEE 29th Ann. Int'l Conf. Eng. in Medicine and Biology Soc. (EMBC '07),* pp. 4595-4598, Aug. 2007.

[12] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning About Systems.* Cambridge Univ. Press, 2004.

[13] H. Klarner, H. Siebert, and A. Bockmayr, "Parameter Inference for Asynchronous Logical Networks Using Discrete Time Series," *Proc. Ninth Int'l Conf. Computational Methods in Systems Biology (CMSB '11),* pp. 121-130, 2011.

[14] S. Meiers, "Graph Traversal Versus Model Checking in Deciding Compatibility of Time Series with Logical Networks," bachelor's thesis, Freie Universität Berlin, 2011.

[15] P.T. Monteiro, D. Ropers, R. Mateescu, A.T. Freitas, and H. de Jong, "Temporal Logic Patterns for Querying Dynamic Models of Cellular Interaction Networks," *Bioinformatics,* vol. 24, no. 16, pp. i227-i233, 2008.

[16] C. Müssel, M. Hopfensitz, and H.A. Kestler, "Boolnet - An R Package for Generation, Reconstruction, and Analysis of Boolean Networks," *Bioinformatics,* vol. 26, no. 10, pp. 1378-1380, 2010.

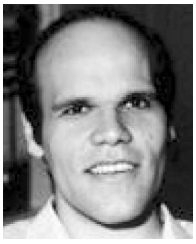
[17] A. Richard, "Negative Circuits and Sustained Oscillations in Asynchronous Automata Networks," *Advances in Applied Math.,* vol. 44, no. 4, pp. 378-392, 2009.

[18] I. Shmulevich and W. Zhang, "Binary Analysis and Optimization-Based Normalization of Gene Expression Data," *Bioinformatics,* vol. 18, no. 4, pp. 555-565, 2002.

[19] R. Thomas, "Regulatory Networks Seen as Asynchronous Automata: A Logical Description," *J. Theoretical Biology,* vol. 153, no. 1, pp. 1-23, 1991.

Hannes Klarner received the master's degree in mathematics from the University College London and began work as a research assistant in the group of Professor Alexander Bockmayr at Freie Universität Berlin in 2009. His research interest are gene regulatory networks and specifically reverse engineering ideas for the formalism of R. Thomas.

Heike Siebert received the undergraduate and doctoral degrees in mathematics from the Christian-Albrechts-Universität Kiel. She received the PhD degree in 2004. The following years, she worked as a research assistant at Freie Universität Berlin, where she currently holds a professorship in applied mathematics. Her interests lie in the field of discrete dynamical systems and application of discrete modeling methods in systems biology.



Alexander Bockmayr received the diploma in mathematics from TU Munich in 1985, the PhD degree from University Karlsruhe in 1990, and the Habilitation from Saarland University in 1996. From 1998 to 2004, he was a professor at University Nancy, France, and head of the MODBIO project at LORIA and INRIA. Since 2004, he has been a professor at DFG Research Center Matheon and FU Berlin. His current research focuses on new approaches for mathematical modeling of metabolic and regulatory networks, with special emphasis on constraint-based methods.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.