Toolkit for Reverse Engineering of Molecular Pathways via Parameter Identification

A Dissertation
Submitted in Partial Fulfilment of the Requirements for the Degree of
Doctor Rerum Naturalium (Dr. rer. nat.)

to the Department of Mathematics and Computer Science
of the Freie Universität Berlin

Adam Streck

Berlin, 2015

Supervisor: Prof. Heike Siebert
Second examiner: Dr. David Šafránek

Date of the defense: 14.3.2016

I would like to dedicate this thesis to Heike Siebert.

Were it not for you, I would not have chosen to become a scientist.

Thank you.

Thanks

I would like to thank to my supervisors, Heike Siebert and Alexander Bockmayr, for their guidance in my post-graduate studies and for all the time they gave me.

For all the help and work in our collaborations, I would like to thank to Kirsten Thobe, Hannes Klarner, Therese Lorenz, David Šafránek, Kaveh Pouran Yousef, and Max von Kleist.

For the advices I received when writing this thesis I would like to thank to Annika Röhl, Hannes Klarner, Tereza Dědová, Stephen Lavelle, Felix Mattes, and Christoffer Anselm.

To my mother, Šárka Strecková, I would like to thank for her unyielding support in the years leading to the moment of writing this paragraph—my whole life.

My gratitude also belongs to all those who made my life in Berlin such a pleasant experience.

Last, but the warmest thank you is to my life companion, Tereza Dědová, who followed me to Berlin across country borders, and whom all love, with my whole heart.

Summary

This thesis is a contribution to the field of systems biology, which is concerned with mathematical and computational modelling of biological systems. The aim of the field is to understand biological processes via holistic computational methods.

One of the standing problems in systems biology is how to derive model of a system, preferably one easily understandable by humans, from experimental data and observations. Understandably, the structure of the problem depends heavily on the system of interest and the available data, therefore it is worthwhile to create new methods that utilize particular features, as there can hardly be a universal solution.

Here we present an approach for modelling and analysis of complex biological networks that uses a high-level, abstract modelling framework—the multi-valued logical networks. In this framework we employ an automated method originating in the theoretical computer science, called model checking, which allows for formal reasoning about dynamical systems. We can then create a multitude of candidate models and use model checking method to compare the behaviour of these to experimental data.

Our approach however produces high volumes of data. To be able to work with the data we use basic statistical methods, which allow us to summarize the dataset into a few key values. In addition, these values can be subsequently compared between multiple datasets. For better understanding we couple these methods with an interactive visualization software.

The whole framework is implemented in a tool called TREMPPI, which is available under an open-source license and distributed together with this thesis.

We illustrate the functions of TREMPPI on three biological studies—two human signalling pathways, related to cancer, and a protection mechanism of the bacteria E. Coli.

Zusammenfassung

Diese Dissertation ist ein Beitrag zur Systembiologie, welche sich mit der mathematischen Modellierung biologischer Prozesse beschäftigt. Das Ziel dieses Forschungsbereiches ist es biologische Vorgänge mit Hilfe von ganzheitlichen, computergestützten Methoden zu verstehen.

Eine aktuelle Herausforderung der Systembiologie ist die automatische Herleitung von Modellen anhand von Beobachtungsdaten und Experimenten, bestenfalls eines Modells, dessen Aufbau dem Nutzer leicht zugänglich ist. Natürlich hängen die Schwierigkeiten dieses Problems von dem zu modellierenden System ab und da kaum zu erwarten ist, dass es eine universelle Lösung geben kann, lohnt es sich spezifische Methoden für spezielle Systeme zu entwickeln.

Wir stellen hier einen Ansatz zur Modellierung and Analyse von komplexen biologischen Netzwerken vor, der auf einer hohen Abstraktion der Vorgänge basiert—den sogenannten Mehrwertigen logischen Netzwerken. In diesem Zusammenhang verwenden wir eine Methode aus der theoretischen Informatik, nämlich die automatische Modellprüfung, welche uns erlaubt formal begründete Aussagen zur Dynamik der modellierten Systeme zu treffen. Wir erzeugen eine Menge von Modellkandidaten und verwenden die Modellprüfung um ihre Dynamiken mit den Beobachtungsdaten zu vergleichen.

Unser Ansatz erzeugt große Mengen an Daten. Wir verwenden statistische Methoden um wichtige Eigenschaften eines Datensatzes hervorzuheben. Zusätzlich können aufgrund dieser Eigenschaften Vergleiche zwischen unterschiedlichen Datensätzen aufgestellt werden. Zum einfacheren Verständnis sind diese Methoden teil einer interaktiven Visualisierungssoftware.

Der gesamte Ansatz ist als Software implementiert, welche TREMPPI heißt und frei, unter einer Open-Source Lizenz, erhältlich ist.

Wir veranschaulichen die Funktionen von TREMPPI an drei biologischen Fallstudien—zwei Signalwege in menschlichen Zellen, welche im Zusammenhang von Krebs untersucht werden, und ein Modell zum Schutzmechanismus von E. Coli.

Contents

CHAPTER 1

Introduction

> ”Most of an organism, most of the time, is developing from one pattern into another, rather than from homogeneity into a pattern. One would like to be able to follow this more general process mathematically also. The difficulties are, however, such that one cannot hope to have any very embracing theory of such processes, beyond the statement of the equations. It might be possible, however, to treat a few particular cases in detail with the aid of a digital computer.”
>
> *Alan Turing, The Chemical Basis of Morphogenesis [Tur52]*

Systems biology [WVD12] is a research field, and a philosophical school, that believes that to understand biological systems, it is necessary not only to know what it is composed of, but also how its components interact and to what end [Nob08]. This presents a herculean problem, as it is generally accepted that living organism are among the most complex systems [RA09], if with humans being possibly the most complex systems in existence, as also illustrated in Figure 1.1. To understand the function of such a system, one usually needs to understand the function of its components, then the function of their components, and so on. This means that in the attempt to understand the function of an organism, we descend through its organs, tissue, cells, proteins, atoms, and possibly even sub-atomic particles, finding that every single component somehow plays a role in the result [AJV09]. However, despite this complexity and all the chaos and uncertainty, the biological systems usually exhibit quite a robust and deterministic behaviour, leading to a belief that in many cases it may be sufficient to focus only on one layer of the system at a time, abstracting from the complexity of the underlying layers [Wol01]. Following this line of thought, many modelling formalisms [dJ02] have been developed in an attempt to foster our understanding of life itself. Each of these has its strengths, weaknesses and is suited to modelling only a particular subset of functions of a biological

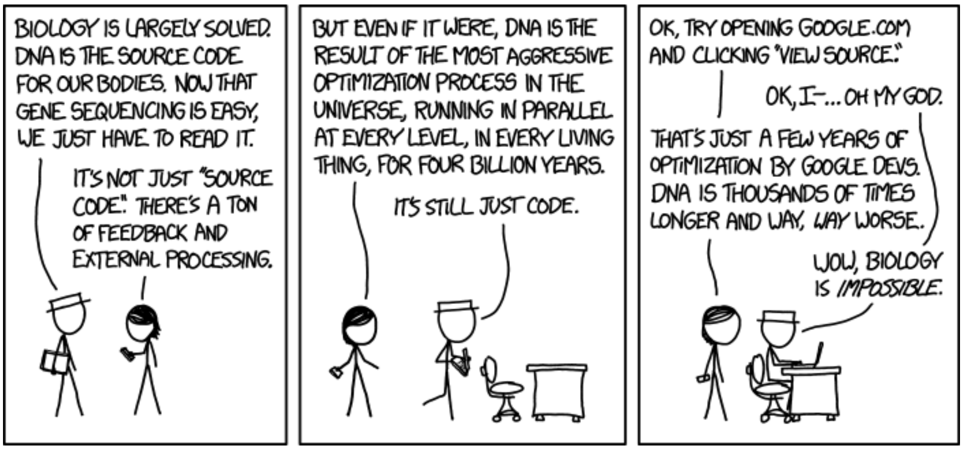

BIOLOGY IS LARGELY SOLVED. DNA IS THE SOURCE CODE FOR OUR BODIES. NOW THAT GENE SEQUENCING IS EASY, WE JUST HAVE TO READ IT.
IT'S NOT JUST "SOURCE CODE." THERE'S A TON OF FEEDBACK AND EXTERNAL PROCESSING.
BUT EVEN IF IT WERE, DNA IS THE RESULT OF THE MOST AGGRESSIVE OPTIMIZATION PROCESS IN THE UNIVERSE, RUNNING IN PARALLEL AT EVERY LEVEL, IN EVERY LIVING THING, FOR FOUR BILLION YEARS.
IT'S STILL JUST CODE.
OK, TRY OPENING GOOGLE.COM AND CLICKING "VIEW SOURCE."
OK, I—... OH MY GOD.
THAT'S JUST A FEW YEARS OF OPTIMIZATION BY GOOGLE DEVS. DNA IS THOUSANDS OF TIMES LONGER AND WAY, *WAY* WORSE.
WOW, BIOLOGY IS *IMPOSSIBLE*.

Figure 1.1: Xkcd comic, Randal Munroe, 2015, `http://xkcd.com/1605/`.

system. None of these frameworks promises to capture a full-scale, realistic snapshot of a living organism, however it may suffice the needs of the modeller, who is usually focusing only on ”making sense” of a particular part [Ing13].

Our primary interest in this work is self-regulation and decision making of a single cell, either as an individual organism, or as a part of a multicellular being. These functions are mediated mainly by two mechanisms: the fast molecular signalling [AJL+08], where molecules interact by physical contact with subsequent chemical reactions; and the slow gene regulation, where molecules react with the DNA transcription machinery to affect the construction of proteins [AJL+08]. Based on these mechanisms, a cell makes decisions, such as whether to proliferate, differentiate, die, etc. In many aspects, both these mechanisms constitute a computational circuitry of a cell, with the input-output behaviour being similar in design to the computational structure of digital circuits [Alo06].

One of the earliest frameworks that focused on modelling cells in terms of logical circuits were the Boolean networks of Stuart Kauffman [Kau69]. In this framework each modelled gene is assigned to one Boolean component, which can then be either logically *true* or logically *false*. These logical values can then be interpreted as being expressed or silenced in the system, active or inactive, etc. Each component is then assigned a rule determining what its future state is going to be, based on the current state of other components that influence it. These rules describe the effect the components have on each other—their mutual *regulations*. These regulations can stem from many mechanisms like phosphorylation, methylation, promotor binding, gene silencing, ubiquitylation, DNA cutting, etc. [AJL+08]. Using these rules, a simulation of a network is possible, where all the components are at the same time updated to their future state in each step of the simulation.

This is an extremely high-level and abstract approach, nonetheless numerous studies were conducted using this framework to this date [WSA12], reporting novel findings. Still it is not hard to imagine that for many applications, this framework is too simple. René Thomas therefore proposed a more realistic framework [Tho91], often referred to as multi-valued networks or Thomas networks. Understandably, in the multi-valued networks a component can have more than two values for the representation of distinct qualitative features. One particular example being the DNA damage response network where the suppressor LexA can prevent expression of none, one, or two protein complexes [Mic05]. Secondly, the expectation that all components change their configuration at the same time is not only unrealistic, but has also strong implications on what behaviour a network can exhibit [Tho13].

As in all mathematical modelling, the choice of framework and approach depends also on what is the interest of the modeller—to create a visual representation of the system, to obtain an executable model allowing for computational simulations, something else entirely? Our goal was to design a stand-alone toolkit that would support creation of models, based on experimental data, with a two-fold purpose. The first goal is for the model to be descriptive and to elucidate the biological mechanism being modelled. The second goal is for the model to allow for evaluation of intervention strategies, *e.g.* chemotherapy. Such models are crucial in understanding diseases, as is for example cancer, where the standard function of a part of a cell is suspended due to factors like mutation, viral hijack, etc. In such cases one first needs to understand what is a difference between the damaged and a healthy cell, usually with only sparse sets of data available. Knowing how a system behaves makes it then easy to design a treatment and test its effect on the said model. The case of personal healthcare is however not the limiting one. Many a biological system are still not well understood and being able to extract executable models from data is persistently one of the top tasks of systems biology [PSRA$^+$11]. We have therefore chosen Thomas networks as our framework, since from our perspective they are complex enough to capture relevant information, however still easy enough to grasp and also amenable to automated analysis.

The automation aspect is quite an important one, since the more precise modelling schemes, like differential or stochastic equations are quite hard to simulate [KS08], let alone if some parameters of a model are uncertain (*e.g.* in the case of mutation). Thomas networks, on the other hand, allow to enumerate and evaluate all possible disruptions to the model, as the space of different mutations and the space of possible states of a network are both finite [Tho91]. In particular, there is an established approach for automated analysis of dynamical systems, called model checking (MC) [CGP99], which has become also a standard tool for analysing Thomas networks since its first use [BCRG04].

The principle of our model checking approach is to overlap all the possible simulations of a model (from different initial conditions, with different choices being made) and analyse the whole resulting structure. Having this structure means that we know, in full, the behaviour of a model. It is therefore easy to evaluate whether it is in line with the experimental observations, or to see what effects would an intervention have (as could be for example caused by administering a drug). In this work we use MC as our main approach for analysis of models and develop methods for improving its effectiveness and efficiency.

In addition, when dealing with an uncertainty or scarcity of data for a particular model, which is often the case in the modelling process, we can use the model checking procedure to enumerate all possible models that fit the known boundaries and for each of them to state whether it measures the constraints imposed by the model checker. Such a process is usually referred to as *parameter identification* [CBBC11]. In this thesis we focus on the parameter identification and describe novel extensions to this method. We are also aiming to tackle two problems that arise from using MC for parameter identification. Firstly, when evaluating one possible model of a system, one is usually focusing again only on some abstract, qualitative feature, *e.g.* whether the system stabilizes or not. While this is useful information, its binary nature means that the set of all possible models is simply split in two, one part having the feature, the other not. This poses a problem if one is aiming to pick an optimal model. Even if multiple features are used, there is likely still a big set of models that has all the features and whose members are further indistinguishable. Secondly, considering all the options leads to a rapid combinatorial explosion, and while quite huge sets can still be easily manipulated and stored by the computer, it becomes swiftly infeasible for the researcher to keep a mental insight into the structure of the set of models. Similarly to recent works of other authors in the field [ASRC$^+$10, GVE$^+$13] we propose to focus not only on the individual models, but also on sets of models based on some selection criteria. Such a set can then be further evaluated by data mining and statistical procedures in order to gain further insights about the system.

In summary, we introduce a unified workflow for parameter identification, illustrated in Figure 1.2, which combines formal and statistical methods with the aim of maximizing the amount of knowledge obtained from data. This workflow is supported by a software called TREMPPI—Toolkit for Reverse Engineering of Molecular Pathways via Parameter Identification. This comprehensive toolkit implements all the methods described in this thesis and was also used for all biological studies presented here. It provides a visual interface for an easy use, data and project management, and a set of highly optimized computational tools. TREMPPI is distributed as an attachment to this thesis or is available on-line under an open-source licence, for further details see Appendix B.1.

The thesis is structured as follows. In the next chapter we provide the

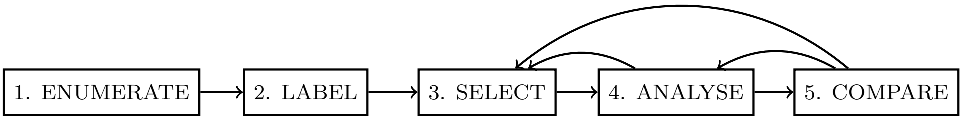


Figure 1.2: Our workflow starts with the enumeration (1) of all possible models that fit the expectations of the modeller about the structure. Each of the models is evaluated for certain properties (2) like dynamical behaviour, and the result of the evaluation is stored with the model as its *label*. After the models are labelled, the user can select (3) a subset of these that seem of special interest. This selection can then be analysed using various tools (4) and compared to other selections (5). The selection or analysis can then be refined based on the newly gained knowledge.

definitions necessary for understanding the methods presented in this thesis, but are not part of the contributions of this thesis. In Chapter 3 we present all our methods in a way that is sufficient for their understanding. These are ordered such that the workflow given in Figure 1.2 is preserved. The following chapter then applies the methods to three different biological networks, each of the studies focusing primarily on one part of the workflow. In Chapter 5 we present our methods again, however this time in a formal fashion, together with the respective proofs and algorithms. Additionally, computational aspects of our implementation are discussed. Finally, a comparison to other existing approaches is given in Chapter 6, together with an overall conclusion.

CHAPTER 2

Background

In this section we provide a formal description of the framework of Thomas networks and of the model checking method. The definitions provided here are compatible with the definitions as given by other authors, see for example [RCB06] and [CGP99].

We conclude this section with a description of methods that were co-developed by the author before starting the research described in this thesis. These methods have been described in [BBK+12, KSv+12].

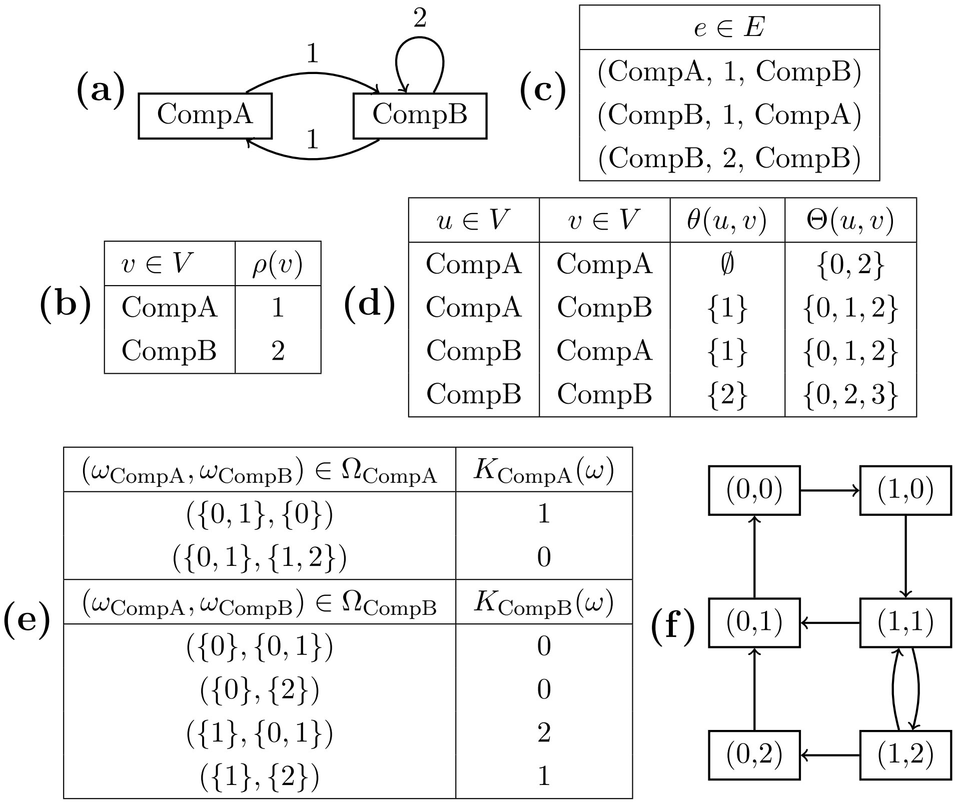
Most of the mathematical notation used throughout the article is quite straightforward. There are however a few more involved elements we review here:

- *substitution*: we use the notation $Vec_{i \leftarrow n}$ to state that in the vector Vec the element with the index i is replaced by the value n. Moreover, unless stated otherwise, the elements are expected to be indexed by natural numbers starting from 1. For example $(a, b, c)_{2 \leftarrow e} = (a, e, c)$.

- *interval*: we use the standard notation for intervals with parenthesis denoting the endpoint exclusion and square brackets the endpoint inclusion. However we extend the notation also to the intervals on integers, for example $[-1, 2) = \{-1, 0, 1\}$. The domain of the interval is always apparent from the context.

- *indexing*: some of the sets we use in the thesis can be naturally ordered in the lexicographical ordering. For convenience, we sometimes use elements of such sets as indices for a vector of the same length. For example have a set $Set = \{p, q, r\}$ and a vector $Vector = (U, V, W)$, then $Vector_q = V$.

- *sets*: we use the symbols $\mathbb{N}_0$ and $\mathbb{N}_1$ to denote the set of natural numbers with and without zero, respectively. For a power set of a set S, we use the notation $\mathbb{P}(S)$. For the size of a set S we use the symbol $[\![S]\!]$.



(b)

$v \in V$	$\rho(v)$
CompA	1
CompB	2

(c)

$e \in E$
(CompA, 1, CompB)
(CompB, 1, CompA)
(CompB, 2, CompB)

(d)

$u \in V$	$v \in V$	$\theta(u,v)$	$\Theta(u,v)$
CompA	CompA	$\emptyset$	$\{0,2\}$
CompA	CompB	$\{1\}$	$\{0,1,2\}$
CompB	CompA	$\{1\}$	$\{0,1,2\}$
CompB	CompB	$\{2\}$	$\{0,2,3\}$

(e)

$(\omega_{\text{CompA}}, \omega_{\text{CompB}}) \in \Omega_{\text{CompA}}$	$K_{\text{CompA}}(\omega)$
$(\{0,1\},\{0\})$	1
$(\{0,1\},\{1,2\})$	0
$(\omega_{\text{CompA}}, \omega_{\text{CompB}}) \in \Omega_{\text{CompB}}$	$K_{\text{CompB}}(\omega)$
$(\{0\},\{0,1\})$	0
$(\{0\},\{2\})$	0
$(\{1\},\{0,1\})$	2
$(\{1\},\{2\})$	1

Figure 2.1: The running example. **(a)** The regulatory network, **(b)** its components, and **(c)** regulations. **(d)** The threshold functions. **(e)** One of the 324 possible parametrizations of the network. **(f)** The asynchronous dynamics encoded by the parametrization.

- *transition composition*: we use the symbol $\to$ to denote the set of transitions (edges) in some oriented graph. For S the states of the graph, this is a relation on S, *i.e.* $\to \subseteq S \times S$. To refer to a transition we then usually use the infix notation, *i.e.* we write $s \to s'$ instead of $(s, s') \in \to$. Lastly, we write $(\to)^k$ for $k > 1$ to describe a path in the system of length $k+1$, *i.e.* $(\to)^k = \{(s^1, \dots, s^{k+1}) \mid \forall i \in [1,k] : s^i \to s^{i+1}\}$. As a special case we put $(\to)^0 = S$.

2.1 Logical Modelling

In this section we define the notions related to the modelling framework. Most of the terms are illustrated in Figure 2.1 on a toy example.

2.1.1 Regulatory Network

The topology of a biological system is encoded as a directed graph, called *regulatory network* (RN) $G = (V, E, \rho)$ where:

- V is a set of named *components*,
- $E \subseteq V \times \mathbb{N}_1 \times V$ is a set of *regulations* s.t. for each $(u, t, v) \in E$ it holds that $t \leq \rho(u)$,
- $\rho : V \to \mathbb{N}_1$ is the *maximum activity* label s.t. each component can adopt an integer from $[0, \rho(v)]$, denoting its current *activity level.*

An example of a simple regulatory network is in Figure 2.1a. We denote $\mathcal{G}$ the set of all possible regulatory networks.

In modelling terms, each component is usually used to describe one functional component of the system, *e.g.* protein, signalling molecule, receptor, etc. The set of activity levels then describes qualitatively different configurations for this component. A regulatory network whose components are all two-valued is called *Boolean network* (BN).

A regulation then states that the source of the regulation has an effect on its target. The effect could be for example that the source breaks down the target. This effect is however dependent on the activity level of the source. For a regulation $(u, t, v) \in E$ the value t denotes a *threshold.* This threshold is the lowest *activity level* of u at which the regulation starts to affect v. The effect persists until the closest higher threshold is reached.

For syntactic purposes we introduce a threshold function $\theta : V \times V \to \mathbb{P}(\mathbb{N})$ s.t. $\theta(u, v) = \{t \mid (u, t, v) \in E\}$ and its extended version Θ s.t. $\Theta(u, v) = \theta(u, v) \cup \{0, \rho(u) + 1\}$ for any pair $u, v \in V$. The extended threshold function of our example is given in Figure 2.1d. Moreover, if $(u, t, v) \in E$ then $t_-, t_+ \in \Theta(u, v)$ denote the closest element below and above t, *i.e.* have $\succ^\Theta$ the ordinal successor function in Θ, then $\succ^\Theta (t_-) = t$ and $\succ^\Theta (t) = t_+$.

2.1.2 Regulatory Contexts

If there are multiple regulations of one component by another, a regulation ceases its function once a higher threshold is reached. For example if $(u, t, v), (u, t_+, v) \in E$, then the regulation (u, t, v) will be active only in the interval $[t, t_+)$ of the activity levels of u. Consequently, the thresholds divide the range of activity levels of a component into so-called *activity intervals*

$$I_v^u = \{[t, \succ^\Theta (t)) \mid t \in \theta(u, v) \cup \{0\}\}. \tag{2.1}$$

Note that if $\theta(u, v) = \emptyset$ then $I_v^u = [0, \rho(u) + 1)$.

For each component we can then create a set of configurations of components of the system, called *regulatory contexts*, where the behaviour of a component $v \in V$ can qualitatively differ from the other contexts, denoted and defined $\Omega_v = \prod_{u \in V} I_v^u$. All the regulatory contexts for the both components of the toy example are given in the left column of the table in Figure 2.1e.

2.1.3 Parametrization

Based on the regulatory contexts of a component, the qualitative behaviour of the component is then fully described through a *partial parametrization* $K_v : \Omega_v \to [0, \rho(v)]$.

The value of the function is called a *logical parameter* or a *target value.* It describes, based on the configuration of regulators, what is the change in the activity of their target. The logical parameters are useful once there is more than one incoming regulation, as competing effects may take place and it is necessary to resolve what is the resulting behaviour. For example, if there is an activator and an inhibitor of a certain gene present at the same time, there are two possible partial parametrizations, depending on whether the inhibitor is stronger than the activator or not.

To describe the behaviour of a regulatory graph we therefore need to assign a partial parametrization to each component, creating a full *parametrization* $K = (K_v)_{v \in V}$. One possible parametrization of the toy example is given in Figure 2.1e.

Note that for each $v \in V$ the domain of the partial parametrization K_v (the set Ω_v) is sufficient to obtain the set of regulators of v. To do so we only need to list the values of the regulatory intervals in all the possible $\omega \in \Omega_v$, since for each $u \in V$ these form exactly the set I_v^u. From I_v^u we can in turn obtain the whole G. A parametrization therefore fully suffices to derive both the behaviour and the topology of a network and we use the notation for a representant of a class, $G = [K]$, to describe the regulatory graph G obtained from parametrization K. For illustration consider the example in Figure 2.1e. From $\Omega = (\Omega_{\mathrm{CompA}}, \Omega_{\mathrm{CompB}})$ we see that there are components CompA, CompB. Then from Ω_{CompA} we see that CompA has levels $0, 1$ and no self-regulation, while CompB has levels $0, 1, 2$ and there is a regulation with the threshold 1. Similarly for Ω_{CompB}.

In particular we use the upper index K to describe a feature relating to the parametrization K only and $[K]$ or G to describe a feature relating to the graph G in total. *E.g.* $V^G = V^{[K]}$ would be a set of components of a network G. Also we further use K to describe a single model and $\mathcal{K}^G$ to denote the set of all possible parametrizations of a regulatory graph G. If G can be arbitrary (but fixed) we use simply $\mathcal{K}$.

2.1.4 Transition System

Having a parametrization we can describe, in total, the dynamic behaviour of the network. This description is provided through a directed graph, called *transition system* (TS), where each node describes one configuration of activity levels of the components—a *state*. The set of all states is then called the *state space*. Because the *state space* is not dependent on the parameters, but only on the components of the RN, we denote and define it as:

$$S^G = \prod_{v \in V^G} [0, \rho(v)].$$

The edges (transitions) are then placed between pairs of states based on the parametrization and also based on the *update scheme* we choose [Ger02]. The update scheme describes how many components can change their value at a time, by how much, and in which order. We focus solely on the *unitary asynchronous* update, meaning that only one component can change by the value of one, but there are multiple transitions possible from one state. This scheme is considered to be the most biologically realistic [TA90]. A transition system for the toy example is given in Figure 2.1f. In the following we explain how the transitions were derived from the example parametrization.

First, K is converted into a so-called *update function* $F^K = (F^K_v)_{v \in V}$ where $F^K_v : S^{[K]} \to [0, \rho(v)]$ for all $v \in V$. Here we exploit the fact that for each $s \in S^{[K]}$ and for each $v \in V$ there exists a context $\omega \in \Omega^{[K]}_v$ such that $s \in \prod_{u \in V} \omega_u$. For brevity we will further write $s \in \omega$ instead of $s \in \prod_{u \in V} \omega_u$. For every $v \in V^{[K]}$ we obtain the function $F^K_v : S^{[K]} \to S^{[K]}$ from a parametrization K_v as

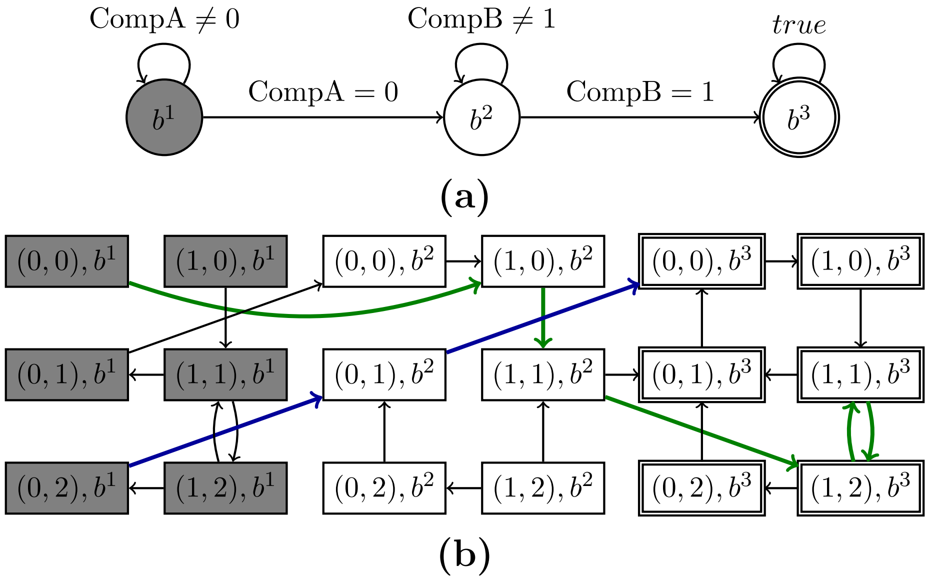
$$F^K_v(s) = \begin{cases} s_v + 1, & \text{if } s_v < K_v(\omega), s \in \omega, \\ s_v, & \text{if } s_v = K_v(\omega), s \in \omega, \\ s_v - 1, & \text{if } s_v > K_v(\omega), s \in \omega. \end{cases} \tag{2.2}$$

Note that we have only three options how the value of a component can change, namely either increase by one, remain constant, or decrease by one. This provides us with a certain notion of direction of a derivative (positive, zero, or negative).

Having F^K, we now assign each model a TS $T^K = (S^{[K]}, \to^T)$ such that $s \to^T s'$ for some $s, s' \in S^{[K]}$ s.t. the following holds:

$$\begin{aligned} &\text{if } s = s' \text{ then } \forall v \in V : F^K_v(s) = s_v, \\ &\text{if } s \neq s' \text{ then } \exists v \in V : F^K_v(s) = n \wedge s' = s_{v \leftarrow n}. \end{aligned} \tag{2.3}$$

where $s_{v \leftarrow n}$ is a state s where the value indexed by v was replaced by the value n.



(a)

(b)

Figure 2.2: **(a)** A BA B^G for the property P^G: "Is there a path where first CompA $= 0$ and then CompB $= 1$?", such that $S^B = \{b^1, b^2, b^3\}, I^B = \{b^1\}, A^B = \{b^2\}$. The initial node is on a gray background, the accepting node has double border. **(b)** The synchronous product of the TS T^K in Figure 2.1e and the BA B^G. In green is one possible run that is a witness of satisfaction of P^G in $X^K = T^K \times B^G$. In blue is a witness for the case the property is encoded via a TBA.

Note that this definition poses two requirements on the structure of the transitions. Either there is a loop exactly on the state where all the parameter values are equal to the current state, or there is a change by exactly one in exactly one component. These are the requirements of an asynchronous transitions system and we denote $\mathcal{T}^{[K]}$ the set of asynchronous transitions systems over the state space $S^{[K]}$. This notion is particularly important in Section 5.2.

2.2 Model Checking

In this section we will explain the details of the model checking (MC) method we use for analysis of dynamical behaviour of models. In general, the model checking method enables the user to form queries about a dynamical system and obtain a yes/no answer to the query. More specifically, the user can ask a question of the form "does this property hold in the system". We use the term *property* for such a query and use the *Büchi Automata* [BK08] (BA) based model checking to decide whether a property holds in a model.

The Büchi Automata based model checking is in general used for model checking of properties described using the *Linear Temporal Logic* (LTL). We conduct our own encoding into BA, which is detailed in Section 5.4 and consequently do not use LTL at all. Therefore we will not provide any formal description of LTL here. Additionally, the BA are actually strictly more expressive than LTL [BK08], providing a possible advantage over the standard LTL model checking.

2.2.1 Büchi Automata

At its core a Büchi automaton is a finite state automaton [Sip96], adapted to words of infinite length (knowledge of automata theory is not necessary for the purposes of this thesis). Formally we describe a BA associated with G as a four-tuple $B^G = (S^B, \xrightarrow{\mathcal{L}(G)}, I^B, A^B)$, where:

- S^B is a set of states,
- $\xrightarrow{\mathcal{L}(G)}$ is a transition relation with propositions where $\mathcal{L}(K) = \mathbb{P}(\{v{*}n \mid v \in V^G, * \in \{\leq, \geq, <, >, =\}, n \in [0, \rho(v)]\})$,
- $I^B \subseteq S^B, A^B \subseteq S^B$ are sets of initial and accepting states, respectively.

The purpose of the automaton is to work as a memory, based on the prepositions that are evaluated as *true*. In each state, the automaton can step to a successor through a transition whose whole label evaluates to *true*. As will be shown later, the automaton is combined with a TS to keep track of which prepositions have been satisfied. Consider the automaton in Figure 2.2a; this automaton steps from b^1 to b^2 whenever CompA $= 0$, as a note that this condition has been satisfied.

Note that the labelling of transitions is defined based on the names and activity levels of components. It is therefore necessary to know the respective network when creating a property.

An automaton is then used to show that a property has been satisfied. A BA accepts if and only if there is a path from an initial state that passes infinitely many times through an accepting state. As we do not use BAs individually, but in combination with some TS, we formalize the acceptance notion in the following section.

2.2.2 Synchronous Product

To assure that a model satisfies a property, its respective TS and the encoding automaton for that property are combined into a *synchronous product* (SP). Then the SP is traversed to establish whether a path that would confirm satisfaction of the property exists or not.

To construct the transition relation of an SP, the propositions on the labels of a BA are then evaluated on the states of a TS. In particular, a state can be interpreted in the terms of propositions valid in the state, *e.g.* for a state $s \in S^{[K]}, s_{CompA} = 0, s_{CompB} = 1$ we can make statements such as:

$$\begin{aligned} &s \models \text{CompA} = 0 \wedge \text{CompB} = 1,\\ &s \models \text{CompA} < 1 \wedge \text{CompB} > 0 \wedge \text{CompB} < 2,\\ &s \not\models \text{CompA} > 0 \end{aligned}$$

where $\models$ denotes the standard logical validity [HR04]. The validity of the labels of BA transitions is then evaluated on the states of TS, so to find the matching BA transition for each TS state. This is crucial to the construction of the SP, which is constructed as:

$$X^K = T^K \times B^{[K]} = (S^{[K]}, \rightarrow^T) \times (S^B, \xrightarrow{\mathcal{L}([K])}, I^B, A^B) = (S^X, \rightarrow^X, I^X, A^X)$$

where

- $S^X = S^{[K]} \times S^B$,
- $(s, b) \rightarrow^X (s', b') \iff (s \rightarrow^T s') \wedge (b \xrightarrow{\phi} b') \wedge (s \models \phi)$,
- $I^X = S^{[K]} \times I^B$,
- $A^X = S^{[K]} \times A^B$.

An example of a SP is given in Figure 2.2a.

Then, a property $P^{[K]}$ encoded by the automaton $B^{[K]}$ is satisfied in T^K if and only if there is an accepting infinite path (usually called *run*) $w^X \in (\rightarrow^X)^{k-1}$. A run is accepting if and only if it leads from an initial state and then goes infinitely through an accepting state. Formally:

$$\begin{aligned} &X^K, w^X \models P^{[K]} \iff \\ &(w_1^X \in I^X) \wedge (\exists i, l \in [1, k] : i < l \wedge w_i^X = w_k^X \wedge w_l^X \in A^X). \end{aligned} \tag{2.4}$$

A run is said to be infinite, because it forms a so-called *lasso*, which is composed of the path from w_1^X to w_i^X and a cycle on $w_i^X = w_k^X$, therefore we can infinitely many times conduct a transitions. Note that it is required that $i \neq k$.

This run w^X then forms a *witness* of satisfaction of $P^{[G]}$ by K in X^K. We also denote W^X the set of all witnesses in X^K, *i.e.*:

$$W^X = \{w^X \mid X^K, w^X \models P^{[K]}\}.$$

For completeness we add that in the standard model checking the notion of *validity*, rather than *satisfiability*, is investigated. In such a case, the property φ is said to be *valid*, written $X^K \models P^{[K]}$, if there is no witness of its negation, formally:

$$X^K \models P^{[K]} \iff \neg(\exists n \in \mathbb{N}_0, \exists w \in (\rightarrow^X)^n : X^K, w \models \neg P^{[K]}).$$

2.2.3 Automata Hierarchy

Throughout the article we distinguish between three sorts of BA: terminal BA, deterministic BA, and non-deterministic BA.

First we introduce the notions of being *deterministic* and *total*. For a TS T^K and an automaton B^K and a state $b \in B^K$ denote:

$$val(b) = \{\{s \in S^{[K]} \mid b \xrightarrow{\phi} b', s \models \phi\} \mid b' \in S^B\}.$$

A BA B is deterministic and total *iff* for each $b \in S^B$, exactly one of the labels valuates to *true*, *i.e.* $\forall S, R \in val(b) : S \cap R = \emptyset$ (deterministic) and $\bigcup val(b) = S^{[K]}$ (total). The advantage is that the number of transitions for each state in T^K remains the same for each corresponding state in X^K, since we always couple them with exactly one transition from $B^{[K]}$. Such an automaton is then called deterministic BA (DBA).

The terminal BA (TBA) allow for encoding of the so-called *guarantee* properties [BK08], effectively allowing to determine whether a sequence of events is possible or not [eP03] and are strictly less expressive than DBA. The great simplification of these is that every final state has a loop, formally a BA B is a TBA *iff* for each $a \in A^B$ it holds that $a \xrightarrow{true} a$. Since in the TS each state has at least one transition under the condition *true*, the acceptance condition (2.4) for the guarantee property $P^{[K]}$ and the path $w^X \in (\rightarrow^X)^{k-1}$ simplifies to:

$$X^K, w^X \models P^{[K]} \iff w_1^X \in I^X \wedge w_k^X \in A^X.$$

The normal LTL MC is actually done on the third sort—the non-deterministic BA (NBA) [BK08]—which are strictly more expressive than DBA. Later we show that we do not need NBA for our encoding, reducing the complexity of the procedure. However, NBA can still be employed, if necessary, as detailed in Section 5.5.5.

In the following, for a property P^G we use the notation $B^G = \text{TBA}(P^G)$ for a terminal, $B^G = \text{DBA}(P^G)$ for a deterministic, and $B^G = \text{NBA}(P^G)$ for non-deterministic BA encoding. If the automaton type is irrelevant, we write $B^G = {}^*\text{BA}(P^G)$.

2.2.4 Property Metrics

In addition to deciding whether a transition system satisfies a property or not, we can also examine the structure of the respective SP to gain additional knowledge. To this end we are using two values—the *cost* and the *robustness*.

The cost for a property X is the length, in the number of states of a shortest witness, defined as:

$$cost^X = \begin{cases} min\{k \mid (s^1, \dots, s^k) \in W^X\} & \text{if } W^X \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

The value is of interest under the assumption that a shorter witness in general means a lower number of qualitative changes and in turn a slower energy consumption by the system. Even in the cases where the energy assumption is not realistic (*e.g.* due to different time scales) the cost value still reflects on how functionally complex the system is. Thus, one is usually interested in minimizing it. However it is important to note that the value is dependent on the encoding and automaton chosen, *e.g.* in Figure 2.2 the $cost^X = 6$ if interpreted as a normal BA (the witness in green) and $cost^X = 3$ if interpreted as a TBA (the witness in blue). This issue is addressed further in the thesis, but in general it signifies that the property metrics are mainly useful when comparing multiple models combined with the same property.

Of special focus in our approach is the set of witnesses with the minimal cost:

$$SW^X = \{(s^1, \dots, s^k) \mid cost^X = k > 0\} \subseteq W^X.$$

Consider the example in Figure 2.2—for the TBA encoding of the property the set SW^X holds the single path $(((0,2), b^1), ((0,1), b^2), ((0,0), b^3))$. While it can be argued that a single witness path is sufficient, having the set SW^X has additional advantages. First, it highlights the areas of the transition system with a strong requirement for ordering of events. Second, we use the set SW^X to compute the *robustness* w.r.t. the property satisfaction.

The robustness is an illustrative measure that states how likely it is that if we take a random walk of the length equal to the cost, we actually find a witness. The term robustness is widely used in systems biology [Kit04] to describe various features of a system, therefore our definition may not relate to the various robustness definitions in other works. This particular notion of robustness reflects on the ability of the model to keep the requested behaviour even though uncertainty is introduced to the model through the modelling framework. The non-determinism of the simulation arises in states where the qualitative behaviour in reality depends on quantitative nuances indistinguishable by our abstraction. The higher the robustness of the model w.r.t. a property, the less sensitive the model is to these quantitative nuances, respectively to perturbation in these. Note that there is no probability assigned to any transition, we therefore assume a uniform distribution, *i.e.* in a random walk the probability of taking any transition from a state s is given as one over the out-degree of s. For a path $w^X = (s^1, \dots, s^k) \in (S^X)^k$ we get the probability as:

$$prob(w^X) := \prod_{i=1}^{k-1} \frac{1}{succ^X(s^i)},$$

where $succ^X : S^X \to \mathbb{P}(S^X)$ is a successor function in X, formally

$$succ^X(s) = \{s' \mid s \to^X s'\}. \tag{2.5}$$

Since the probability of starting in any state is also uniformly distributed, the robustness is obtained as:

$$\mathit{robustness}^X := \frac{\sum_{w^X \in \mathrm{SW}^X} (\mathit{prob}(w^X))}{[\![I^X]\!]}.$$

For the TBA encoding in Figure 2.2 we have that $\mathit{robustness}^X = \frac{1}{3}$, since there is only one shortest witness w^X that has $\mathit{prob}(w^X) = 1$ and there are 3 states that match the initial measurement.

CHAPTER 3

Methods

In this Section we describe the new methods developed for the modelling and analysis of RN. Most of the methods were introduced in our conference publications [TSKS14, STS15a, STS15b]. The order of the section follows the workflow as presented in Figure 1.2 and provides a formal description of the methods in use. Technical details, algorithms, and proofs are omitted from this chapter for brevity and are later presented in Chapter 5, keeping the ordering of this chapter.

3.1 Enumeration

The first step of our workflow is the enumeration of all the feasible models. These models are subsequently stored in a database, where they can be further annotated or analysed.

In an unconstrained form, we consider every possible parametrization of the network, *i.e.* for a network $G = (V, E, \rho)$ we put:

$$\mathcal{K}^G = \prod_{v \in V} \{K_v : \Omega_v \to [0, \rho(v)]\}.$$

Note that for a network with Boolean components $[\![\mathcal{K}^G]\!] \in \mathcal{O}(2^{2^{[\![V]\!]}} \cdot [\![V]\!])$, as the number of possible boolean functions with n variables is 2^{2^n} [HR04]. For non-Boolean components, this number is even higher.

Usually, the regulatory networks are sparse, and the size of the parametrization space is orders of magnitude smaller than the worst case. In most cases, however, some reductions are necessary before the enumeration step.

3.1.1 Regulation Constraints

The most important pre-enumeration reduction tool are constraints on the semantics of the regulations, often also referred to as *edge labels*. Biologically, the nature of a regulation is quite often expected to follow certain rules, *e.g.*

to phosphorylate the target, meaning the target becomes *active*, therefore the regulation is *activating*.

There are two common basic regulation constraints [WSA12]—*activation* and *inhibition*— here denoted $\uparrow$ and $\downarrow$ respectively. These we formulate as predicates over regulations s.t.:

$$\begin{aligned}\uparrow(K,(u,t,v)) &\iff \exists\omega \in \Omega_v, \omega_u = [t_-,t) : K_v(\omega) < K_v(\omega_{u\leftarrow[t,t_+)}),\\ \downarrow(K,(u,t,v)) &\iff \exists\omega \in \Omega_v, \omega_u = [t_-,t) : K_v(\omega) > K_v(\omega_{u\leftarrow[t,t_+)}).\end{aligned}$$

We can then place these predicates on the individual regulations and remove those parametrizations for which the predicates do not evaluate to true. To this end we use the edge-labelling $\lambda : E \to \mathcal{L}(\uparrow,\downarrow)$ where $\mathcal{L}(\uparrow,\downarrow)$ is the language of propositional formulas with the predicates $\uparrow,\downarrow$. For example in Figure 2.1 we have for K that $\not\uparrow(K,(CompB,1,CompA)) \equiv false$ and we would therefore remove it under the labelling λ s.t. $\lambda(CompB,1,CompA) =\uparrow$.

There are altogether 16 different logical functions over 2 predicates. We use the following 12, which are also assigned names:

Name	Label
Activating	$\uparrow$
Activating Only	$\uparrow \wedge \neg \downarrow$
Not Inhibiting	$\neg \downarrow$
Inhibiting	$\downarrow$
Inhibiting Only	$\downarrow \wedge \neg \uparrow$
Not Activating	$\neg \uparrow$
Observable	$\uparrow \vee \downarrow$
Not Observable	$\neg \uparrow \wedge \neg \downarrow$
Monotone	$\neg(\uparrow \wedge \downarrow)$
Not Monotone	$\uparrow \wedge \downarrow$
Monotone Observable	$(\downarrow \wedge \neg \uparrow) \vee (\uparrow \wedge \neg \downarrow)$
Free	*true*

We do not use the remaining four possibilities, as they seem highly biologically irrelevant, or, in the case of $false$, directly infeasible.

Subsequently, we add the edge labelling as a constraint to the enumeration procedure so that:

$$\mathcal{K}^{G,\lambda} = \{K \in \mathcal{K}^G \mid K \models \bigwedge_{e \in E^G} \lambda(e)\}.$$

3.1.2 Direct Constraints

Even when using the edge constraints, the parametrization space will grow very quickly w.r.t. the number of regulators of a single component. For a Boolean component with 6 regulators one has already $2^{2^6} \approx 10^{19}$ possible Boolean functions. If 6 or more regulators are present, the parametrization space needs to be reduced by the modeller explicitly.

In our framework, it is possible to specify the parameter values directly by creating a set of constraints for each component such that:

$$\gamma : V \to \mathbb{P}(\{K_v(\omega) * n \mid \omega \in \Omega_v^{[K]}, * \in \{<, \leq, =, \neq, \geq, >\}, n \in [0, \rho(v)]\}).$$

As in the previous sections, we require these constraints to hold for a parametrization to be selected, *i.e.*:

$$\mathcal{K}^{G,\gamma} = \{K \in \mathcal{K}^G \mid K \models \bigwedge_{v \in V} \gamma(v)\}.$$

3.1.3 Normalization Constraint

The last reduction on the enumeration procedure comes from a subtle notion of the *dynamical equivalence* in the multi-valued models, which is detailed in Section 5.2. Here we limit ourselves to the statements that two parametrizations are dynamically equivalent *iff* they produce an equivalent transition system. Such an event is not likely to occur in general and it does not affect any of the studies in Chapter 4, meaning we list this constraint here only for completeness. Should there be some *dynamically equivalent* models, spurious behaviour may occur—as described in Section 5.2, where we show that some regulations may become observable even though they have no effect. To prevent such a behaviour, we remove all parametrizations that are not *normalized*, as explained in Section 5.3. The resulting *normalization constraint* is defined as

$$\eta(v) \equiv \bigwedge \omega \in \Omega_v : \mathrm{Norm}(K, v, \omega) = K_v(\omega).$$

where Norm is the *normalization algorithm* described in Section 5.3. Understandably a parametrization then belongs to a normalized parametrization space only if it satisfies the *normalization constraint*:

$$\mathcal{K}^{G,\eta} = \{K \in \mathcal{K}^G \wedge K \models \bigwedge_{v \in V} \eta(v)\}.$$

The normalization constraint is not used for the example in Figure 2.1, which therefore has 324 parametrizations. If it was used, there would be only 144. Both sets are available with the toy network data, as detailed in Appendix B.3.

3.2 Data Encoding

After constructing a formal description of the system, the second step is usually encoding of experimental data and behavioural observations about the system.

In our framework we encode each data set into a property that is composed of up to four different constraints:

- $\overrightarrow{M}^G$ is a sequence of measurements,
- $\overrightarrow{D}^G$ is a sequence of delta constraints,
- End^G is an ending,
- Exp^G is an experiment.

Each of the elements we describe in detail in the following.

As suggested in Chapter 2 we use the model checking technique to analyse the dynamics of each model and therefore it is necessary to encode the data in a way suitable for the MC method. Here we focus on the semantics of the encoding, the encoding itself with the related proofs is detailed in Section 5.4. In particular we will be expressing the semantics of the properties in terms of paths in TS, ignoring the MC procedure for the moment. For each constraint composing the property $P^{[K]}$ we therefore defined the meaning of the relation $\models$ and then ask if there exists a witness $w^T = (s^1, \dots, s^k) \in (\rightarrow^T)^{k-1}$ in T^K, *i.e.* if it holds that $T^K, w^T \models P^{[K]}$.

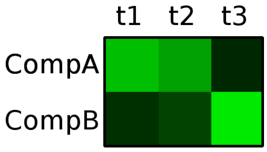
In the following we will use the term witness only for a path that models the property and is minimal w.r.t. the number of steps. In other words, we focus only on the shortest paths in the cases where there are witnesses of various lengths.

3.2.1 Measurements

A necessary step for formalization of data is to be able to express them in the terms of the discrete activity levels. The usual problem is to decide for a component, what is its threshold concentration, *i.e.* how many molecules of a certain component there must be in the system for us to consider it present.

Our framework does not promote or supplement any form of discretization in particular and all the data are expected to be provided in an already discretized form. In the studies in Chapter 4 we always describe how the particular discretization was done. For more details about discretization we refer to the review [GCC+15].

A single discretized measurement is described as $M^G = \prod_{v \in V^G} m_v$ where $m_v \subseteq [0, \rho(v)]$. Note that in particular $M^{[K]} \subseteq S^{[K]}$, therefore we express about states of a TS in terms of belonging to a measurement.



(a)

	$t1$	$t2$	$t3$
CompA	0.7	0.65	0.12
CompB	0.15	0.21	0.94

(b)

	M^1	M^2
CompA	1	0
CompB	0	2

(c)

(d)

$$\begin{aligned}
&\overrightarrow{M}^G = (M^1, M^2), M^1 = \{(1,0)\}, M^2 = \{(0,2)\} \implies \\
&w^T = ((1,0),(1,1),(1,2),(0,2)) \implies \\
&w^T_1 \in M^1, w^T_4 \in M^2 \implies \\
&I(w^T, \overrightarrow{M}^G) = (i^1, i^2, i^3) = (1,4,4) \implies \\
&T^K, w^T \models \overrightarrow{M}^G
\end{aligned}$$

(e)

$$\begin{aligned}
&\overrightarrow{D}^G = (D^1, D^2), D^1 = \{(\mathit{down}, \mathit{up})\}, D^2 = \{(\mathit{none}, \mathit{none})\} \implies \\
&w^T = ((1,0),(1,1),(1,2),(0,2)) \wedge I(w^T, \overrightarrow{M}^G) = (1,4,4) \implies \\
&(w^T_1)_{\text{CompA}} \geq (w^T_2)_{\text{CompA}} \geq (w^T_3)_{\text{CompA}} \geq (w^T_4)_{\text{CompA}} \wedge \\
&(w^T_1)_{\text{CompB}} \leq (w^T_2)_{\text{CompB}} \leq (w^T_3)_{\text{CompB}} \leq (w^T_4)_{\text{CompB}} \implies \\
&T^K, w^T \models \overrightarrow{D}^K
\end{aligned}$$

(f)

$$\begin{aligned}
&\mathit{End}^G = \mathit{open}, \mathit{Exp}^G = (\{0,1\}, \{0,1,2\}) \\
&T^K, ((1,0),(1,1),(1,2),(0,2)) \models P^G \equiv (\overrightarrow{M}^G, \overrightarrow{D}^G, \mathit{End}^G, \mathit{Exp}^G)
\end{aligned}$$

Figure 3.1: **(a)** An example of time-series measurements, where the brightness of each patch corresponds to the strength of gene expression. **(b)** Numerical values from the measurements. **(c)** Possible discretization of the data. Since the timepoints $t1$ and $t2$ would discretize to the same measurement, we merge them together. **(d)** Measurement sequence representation of the data, which are satisfiable by the T^K from Figure 2.1. The path w^T is the witness of satisfaction. Note that any path containing w^T would also be a witness, but any path shorter than w^T could not be. **(e)** Example delta constraints. **(f)** A full property satisfied by T^K. Also note that the system could not have a *stable* ending and that none more restrictive experiment would be allowed for these measurements.

A sequence of measurements is described via a measurements vector $\overrightarrow{M}^G = (M^1, \ldots, M^n)$ for some $n \in \mathbb{N}_1$. For a path $w^T \in (\rightarrow^T)^{k-1}$ to match the measurements there must be a sequence of states that match the measurements in the given order. For such a path to exist there must be a vector of indices

$$I(w^T, \overrightarrow{M}^{[K]}) = (i^1, \ldots, i^{n+1}) \in \mathbb{N}_1^n$$

where the following five conditions hold:

1. $\forall j \in [1, n] : w^T_{i^j} \in M^j$—each measurement has a state matching the measurement.
2. $\forall j \in [1, n], \forall l \in (j, n] : i^j \leq i^l$—the measurements are matched in order. Note however that multiple indices in a sequence may be the same.
3. $i^1 = 1$—the first measurement is matched immediately. This condition follows from the fact that we are focusing only on the shortest witnesses.
4. $\forall j \in (1, n], \forall l \in [i^{j-1}, i^j) : \neg w^T_l \in M^j$—the state that is associated to each measurement is the first one that matches the current measurement after the previous one has been matched. This condition is used for encoding of additional constraints.
5. $i^{n+1} = k$—this is a trivial condition that is useful for encoding of additional constraint and is utilized later in this chapter. In general, the index $n + 1$ is not used for features related to measurements, but for the behaviour after the last measurement.

Then we can express the satisfaction of the measurements as:

$$T^K, w^T \models \overrightarrow{M}^{[K]} \iff \exists I(w^T, \overrightarrow{M}^{[K]}).$$

3.2.2 Delta Constraints

Until now we were focusing on passing through measurement points, without any specifications on the behaviour between them. There are multiple related, biologically relevant constraints that can be implemented by further reducing the product structure. In particular, we may want to require a component not to change in between two measurements, or to change only once prohibiting unobserved oscillations. We define an additional constraint related to a measurement called *component delta*:

$$D^G \in \prod_{v \in V^G} \{up, down, stay, none\}.$$

The intuitive meaning for these constraints is:

- *up*: the component can not decrease its activity level,
- *down*: the component can not increase its activity level,
- *stay*: the component can not change its activity level,
- *none*: the component is not constrained.

This puts additional requirements on the satisfaction of a property by a TS. In particular a path $w^T = (s^1, \dots, s^k) \in (\rightarrow^T)^{k-1}$ satisfies the delta constraint D^G, written $w^T \models D^G$, *iff* for each $v \in V^G$ and all the pairs (s^i, s^{i+1}) s.t. $i \in [1, k)$ we have that:

- if $D_v = up$ then $s^i_v \leq s^{i+1}_v$,
- if $D_v = down$ then $s^i_v \geq s^{i+1}_v$,
- if $D_v = stay$ then $s^i_v = s^{i+1}_v$.

For a sequence of measurements, the sequence of delta constraints is then again a vector $\overrightarrow{D}^G = (D^1, \dots, D^n)$. We expect that the two sequences are the same length, *i.e.* $[\![\overrightarrow{D}^G]\!] = [\![\overrightarrow{M}^G]\!]$. If there were more measurements than deltas, we can add the *none* constraints and conversely, if there were more delta constraints, we can add empty measurements to assure that both have the same length. Then having the path $w^T \in (\rightarrow^G)^{k-1}$ and $I(w^T, \overrightarrow{M}^G) = (i^1, \dots, i^{n+1})$ we put:

$$T^K, w^T \models \overrightarrow{D}^{[K]} \iff \forall j \in [1, n] : (s^{i^j}, \dots, s^{i^{j+1}}) \models D^j.$$

Due to the minimality criterion (4) on the indices we know that the sequence between two indexed states is exactly from when the current measurement is satisfied until the following measurement is satisfied. In our understanding of monotonicity, that is the part of the transition system where the monotonicity is intuitively expected. Note that the last constraint is required after the last measurement and is practical only when used with the ending, as described in the following section.

Note that the the notion of monotonicity as we define it here is only one-sided. This means that we can either require for a component that it is monotonously increasing (*up*) or monotonously decreasing (*down*), but we can not say that it is only monotonous, without specifying the direction. The general monotonicity is more complicated and we do not provide it in our framework. For a possible implementation see [Kla15].

3.2.3 Ending

An ending is a logical variable $End^G \in \{open, stable, cyclic\}$.

The above encoding is sufficient if we want to pass through a set of measurements. We call this an *open* ending. For any $k \in \mathbb{N}_1$ and any path $w^T \in (\rightarrow^T)^{k-1}$ it holds that $T^K, w^T \models (End^{[K]} = open)$. When combined with a measurement series, $\overrightarrow{M}^G$ with $[\![\overrightarrow{M}^G]\!] = n$, we in general require that the witness of a property is a shortest such path, therefore:

$$\begin{aligned}&T^K, w^T \models (\overrightarrow{M}^{[K]}, End^{[K]} = open) \iff \\ &\exists I(w^T, \overrightarrow{M}^{[K]}) = (i^1, \dots, i^{n+1}) \wedge i^n = i^{n+1}.\end{aligned}$$

Sometimes it is however expected that the last measurement represents a stable state of the system. Many a biological system are expected to be in a stable configuration under standard circumstances [WVD12]. We call this a *stable* ending. For a path $w^T \in (\rightarrow^T)^{k-1}$ have that

$$T^K, w^T \models (End^G = stable) \iff w^T_k \rightarrow^T w^T_k.$$

The last, and the most complicated option is to require that after matching the last measurement, the system returns to the initial state, forming a cycle. We call this a *cyclic* ending and for a path $w^T \in (\rightarrow^T)^{k-1}$ we require

$$T^K, w^T \models (End^G = cyclic) \iff w^T_1 = w^T_k.$$

3.2.4 Experimental Setup

Lastly we focus on a configuration of an experiment. Experimentally, measurements are conducted under specific conditions, *e.g.* presence of certain nutrients in a medium, treatment with inhibitors, etc. Such conditions are usually expected to stay constant for the duration of the experiment. If they are explicitly modelled, *e.g.* with a component representing an inhibitor, we can enforce them by removing the states that do not match the corresponding component value from the TS, together with the respective transitions.

For a network G we denote $Exp^G_v \in \{[i,j] \mid 0 \leq i, \rho(v) \geq j\})$ the experimental setup for a component $v \in V^G$ and $Exp^G = (Exp^G_v)_{v \in V}$ the experimental setup of the whole network. We then restrict the state space s.t. $S' = \prod_{v \in V} Exp^{[K]}_v$. To make sure that the parametrizations are consistent with the experiment, we also transform a parametrization to have all target values in the reduced state space. For this we use the function $Reduce^G : \mathcal{K}^G \times \mathbb{P}(S^G) \rightarrow \mathcal{K}^G$, s.t. $Reduce^G(K, Exp^{[K]}) = K'$ where:

$$\forall v \in V^{[K]}, \forall \omega \in \Omega^{[K]}_v : K'_v(\omega) = \begin{cases} max(Exp^{[K]}_v) & \text{if } K_v(\omega) > max(Exp^{[K]}_v) \\ K_v(\omega) & \text{if } K_v(\omega) \in Exp^{[K]}_v \\ min(Exp^{[K]}_v) & \text{if } K_v(\omega) < min(Exp^{[K]}_v) \end{cases}$$

Structural Labels	
$sign^K(CompA, 1, CompB)$	+
$sign^K(CompB, 1, CompA)$	-
$sign^K(CompB, 2, CompB)$	-
$indegree^K(CompA)$	1
$indegree^K(CompB)$	2
$indegree^K(SUM)$	3
$bias^K(CompA)$	1
$bias^K(CompB)$	2
$impact^K(CompA, 1, CompB)$	0.905
$impact^K(CompB, 1, CompA)$	-1
$impact^K(CompB, 2, CompB)$	-0.302

(a)

Regulatory Functions	
RF^K_{CompA}	1&CompB{0}
RF^K_{CompB}	1&CompA{1}&CompB{2}|2&CompA{1}&CompB{0,1}

(b)

Property Labels	
$cost^K(P^{[K]})$	4
$robustness^K(P^{[K]})$	0.25

(c)

Figure 3.2: Illustrative labels for the toy example from Figure 2.1. **a)** All the possible structural labels. For clarity we use the symbol SUM to denote the sum of the *indegree* values. **b)** The *regulatory function* labels corresponding to the given parametrization. **c)** The dynamic labels for the property $P^{[K]} \equiv P^G$ as given in the figure 3.1f.

Note that changing the parametrization does not affect the transition system as such, since it only removes the transitions leading from the reduced state space, which change is already implied by the removal of the states outside the reduced state space. However it affects whether a state is considered stable. In particular a component that is reduced to one activity level is automatically stable, no matter what was its original parametrization. As an example, consider our toy network and its parametrization from Figure 2.1, which normally never stabilizes, however the following holds:

$$\overrightarrow{M}^G = (\{(0,0)\}), End^G = stable, Exp^G = (\{0,1\},\{0\}) \implies$$
$$T^{K'}, ((0,0),(1,0)) \models (\overrightarrow{M}^G, End^G, Exp^G)$$

as for each $\omega \in \Omega_{CompB}$ we have $K'_{CompB}(\omega) = 0$ and thus the state $(1,0)$ is then rendered stable by the experiment.

3.3 Labels

Having a model, one is usually interested in what its properties are, *e.g.* which of the regulations are effective, how it behaves dynamically etc. We call functions that provide such information *labels*, and discern these into two classes: *static* and *dynamic* labels. The static labels are derived from the model itself and are usually quite straightforward. The dynamic labels require construction of transition systems for their computation and therefore are exponentially harder to compute. To alleviate this difficulty we have devised optimized methods of their computation, which are presented in Section 5.5 and published in [SS15]. In Section 5.6 we then provide some additional details on the label computation in general.

Notation-wise, the domain of a label usually depends on the respective regulatory graph, we use the symbol l for a label in general, and l^K to denote that the label depends on the graph encoded by K and is evaluated under K.

All the labels are illustrated in Figure 3.2 on the toy example from Figure 2.1.

3.3.1 Sign

The sign label complements the regulation constraints and provides the actual regulation semantics for each model. We derive the label $sign^K : E^{[K]} \to \{0, +, -, 1\}$ where:

$$sign^K(e) = \begin{cases} 0 & \text{if } \neg \uparrow^K(e) \wedge \neg \downarrow^K(e), \\ + & \text{if } \uparrow^K(e) \wedge \neg \downarrow^K(e), \\ - & \text{if } \neg \uparrow^K(e) \wedge \downarrow^K(e), \\ 1 & \text{if } \uparrow^K(e) \wedge \downarrow^K(e). \end{cases}$$

Note that the 0 value means that the regulation has no effect on its target and could be removed without affecting the dynamics, which is utilized in the following label *indegree*. The value 1 describes the situation where a regulation has ambiguous semantics, not meeting the *monotionicity condition*, which is usually contrary to the expectation of the modeller about the system [Sno89].

3.3.2 Indegree

This is a simple label that counts the number of effective incoming regulations. Formally we denote $indegree^K : V^{[K]} \to \mathbb{N}_0$ the number incoming regulations with a non-zero sign, defined as:

$$indegree^K(v) = [\![\{(u, t, v) \in E \mid u \in V^{[K]}, Sign(u, t, v) \neq 0\}]\!].$$

Additionally, the function is extended to capture the sum of the *indegree* values of all the components, such that $indegree^K(V) = \sum_{v \in V^{[K]}} indegree^K(v)$. The sum of *indegree* values is of a special interest, as quite often one is interested in structures that are minimal w.r.t. number of regulations.

3.3.3 Cost

The *cost* label is drawn directly from the definition in Section 2.2.4. In Section 5.4 we show that for the properties specified in Section 5.4 it holds that a cost of a witness in a TS is equal to the cost of a witness in the respective SP. Formally, for $X^K = T^K \times P^{[K]}$ we have $\mathit{cost}^X = \mathit{cost}^T$. This is because the structure of the product guarantees that there are no spurious steps conducted, as we explain in section 5.4.

3.3.4 Robustness

The *robustness* is again drawn from definition in Section 2.2.4. Note that if a measurement is not satisfiable, then there are no witnesses, therefore by definition in Section 2.2.4 we have that $\mathit{cost}^X = 0 \iff \mathit{robustness}^X = 0$.

As it was with the cost, it is possible to guarantee that $\mathit{robustness}^X = \mathit{robustness}^T$. However, satisfying the equation would compromise the performance improvements achieved by the delta constraints and the experimental setup, as we will explain in Sections 5.4.4, 5.4.8. Since we use robustness as a comparison value, and relative differences between parametrizations are much more relevant than the individual absolute values, we relax this requirement. Thus, in the case that delta constraints or experimental setup are used, the following inequality holds $\mathit{robustness}^X \leq \mathit{robustness}^T$.

In informal terms, this inequality stems from the fact that disabled transitions (those that break the monotonicity or experiments) are not counted in the computation of robustness, which can however also be perceived as the expected behaviour.

3.3.5 Impact

The *impact* label represents the relation between a regulator and its target via the function $\mathit{impact}^K : E^{[K]} \to [-1, 1]$. For a regulation $(u, t, v) \in E^{[K]}$ we obtain the impact of u on v by computing the correlation of the activity level of the regulator and the respective parameter value. As we are interested only in parameters that are directly affected by this regulation, we take a subset of regulatory contexts on the border of the threshold value t. These we list as a vector

$$\Omega^{[K]}_{v,t} = (\omega \in \Omega^{[K]}_v \mid \omega_u \in \{[t_-, t), [t, t_+)\}).$$

To indicate presence or absence of the said regulation, we use the function $\mathit{pres}_u : \Omega^{[K]}_u \to [0, \rho(u)]$ that projects the activity interval of u on its lower boundary, *i.e.* if $\omega_u = [t_-, t)$ then $\mathit{pres}_u(\omega) = t_-$. The impact of (u, t, v) is then equal to the Pearson correlation coefficient between the image of $\Omega^{[K]}_v, t$

under $Pres_u$ and K_v:

$$impact^K(u,t,v) = \frac{cov(pres_u(\omega)_{\omega \in \Omega_{v,t}^{[K]}}, K_v(\omega)_{\omega \in \Omega_{v,t}^{[K]}})}{std(pres_u(\omega)_{\omega \in \Omega_{v,t}^{[K]}}) \cdot std(K_v(\omega)_{\omega \in \Omega_{v,t}^{[K]}})},$$

where *cov* is the covariance and *std* is the standard deviation. This value is quite helpful when one is searching for the key regulators of a certain component. The further the value is from 0, the more prominent the regulation is.

3.3.6 Bias

By the term *bias* we here mean the general tendency of a parametrization to push a component towards higher or lower activity levels. The label $bias^K : V^{[K]} \to [0,1]$ is obtained simply as

$$bias^K(v) = \sum_{\omega \in \Omega_v^{[K]}} K_v(\omega) \cdot [\![\Omega_v^{[K]}]\!]^{-1} \cdot \rho(v)^{-1}.$$

For a Boolean component this coincides with the notion as defined by other authors, *e.g.* [SK04].

As a component has in general more effect on the other components at higher activity levels, the *bias* label allows to distinguish the components whose presence seems to be crucial for the activity of the network.

3.3.7 Regulatory Function

While not being a label *per se* we also assign a logical *regulatory function*, providing a more human-readable description of a parametrization. In particular, we describe each partial parametrization as a Post Algebra [MT07] expression in a disjunctive normal form (DNF) of cardinality $n = max\{\rho(v) \mid v \in V^G\}$ for a RN G.

A Post Algebra expression $RF^{[K]}$ in DNF of cardinality n is in our case described using the grammar:

$$\begin{aligned} RF^{[K]} &\to \mathrm{M|M} \mid \mathrm{M} \\ \mathrm{M} &\to \mathrm{V\&A} \mid \mathrm{V} \\ \mathrm{V} &\to 0 \mid \cdots \mid n \\ \mathrm{A} &\to \mathrm{A\&A} \mid v\{\mathrm{L}\} \\ \mathrm{L} &\to \mathrm{LL} \mid \mathrm{V} \end{aligned}$$

where $v \in V^{[K]}, |, \&, \{, \}, 0, \ldots, n$ are terminals, and $RF^{[K]}$, M, V, A, L are non-terminals. The semantics are such that an atom, *i.e.* an expression of the form $v\{\mathrm{L}\}$, evaluates to n if the variable v is at a level listed in L and to 0 otherwise. The binary operator & evaluates to the smaller of its operands and the binary operator | evaluates to the bigger of its operands. *E.g.* consider the function in Figure 3.2b and an interpretation $\mathrm{CompA} = 1, \mathrm{CompB} = 1$. Then we can do the following valuation:

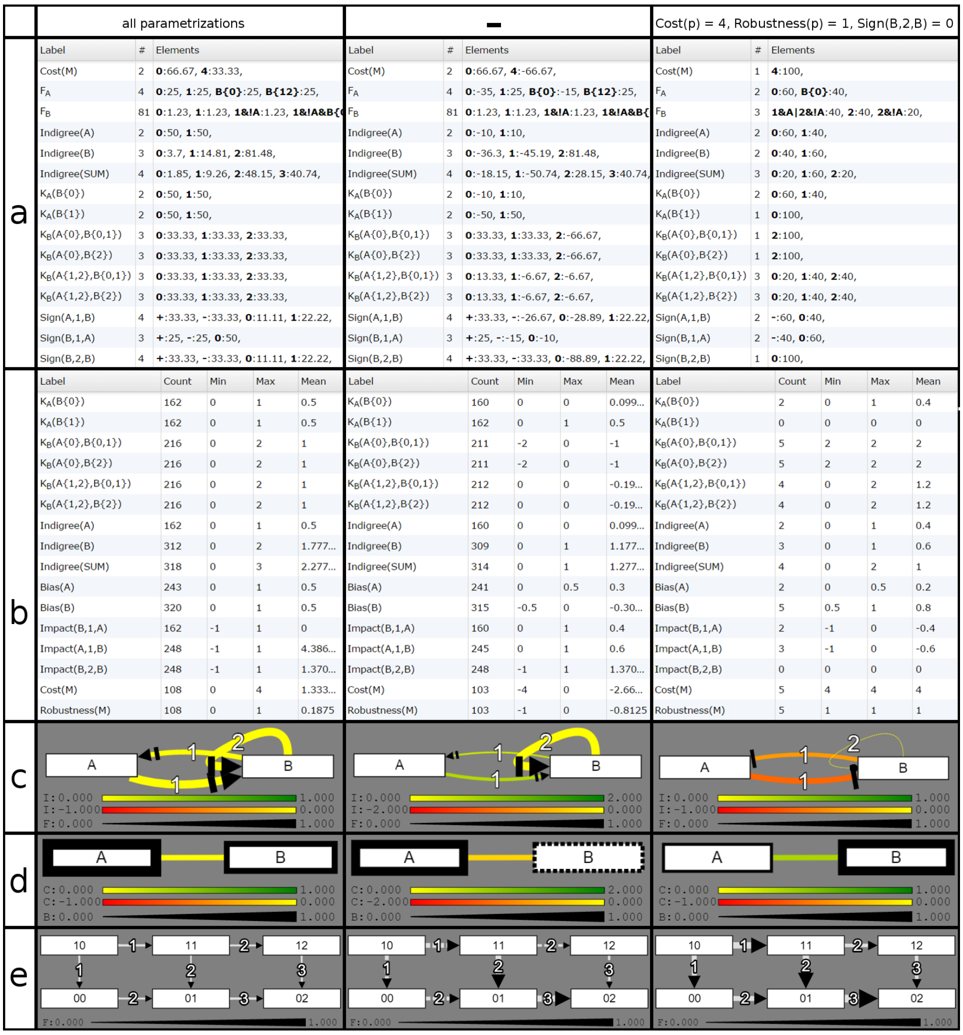
$$\begin{aligned}
&1\&\mathrm{CompA}\{1\}|2\&\mathrm{CompB}\{0,2\} \mapsto\\
&1\&2|2\&\mathrm{CompB}\{0,2\} \mapsto\\
&1\&2|2\&0 \mapsto\\
&1|2\&0 \mapsto\\
&1|0 \mapsto\\
&1.
\end{aligned}$$

Note that in the Boolean case the operator & corresponds to the logical conjunction, the operator | to the disjunction, $v\{1\}$ to the simple v, and $v\{0\}$ to $\neg v$.

3.4 Selection and Analysis

While the individual parametrizations can be at least partially ordered by the values of their labels, it is only seldom that a single parametrization would appear as the optimal one. Moreover, even if one aims to find a parametrization that scores the best in all the metrics, *i.e.* minimum cost, maximum robustness, minimum indegree etc., usually there are multiple parametrizations with the best score or those that are pairwise incomparable. We therefore focus on so-called *selections*, *i.e.* sets of parametrizations that fit certain criteria on the labels and analyse the whole selection. The computation of the analyses is mostly straightforward. For the few cases where it is not, we provide necessary details in Section 5.6.

Have a parametrization space $\mathcal{K}$ and a sequence of predicates $\Phi = \Phi_1, \dots, \Phi_n$ where $\Phi_i : \mathcal{K} \to \mathbb{B}$ for each $i \in [1, n]$. A *selection by* Φ we call the set of parametrizations denoted $\mathcal{K}^\Phi$ s.t. for each $K \in \mathcal{K}$ we have that $K \in \mathcal{K}^\Phi$ if and only if $\bigwedge_{i=1}^n \Phi_i(K)$ holds true. As the selections may contain millions, or even more parametrizations, approaches that allow to evaluate the whole selection at once are necessary to gain understanding of the nature of the selection. We present five different methods, each used to depict some of the labels in a manner that generalizes the values of the labels from members of the selection to the whole selection. A visual representation of such data is then called a *report.* Additionally, each of the reports features an individual method of *comparison*—having two different selections $\mathcal{K}^\phi$, $\mathcal{K}^\Psi$ we create a third report which illustrates the difference between the two selections. This we denote using the minus $(-)$ symbol, illustrating the fact that it is a *non-commutative* difference operation. All



all parametrizations

Label	#	Elements
Cost(M)	2	**0**:66.67, **4**:33.33,
F_A	4	**0**:25, **1**:25, **B{0}**:25, **B{12}**:25,
F_B	81	**0**:1.23, **1**:1.23, **1&!A**:1.23, **1&!A&B{**
Indigree(A)	2	**0**:50, **1**:50,
Indigree(B)	3	**0**:3.7, **1**:14.81, **2**:81.48,
Indigree(SUM)	4	**0**:1.85, **1**:9.26, **2**:48.15, **3**:40.74,
K_A(B{0})	2	**0**:50, **1**:50,
K_A(B{1})	2	**0**:50, **1**:50,
K_B(A{0},B{0,1})	3	**0**:33.33, **1**:33.33, **2**:33.33,
K_B(A{0},B{2})	3	**0**:33.33, **1**:33.33, **2**:33.33,
K_B(A{1,2},B{0,1})	3	**0**:33.33, **1**:33.33, **2**:33.33,
K_B(A{1,2},B{2})	3	**0**:33.33, **1**:33.33, **2**:33.33,
Sign(A,1,B)	4	**+**:33.33, **-**:33.33, **0**:11.11, **1**:22.22,
Sign(B,1,A)	3	**+**:25, **-**:25, **0**:50,
Sign(B,2,B)	4	**+**:33.33, **-**:33.33, **0**:11.11, **1**:22.22,

Label	Count	Min	Max	Mean
K_A(B{0})	162	0	1	0.5
K_A(B{1})	162	0	1	0.5
K_B(A{0},B{0,1})	216	0	2	1
K_B(A{0},B{2})	216	0	2	1
K_B(A{1,2},B{0,1})	216	0	2	1
K_B(A{1,2},B{2})	216	0	2	1
Indigree(A)	162	0	1	0.5
Indigree(B)	312	0	2	1.777...
Indigree(SUM)	318	0	3	2.277...
Bias(A)	243	0	1	0.5
Bias(B)	320	0	1	0.5
Impact(B,1,A)	162	-1	1	0
Impact(A,1,B)	248	-1	1	4.386...
Impact(B,2,B)	248	-1	1	1.370...
Cost(M)	108	0	4	1.333...
Robustness(M)	108	0	1	0.1875

−

Label	#	Elements
Cost(M)	2	**0**:66.67, **4**:-66.67,
F_A	4	**0**:-35, **1**:25, **B{0}**:-15, **B{12}**:25,
F_B	81	**0**:1.23, **1**:1.23, **1&!A**:1.23, **1&!A&B{**
Indigree(A)	2	**0**:-10, **1**:10,
Indigree(B)	3	**0**:-36.3, **1**:-45.19, **2**:81.48,
Indigree(SUM)	4	**0**:-18.15, **1**:-50.74, **2**:28.15, **3**:40.74,
K_A(B{0})	2	**0**:-10, **1**:10,
K_A(B{1})	2	**0**:-50, **1**:50,
K_B(A{0},B{0,1})	3	**0**:33.33, **1**:33.33, **2**:-66.67,
K_B(A{0},B{2})	3	**0**:33.33, **1**:33.33, **2**:-66.67,
K_B(A{1,2},B{0,1})	3	**0**:13.33, **1**:-6.67, **2**:-6.67,
K_B(A{1,2},B{2})	3	**0**:13.33, **1**:-6.67, **2**:-6.67,
Sign(A,1,B)	4	**+**:33.33, **-**:-26.67, **0**:-28.89, **1**:22.22,
Sign(B,1,A)	3	**+**:25, **-**:-15, **0**:-10,
Sign(B,2,B)	4	**+**:33.33, **-**:33.33, **0**:-88.89, **1**:22.22,

Label	Count	Min	Max	Mean
K_A(B{0})	160	0	0	0.099...
K_A(B{1})	162	0	1	0.5
K_B(A{0},B{0,1})	211	-2	0	-1
K_B(A{0},B{2})	211	-2	0	-1
K_B(A{1,2},B{0,1})	212	0	0	-0.19...
K_B(A{1,2},B{2})	212	0	0	-0.19...
Indigree(A)	160	0	0	0.099...
Indigree(B)	309	0	1	1.177...
Indigree(SUM)	314	0	1	1.277...
Bias(A)	241	0	0.5	0.3
Bias(B)	315	-0.5	0	-0.30...
Impact(B,1,A)	160	0	1	0.4
Impact(A,1,B)	245	0	1	0.6
Impact(B,2,B)	248	-1	1	1.370...
Cost(M)	103	-4	0	-2.66...
Robustness(M)	103	-1	0	-0.8125

Cost(p) = 4, Robustness(p) = 1, Sign(B,2,B) = 0

Label	#	Elements
Cost(M)	1	**4**:100,
F_A	2	**0**:60, **B{0}**:40,
F_B	3	**1&A|2&!A**:40, **2**:40, **2&!A**:20,
Indigree(A)	2	**0**:60, **1**:40,
Indigree(B)	2	**0**:40, **1**:60,
Indigree(SUM)	3	**0**:20, **1**:60, **2**:20,
K_A(B{0})	2	**0**:60, **1**:40,
K_A(B{1})	1	**0**:100,
K_B(A{0},B{0,1})	1	**2**:100,
K_B(A{0},B{2})	1	**2**:100,
K_B(A{1,2},B{0,1})	3	**0**:20, **1**:40, **2**:40,
K_B(A{1,2},B{2})	3	**0**:20, **1**:40, **2**:40,
Sign(A,1,B)	2	**-**:60, **0**:40,
Sign(B,1,A)	2	**-**:40, **0**:60,
Sign(B,2,B)	1	**0**:100,

Label	Count	Min	Max	Mean
K_A(B{0})	2	0	1	0.4
K_A(B{1})	0	0	0	0
K_B(A{0},B{0,1})	5	2	2	2
K_B(A{0},B{2})	5	2	2	2
K_B(A{1,2},B{0,1})	4	0	2	1.2
K_B(A{1,2},B{2})	4	0	2	1.2
Indigree(A)	2	0	1	0.4
Indigree(B)	3	0	1	0.6
Indigree(SUM)	4	0	2	1
Bias(A)	2	0	0.5	0.2
Bias(B)	5	0.5	1	0.8
Impact(B,1,A)	2	-1	0	-0.4
Impact(A,1,B)	3	-1	0	-0.6
Impact(B,2,B)	0	0	0	0
Cost(M)	5	4	4	4
Robustness(M)	5	1	1	1

Figure 3.3: Reports produced by TREMPPI for the toy network G from Figure 2.1 and the property P^G from Figure 3.1. For the interactive version, please refer to Appendix B.3. **Left:** Reports for $\mathcal{K}^G$. **Right:** Reports for $\mathcal{K}^{G,\Psi}$ with $\Psi \equiv cost^K(P^G) = 4 \wedge robustness^K(P^G) = 1 \wedge sign^K(B,2,B) = 0$. **Middle:** A comparison left - right. **a)** *A qualitative* report. The label F_B is not fully listed. **b)** A *quantitative* report. **c)** A *regulation* graph. **d)** A *correlation* graph. **(e)** A witness graph for the property P^G.

the reports are illustrated in Figure 3.3 on the example network from Figure 2.1. Each report provides a comparison between the set of all 324 parametrizations, *i.e.* a selection by $\Phi \equiv true$ and a selection where the M^K from Figure 3.2 has minimal cost and maximal robustness and where the self-regulation of the component B is not present, *i.e.* a selection by $\Psi \equiv (cost^K(series) = 4 \wedge robustenss^K(series) = 1 \wedge sign^K(B,2,B) = 0)$.

3.4.1 Explicit Qualitative Report

The first tool we employ is a *qualitative* summary, which describes an image of a label in the selection, *i.e.* all the distinct label values that appear in the selection and their frequency in percent. Have a label $l : Y \to Z$, where Y, Z are some sets and a selection $\mathcal{K}^\Phi$. For example in the case $l = sign$, we have $Y = E$ and $Z = \{0, +, -, 1\}$. For each value $y \in Y$ we then set:

$$\begin{aligned} qual(\mathcal{K}^\Phi, l, y) &= (size(\mathcal{K}^\Phi, l, y), elems(\mathcal{K}^\Phi, l, y)),\\ size(\mathcal{K}^\Phi, l, y) &= [\![elems(\mathcal{K}^\Phi, l, y)]\!],\\ elems(\mathcal{K}^\Phi, l, y) &= \{(z, q) \mid q = [\![\{K \in \mathcal{K}^\Phi \mid l^K(y) = z\}]\!] \cdot 100 \cdot [\![\mathcal{K}^\Phi]\!]^{-1}\}. \end{aligned}$$

A comparison of two qualitative summaries that are based on the selections $\mathcal{K}^\Phi, \mathcal{K}^\Psi$, is obtained by subtracting the two pairs, where $elems(\mathcal{K}^\Phi, l, x) - elems(\mathcal{K}^\Psi, l, x)$ is computed as:

$$\{(z, q^\Phi - q^\Psi) \mid (z, q^\Phi) \in elems(\mathcal{K}^\Phi, l, y), (z, q^\Psi) \in elems(\mathcal{K}^\Psi, l, y)\}.$$

Note that it some feature z is not present in both the sets, the resulting subtraction will have less elements than the original qualitative summaries. Since the set of parametrizations is finite, all values have finite domain and are thus suitable to this form of presentation. However, in the case of labels that project to rational numbers, *i.e.* robustness, bias, and impact values, the size of the image quite often threatens to be almost as big as the selection itself, therefore we chose not to include them in the qualitative report.

3.4.2 Explicit Quantitative Report

Similarly to the previous, we summarize the overall nature of quantitative labels, *i.e.* those whose image is a subset of rational numbers, using the quadruple:

$$\begin{aligned} quan(\mathcal{K}^\Phi, l, y) &= (count(\mathcal{K}^\Phi, l, y), min(\mathcal{K}^\Phi, l, y),\\ &\qquad max(\mathcal{K}^\Phi, l, y), mean(\mathcal{K}^\Phi, l, y)),\\ count(\mathcal{K}^\Phi, l, y) &= [\![\{K \in \mathcal{K}^\Phi \mid l^K(y) \neq 0\}]\!],\\ min(\mathcal{K}^\Phi, l, y) &= min\{l^K(y) \mid K \in \mathcal{K}^\Phi\},\\ max(\mathcal{K}^\Phi, l, y) &= max\{l^K(y) \mid K \in \mathcal{K}^\Phi\},\\ mean(\mathcal{K}^\Phi, l, y) &= \sum_{K \in \mathcal{K}^\Phi} l^K(y) \cdot [\![\mathcal{K}^\Phi]\!]^{-1}. \end{aligned}$$

The difference between the quantitative reports of two selections $\mathcal{K}^\Phi$, $\mathcal{K}^\Psi$ is then set simply as the subtraction of the two quadruples.

Note that the *count* has a somewhat special meaning, as the 0 value is of particular interest for some of the labels. In the case of cost for example, it denotes that the respective measurement series is not satisfiable or for sign it states that the edge is absent.

3.4.3 Inferred Regulation Graph

Based on impact and sign, we can summarize the average effect of regulations of a sample. The impact can be easily extended from a parametrization to a sample for each $e \in E$ as

$$\mathit{impact}^{\mathcal{K}^\Phi}(e) = \sum_{K \in \mathcal{K}^\Phi} \mathit{impact}^K(e) \cdot [\![\mathcal{K}^\Phi]\!]^{-1}.$$

For the sign we take a supremum under the partial ordering $0 < - < 1$ and $0 < + < 1$, *i.e.* for any $e \in E$ we set:

$$\mathit{sign}^{\mathcal{K}^\Phi}(e) = \mathit{sup}\{\mathit{sign}^K(e) \mid K \in \mathcal{K}^\Phi\}.$$

Lastly we depict the frequency of a regulation, which states how often a regulation is functional, *i.e.* has a non-zero sign, formally:

$$\mathit{frequency}^{\mathcal{K}^\Phi}(e) = [\![\{K \in \mathcal{K}^\Phi \mid \mathit{sign}^K(e) \neq 0\}]\!].$$

Visually, the impact value is mapped to a color gradient of the regulation edge with the color red representing the value -1, yellow representing 0, and green representing 1. The frequency is mapped to the width of an edge. When the frequency is equal to 0, the edge is then displayed as dotted. Lastly, the sign is reflected in the shape of the head of the edge. The $+$ sign is mapped to a pointed arrow shape, the $-$ to a rectangle shape (also known as *blunt arrow*), the 1 to a combination of both and the 0 to a circle.

To create a comparison, the impact and frequency values are directly subtracted. The sign can not be clearly interpreted in the comparison and for simplicity it is kept from the minuend. Note that the subtraction means that the result lies behind the original boundaries of a value. The color gradient is therefore stretched to the range $[-2, 2]$ and a negative frequency value is depicted by a dashed edge.

3.4.4 Correlation Graph

Similarly to the regulation graph we also create a correlation graph, based on the bias label. The label extended similarly to the impact label, *i.e.*:

$$\mathit{bias}^{\mathcal{K}^\Phi}(v) = \sum_{K \in \mathcal{K}^\Phi} \mathit{bias}^K(v) \cdot [\![\mathcal{K}^\Phi]\!]^{-1}.$$

Additionally, one is usually interested in whether there is a relation between activities of multiple components, *e.g.* if one component seems to be taking over if another is missing. This is obtained as the correlation between the bias of individual components in a sample, *i.e.* for $u, v \in V^{[K]}$:

$$\mathit{correlation}^{\mathcal{K}^\Phi}(v, u) = \frac{\mathit{cov}(\mathit{bias}^K(v)_{K \in \mathcal{K}^\Phi}, \mathit{bias}^K(u)_{K \in \mathcal{K}^\Phi})}{\mathit{std}(\mathit{bias}^K(v)_{K \in \mathcal{K}^\Phi}) \cdot \mathit{std}(\mathit{bias}^K(u)_{K \in \mathcal{K}^\Phi})}.$$

The correlation value is mapped to a color gradient in the same manner as the impact value in the regulation graph. The bias value is mapped to the width of the border of the respective component in a manner similar to the edge width in the case of the frequency value.

To create a comparison both values are simply subtracted.

3.4.5 Witness Graph

Lastly, we provide a tool for displaying the witnesses of the individual properties. In Section 3.2 we explain how a property is tied to a path in a TS: its witness. Subsequently, in order to explain how a property is satisfied, we aim to present the user with a witness in form of an ordered graph. In Section 2.2.4 we defined the set SW^X of all the shortest witnesses, that serves as the basis for this tool. However, in practice we are not interested in the configuration of the controlling BA, but rather about how the witness passes through a TS, therefore we display each state together with its index in a witness. We use the term *trace* to describe a set of transitions belonging to some set SW^X. For a property $P^{[K]}$ encoded by the SP $X^K = T^K \times B^{[K]}$:

$$\begin{aligned}&\mathit{trace}^K(P^{[K]}) = \\ &\{((s,j,s')) \mid \exists w \in SW^X, \exists b,b' \in S^B : w_j = (s,b) \wedge w_{j+1} = (s',b')\}.\end{aligned}$$

For a witness we allow to combine multiple properties, as the individual automata states are removed from the traces and therefore states from multiple properties can be overlapped. For a sequence of properties $\overrightarrow{P}^{[K]} = (P^1,\dots,P^n)$ we create the witness:

$$\begin{aligned}\mathit{witness}^K(\overrightarrow{P}^{[K]}) = \{((s,j,s'),o) \in S^{[K]} \times \mathbb{N}_1 \times S^{[K]} \times \mathbb{Q} \mid o = \frac{[\![L]\!]}{n} > 0\},\\ L = \{l \mid (s,j,s)) \in SW^X, X^K = T^K \times {}^*\mathrm{BA}(P^l)\},\end{aligned}$$

where the value o states the occurrence of the transition, *i.e.* how many properties have this transition. Note that we take into account only edges with the occurrence higher than 0. The witness can be then easily extended to a selection of parametrizations by taking the mean, with:

$$\mathit{witness}^{\mathcal{K}^\Phi}(P^{[K]}) = \sum_{K \in \mathcal{K}^\Phi} \mathit{witness}^K(P^{[K]}) \cdot [\![\mathcal{K}^\Phi]\!]^{-1},$$

where

$$\begin{aligned}((s,j,s'),o) + ((s,j,s'),o') = ((s,j,s'),o+o'),\\ ((s,j,s'),o) \cdot q = ((s,j,s'),o \cdot q).\end{aligned}$$

The witness report is then created by plotting the witness set as a graph with the occurrence being mapped to a width of a transition. Unfortunately the transition-based representation does not allow for properties with cost equal to 1 to have a witness displayed, more precisely the witness is an empty graph. However since we are generally interested in paths, it is of a little consequence.

3.5 TREMPPI

We conclude this chapter with a short description of the TREMPPI tool. For details on its usage see Appendix B.1.

TREMPPI combines in itself all the methods presented in this and the previous chapter. The main aim is to provide a modelling support together with a model checker and analytical features in a single platform.

The core of TREMPPI is a multitude of optimized *C++* [Str97] programs that separately conduct the individual steps of our workflow. We split these into 5 classes, roughly corresponding to the organization of Chapter 3:

1. The tool for construction of the model space.
2. Labelling tools for those labels that do not relate to any dynamical properties (static labels).
3. Labelling tools for model checking related (dynamic) labels.
4. Reporting tools. These mostly only gather data for a selection.
5. Maintenance tools.

Tools of each class have, in general, similar requirements on running time and storage space, which we will discuss in more detail in Section 5.6.

Each of the programs can be execute via a unified binary, which outputs to a terminal and reports the progress of the tool in percent. The data for both input are required to be in a `data` folder, which is also used for the output. The individual models, together with their labels, are stored in a single *SQlite3* [OA10] database, where each label value occupies a single column. The data from the reports are stored in *JSON* [Bra14] files in sub-folders named in correspondence to these reports.

However, we do not expect the user to execute the commands from a terminal or access the files on the file system, or only in exceptional cases. Instead we provide a visual interface capable of executing the individual tools and of execution management. The main advantage of the visual interface is that both the data input and the output can be represented in an intuitive, comprehensible way, rather than in extensive text files. Examples of the visual output for the reports are shown in Figure 3.3.

From the technical perspective the interface of TREMPPI is provided as a *HTML* [Fla06] page that uses the *JavaScript* [Fla06] language for creation of dynamic features. Due to security limitation, JavaScript code is not allowed to access the local files [Fla06]. To overcome this limitation, TREMPPI has a third component that connects the computational tools and the visual interface: a *Python* [van95] server. This server is started by execution of the TREMPPI binary and exists until TREMPPI is exited. The visual interface is then sending asynchronous requests to the server, which in turn either executes local sub-programs or communicates with the file systems and the sends a response back.

While having the computational tools separately from the visual interface certainly brings some technical difficulties, there are also considerable advantages, of which we point two. First and foremost, after a project is finished, it can be easily distributed as a plain HTML page, without the necessity for the recipient to have TREMPPI installed. This is also utilized for all the applications in this thesis. Second, in the future we plan to make TREPPI available as a public server, as we have already proposed in [SKSv13]. We therefore already have most of the components necessary for the task and only a few features, like multi-user management, need to be added.

CHAPTER 4

Applications

In this section we present three case studies that have been co-developed by the author and demonstrate different aspects of TREMPPI. Understandably, each of these studies was conducted with the aim of drawing relevant biological information. However, the focus of this thesis is on the mathematical and software side of the problem, hence we limited the biology only to the parts which are necessary for understanding each of the studies.

4.1 EGFR Signalling

The first study we present is on the EGFR signalling network, which was done in cooperation with Kirsten Thobe and published in [STS15a]. The main focus of this study is application of the efficient encoding of experimental data, as presented in Section 3.2.

For this study we utilize a comprehensive data set provided by Klinger et al. in [KSFG$^+$13]. In the study, human colorectal cancer cell lines were treated with stimuli and inhibitors in order to elucidate the underlying network structure of the pathway using a semi-quantitative modelling approach. Here, we generate and analyse comprehensive model pools for the different cell lines and evaluate the performance of the tool. For the details on the biological part of the study, please refer to the original publication [STS15a]. All the data of this study are available as attachment to this thesis, see Appendix B.4.

4.1.1 Model Building

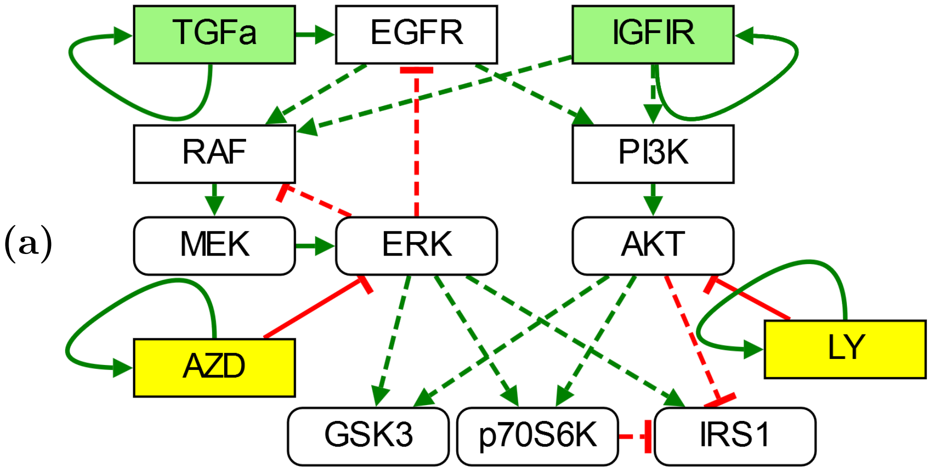
Based on the model of [KSFG$^+$13] we constructed a Boolean network, depicted in Fig. 4.1. We kept the original components and regulations, with a few exceptions. As the IGF1 stimulus is the only regulator of IGFIR we know that IGFIR copies its value and therefore we modelled the stimulation directly on IGFIR, removing IGF1 completely. Additionally, p70S6K is depicted as activator of IRS1, however based on [TJ09] we modelled it as an inhibition. The same for AKT which is known to repress IRS1 indirectly through mTorC1 [TJ09]. Note that these changes are to regulations of IRS1 only, which is an output component and therefore can not affect the upstream feedback loops. Any resulting inconsistencies with [KSFG+13] should therefore be localised to IRS1. Since the data originates from cancer cells, we accounted for possible disruptions in the network due to mutations by not requiring regulations to be functional. However, stimuli and inhibitions as well as components with a single regulator (MEK, AKT) were set as always functional. In the data there are two stimuli, TGFa and IGF1, and two effective inhibitors, MEK inhibitor AZD6244 and the PI3K inhibitor LY294002. There are two more inhibitors in the original data set on GSK3 and IKK, which were found to be non-effective and therefore neglected here. In our model, we set the stimuli as $Exp^{\mathrm{EGFR}}_{\mathrm{TGFa}} = \{1\}$ if TGFa is stimulated in an experiment and $Exp^{\mathrm{EGFR}}_{\mathrm{TGFa}} = \{0\}$ otherwise, and the same for IGFIR. The inhibitors do not remove the targets from the system, only prohibit their effect on the down-stream components. We therefore added them as extra components LY and AZD, and modelled them analogously to stimuli. Additionally we set the regulatory functions $K_{\mathrm{ERK}} \iff s_{\mathrm{MEK}} = 1 \wedge s_{\mathrm{AZD}} = 0$ and $K_{\mathrm{AKT}}(s) \iff s_{\mathrm{PI3K}} = 1 \wedge s_{\mathrm{LY}} = 0$ to enforce the correct inhibition semantics. After having resolved all the edge constraints, we obtained a model pool $\mathcal{K}^{\mathrm{EGFR}}$ with 259200 models. Note that the inhibitors and stimuli are fixed components, they do not contribute to the size of the state space, which then only has $2^9 = 512$ states instead of 2^{13}.

In their experiments, Klinger et al. used a high-throughput immunoblotting method, called Luminex assay, which measures intensities of labelled antibodies that bind the phosphorylated components, showing their activity (for a detailed description see [KSFG+13]). Here, we used a reduced data set containing experiments on 5 human colorectal cancer cell lines. Each of the cell lines was treated with each pairwise combination of one stimuli (TGFa, IRS1, no stimulus) and one inhibitor (AZD, LY, no inhibitor), which were then compared to the measurements before treatment. Since the configuration without stimulus and inhibitor is not expected to change, we did not include it.

4.1.2 Data Encoding

Having the regulatory network, the next task is to encode the data for each experiment. In our data set, there were multiple measurements available for some of the experiments. To be able to use the data further, we have taken the mean of all measurements for each of the experiments.

The next step is then to discretize the data. As we suggested in Section 3.2.1, one usually uses a software which creates a threshold value that separates the range of measured values for each component. In our data set some of the values however almost do not change between measurements



(b)

cell line	TGFa	IGFIR	AZD	LY
LIM1215	1	0	1	0
LIM1215	0	1	0	1
HCT116	0	1	0	1
SW403	0	1	0	1
SW480	0	1	0	1
HT29	1	0	1	0

(c)

cell line	transient	partially stable	stable
LIM1215	180000	6100	40
HCT116	129600	5580	2
SW403	180000	111000	840
SW480	136800	74670	36
HT29	163800	101010	216

Figure 4.1: **(a)** EGFR signalling Boolean network. The full green edges are constrained as *activating only*, the dashed green as *not inhibiting*, the full red as *inhibiting only*, and the dashed red as *not activating*. Stimuli are on the green fields, inhibitors on the yellow ones. The measured nodes are semi-oval. **(b)** Experimental set up causing logical inconsistencies after discretization. **(c)** Sizes of a parametrization sets matching the data from all the consistent experiments for each cell. Monotone property sets are not listed as monotonicity did not cause any reduction.

being *e.g.* at a plateau and therefore should not be assigned with different states. To avoid this separation, we focused on a fold change, which shows the measured activity of the treated sample relative to the measurement before the treatment. Here, we rely on an assumption that a fold change of two or more is significant, which is to the best of our knowledge a common practice and in our case seems to produce a good separation.

Since the focus of this study is on evaluating the effect of regulatory influences, we assigned Boolean values to the component measurements consistent with the nature of the fold changes found in the data. If we observed an increase by a factor of at least two, we assigned the value 0 to the measurement before and 1 after the treatment. Analogously, we encoded a decrease by a factor of at least two. If the change factor is less than two, we did not specify the value, but required the component to be stable, as explained in Section 3.2.2. In this approach, interpretation of the qualitative dynamics heavily focuses on the component changes indicating actively regulated behaviour, as is our intention. Note that it therefore differs from the often employed interpretation of the Boolean component values as an abstraction for ranges of quantitative values. In our approach the same quantitative value might be assigned different Boolean counterparts depending on the observed component behaviour in the respective experiments. In our opinion, this does not pose a problem, since we are focusing on the qualitative dynamics and thus the values 0 and 1 can be viewed as labels of qualitative change, rather than ranges of quantitative values. Presumably, if a component can undergo both a significant increase and a decrease in its concentration, such mechanics should be allowed by the network without contradicting the effects of the regulations.

As we considered 8 treatments for 5 cell lines, we obtained altogether 40 measurement pairs. In [KSFG$^+$13] the authors argue that at the time of the measurements the system is expected to reach a stable plateau. However, Figure S1 therein shows that the kinetics of some components have an unstable behaviour after the time point of measurement. To investigate the impact of the steady state assumption, we created a *stable* and *transient* (*i.e.* not required to be stable) version of each time series, as explained in Section 3.2.3. Additionally, we were interested in what effects the monotonicity constraints have on the results. We therefore also considered for each property a version where all the components that are measured and not stable are required to be monotonous in their behaviour. By combining the treatments, cell lines and constraints we obtained 160 properties. The properties are listed in the supplementary files.

From the optimization perspective, we can reduce the encoding TBA just to 1 state, as will be explained in Section 5.4.5, keeping the size of the product at the 512 states.

4.1.3 Results

Initially we found that each of the cell lines shows inconsistencies in at least one measurement pair. In each of these, the experimental setup, listed in Figure 4.1b, requires that a component whose activator was inhibited undergoes itself an activation, which is logically inconsistent. For example cell line SW403 shows with IGF1 stimulus an over 4-fold increase in concentration of AKT under inhibition of PI3K, its only activator. This is still comparably lower than the about 12-fold increase without the inhibition, showing that the inhibitor is working, but the dose is not sufficient to lower the activity of AKT to the threshold of being inactive after discretizing. Since dose-dependent processes are not considered in this formalism, we removed the respective experiments from the testing set. After the removal we have sets of 7 measurement pairs for each cell line except LIM1215 where there are only 6. We therefore further used only 34 measurement pairs, yielding 136 properties when combined with different path constraints. In Figure 4.1c, column *transient*, which represents the weakest assumption concerning the stability of the system, shows how many members of $\mathcal{K}^{\mathrm{EGFR}}$ fit all the measurements for each respective time series. Note that each set remains more than one half in size compared to the set of models consistent with the constraints derived from the network structure, suggesting that the topology itself already strongly determines the dynamics.

In [KSFG$^+$13] the modular response analysis (MRA) method was used to identify non-functional connections in the network for the different cell lines. Here, we aimed to compare the topologies of their resulting networks with the topologies that occur in our model pools. To improve comparability, we used a stability requirement for the measurements in each cell line to account for the steady-state assumption necessary for the MRA approach. The sizes of the parametrization sets are listed in Figure 4.1c-*stable*. Note that there is a much stronger reduction than in the transient case, suggesting that the stability requirement is indeed very strong for this network, presumably due to the negative feedback mediated by ERK.

As our method allows for testing transient states, and the time series measurement in Figure S1 of [KSFG$^+$13] illustrates that AKT and ERK may not be in steady state at the time point of measurement, we also created a *partially stable* selection. Here, those components which are not stimulated are assumed to be in steady-state. Stimulated samples are allowed to be in a transient state, since their last treatment was shortly before sampling. In our opinion, this scenario accounts for the most biologically realistic assumptions and we used it as the basis for the subsequent analysis (see Figure 4.1c-*partially stable*).

4.1.4 Performance

Aside from the biologically motivated analysis, we also used the case study to evaluate performance of our constraint encodings. We have executed the validation in batches for each set of 40 properties with different path constraints. The execution time for the transient properties was 9330s, for the monotone 9080s, for the stable 10646s, and for the stable monotone 10027s. The program did not use more than 7MB of memory at any time. The program was executed as a single-threaded instance on a Debian 3.2.65 workstation with a processor i5-2400S, 2.5GHz, and 4GB RAM.

To compare with existing methods, we have also tested execution with a script [Kla15] that called the NuSMV model checker using a respective LTL formula for only one of the 160 properties and the computation took roughly a week, illustrating that customization was necessary for the problem to be solvable in a reasonable time.

4.2 HGF Signalling

The second case study is aimed at providing a demonstration of the analytical methods like labels and reports. For this study we have utilized the data provided by D'Allesandro *et al.* in their study of hepatocyte growth factor (HGF) signalling [DSM$^+$15]. The model was prepared in cooperation with Kirsten Thobe and published in [STS15b]. All the produced data are again available in the attachment, for details please see Appendix B.5.

4.2.1 Network and Properties

In the original article the authors constructed a core network, illustrated in Figure 4.2, with a set of regulations that are with high certainty present. Afterwards, a qualitative method is used to find an optimal structure that combines the core network with a subset of possible edges. To this end the authors obtained a rich set of experimental data, which they later discretized to meet the needs of their qualitative framework. The discretized data features measurements of 6 components in 6 different experimental setups. In each of the experiments one or two of the components of the network are inhibited and later the HGF stimuli is added. Additionally the authors provide a control measurement where no inhibition is present.

In the study, the data are present as fold-change comparisons between some of the experiments. For each component there are 9 time-points measured, however in the discretized form these are divided at the time of 30 minutes into an early and late response, as it is expected that around that time feedback effects start to play a role in the behaviour of the system. As the fold-change scheme is not suitable for encoding as a time-series, we reinterpreted the data into a measurement scheme, where the fold change

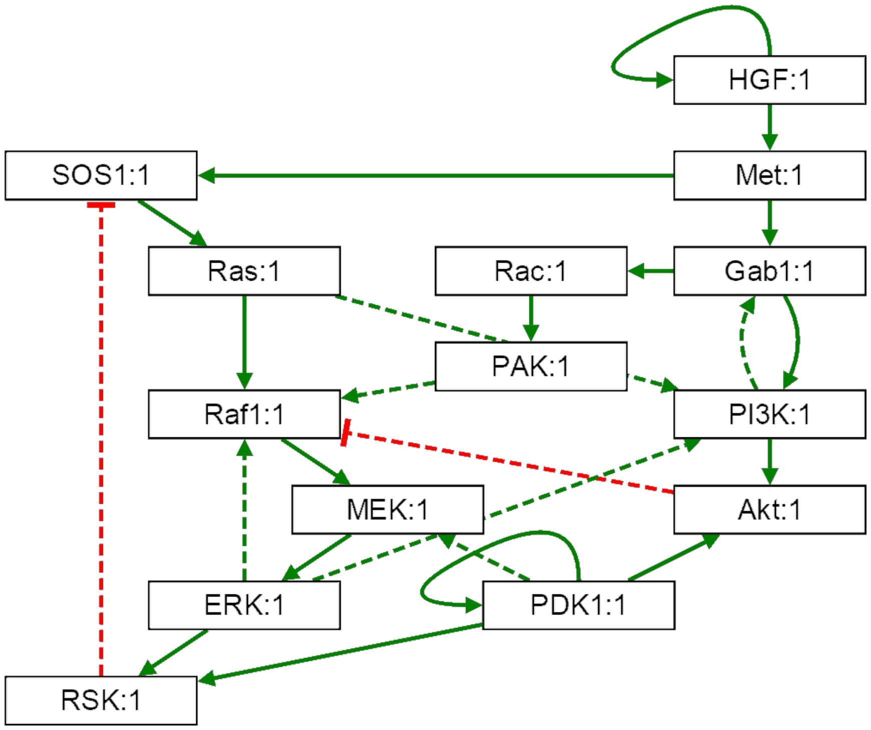


Figure 4.2: The structure of the model that was identified as optimal in [DSM+15]. The regulations denoted by a full line constitute the core network, whereas those that are dashed are added from the pool of optional regulations. Again the green edges represent activation and the red ones inhibition.

translates to a difference between two measurements. This means that from two fold-changes we obtain three measurements. The particular values for the measurements were determined in the following manner:

- In the experiment a Met inhibitor was used that blocks the receptor of the pathway and thereby downregulates all signalling processes even under stimulation. The fold-change comparison to the control shows a significant downregulation in all read-outs therefore we conclude that the control state has active read-outs.
- For other setups, if there is a significant decrease [DSM+15] in the fold change, the component is expected to be at the level 0 after the change.
- Likewise, if there is a significant increase, the component is expected to be at the level 1.
- If there is no significant change, no requirement is placed on the value.

- We require the full monotonicity in the behaviour, *i.e.* if a value of a component does not change between two timepoints, we require that it cannot change in the simulation either. If the value differs between two measurements, we require that there is exactly one change of that value. For details please refer to [STS15a].

Altogether we have obtained 5 properties, which are detailed in the supplement B.5. Even though this interpretation of the data is quite strict, we obtained that even the core network is capable of satisfying all the experiments for each of the possible parametrizations. We have therefore focused instead on the structure identified as optimal by the authors to see whether the addition of some of the optional edges can disrupt the expected function of the network. This optimal network is depicted in Figure 4.2. This is a slightly simplified version of the original, which features two components for RSK in an activation cascade. We joined this cascade into a single node, which is a preserving operation in our framework [SAR13].

4.2.2 Selection and Analysis

For the purposes of the analysis we have created three selections:

1. ALL is the set of all 223776 possible parametrizations.

2. $VALID$ is the set of 149184 parametrizations that satisfy all the measurement series.

3. $MINCOST$ is the set of 135072 parametrizations that have the minimal *cost* for all the measurement series.

As can be immediately seen, there exist parametrizations over the optional edges that render one or more of the measurement series not satisfiable.

Data of all the reports are in an interactive form available in the Appendix B.5. Here we provide some of the possible observations about the data. Firstly we analyse the comparison $VALID - ALL$. In the quantitative report we see that $K_{\text{MEK}}(\omega)$ with $\omega_{\text{PDK1}} = \{0\}, \omega_{\text{Raf}} = \{1\}$ is for the $VALID$ set bound to the value 1, meaning it is necessary that Raf1 alone can activate MEK. We also see that three out of the five measurement series are satisfied by all parametrizations, whereas the remaining two are satisfied exactly by those in the $VALID$ selection, meaning that both place the same requirement on the behaviour of the network. In the qualitative report we can see that the function $RF^K_{\text{MEK}} = \text{Raf1\&PDK1}$ is completely missing, in accordance with the quantitative observation, and the functions $RF^K_{\text{MEK}} = \text{Raf1}$ and $RF^K_{\text{MEK}} = \text{Raf1|PDK1}$ are now present with the same frequency, suggesting that the $(\text{PDK1}, 1, \text{MEK})$ regulation is superfluous and should probably be removed. In the regulation graph we then see that both the frequency and the impact of PDK1 on MEK decreases and lastly in the correlations report we see an increase in the bias of MEK.

Secondly we analyse the comparison $MINCOST - VALID$. From the quantitative report we see that there is a slight increase in bias of Raf1. In the qualitative report we can see that 14 regulatory functions for Raf1 disappear completely, however as there are still 134 remaining, this does not provide too much information. A much cleaner picture can be gained from the regulation graph where we can see a decrease in the impact and the frequency of ERK on Raf1 in favour of both inhibition by Akt and activation by PAK.

4.3 E. Coli Biofilm Production

In this section we present a study of bistability in the production of protective biofilm (composed of curli fibrae) in the bacteria E. Coli. The original study [YSS^{+}15] was conducted in cooperation with Kaveh Pouran Yousef, who designed Figure 4.3 and is the author of Section 4.3.1.

The purpose of the study was to examine the conditions under which the bacteria makes a decision whether to start producing biofilm or not. To examine the behaviour of the system deemed to be responsible for this decision, we have first constructed a multi-valued model and analysed its individual parametrizations. The knowledge obtained from this analysis was then forwarded to our colleagues who subsequently built a continuous and stochastic model, which are not presented in this thesis. This application demonstrates how TREMPPI can be used to complement other modelling frameworks, and charts some possible options for how to transfer the knowledge obtained from it.

All the data relating to the logical model are available in the attachment, see Appendix B.6.

In the following we use lower case first letter for a name of a gene and an upper case for its respective protein.

4.3.1 Curli Fimbriae Expression

By Kaveh Pouran Yousef [YSS^{+}15]

The second messenger molecule bis-(3'-5')-cyclic dimeric guanosine monophosphate (c-di-GMP) is the central component within the regulatory signalling network of curli fimbriae during the stationary phase of the bacterial population growth cycle and upon induction of various stress conditions. C-di-GMP positively contributes to the expression of CsgD, the key biofilm regulator, activating the expression of the curli regulon (csgBAC), with CsgB and CsgA proteins constituting the curli fiber [Hen09, WPP^{+}06]. As shown in Figure 4.3B, the synthesis and degradation of c-di-GMP within this net-

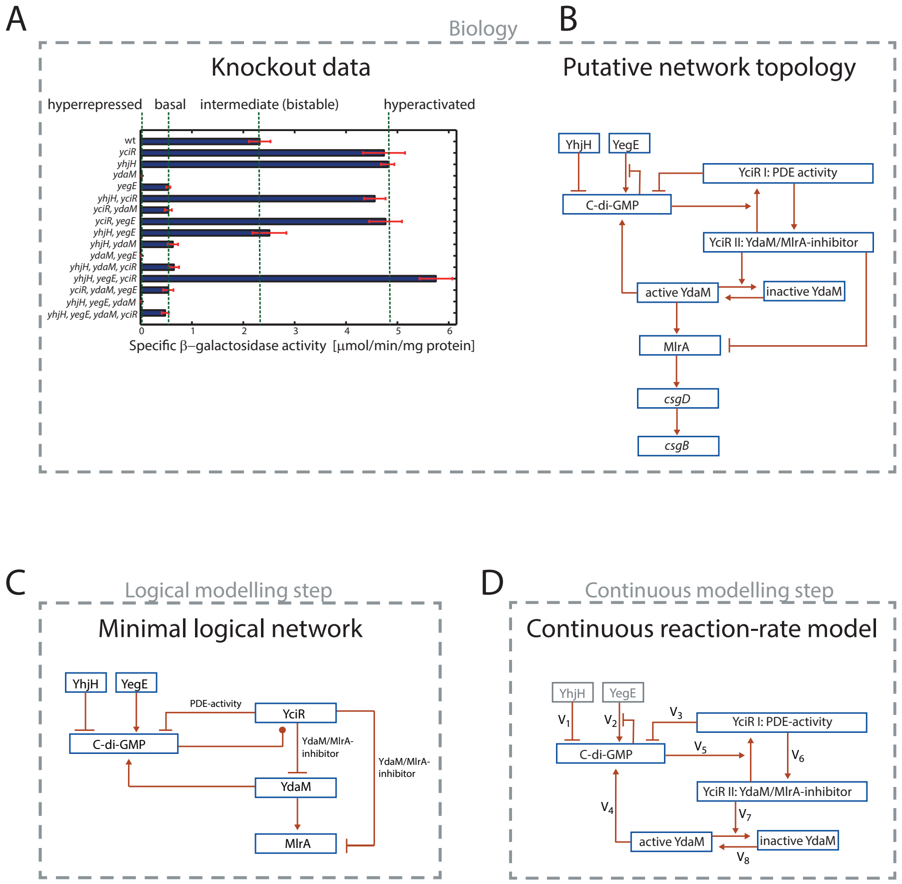


Figure 4.3: **Logical-continuous modelling pipeline for the bistability analysis of curli regulation in *E. coli*.**
By Kaveh Pouran Yousef [YSS+15]

(A) Expression of the curli gene *csgB* in mutants with single or multiple knockout mutations in YegE/YhjH and YdaM/YciR c-di-GMP control modules. Derivatives of *E. coli* K-12 W3110 carrying a single copy *csgB*::*lacZ* reporter fusion were grown in LB at 28°C for 24 hours and β-galactosidase activities were determined. Figure reproduced from [LKPH13]. **(B)** Signalling network regulating the expression of curli fibers, as suggested in [LKPH13]. **(C)** Logical formulation of the network topology. The two different functional states of YciR are modelled by two types of interactions, only the inhibitory effects exerted by YciR are represented by edges in the figure. Full details on the regulatory activity of YciR and the dependence of the two functional states on the activity of c-di-GMP is formalised in the SI section. **(D)** Continuous model incorporating kinetic reaction rates with the validated network topology.

work is maintained by two diguanylate cyclase (DGC)/phosphodiesterase (PDE) pairs: YegE/YhjH (module I) and YdaM/YciR (module II). While DGC-type enzymes synthesise c-di-GMP from two GTP substrates, PDE-type enzymes are responsible for the degradation of c-di-GMP. As shown in Figure 4.3B, besides the degradation function of c-di-GMP, YciR exhibits a second activity: the inhibition of YdaM and MlrA. Importantly, YdaM is the key activator of the transcription factor MlrA, which directly activates transcription at the csgD promoter region. A sufficiently high amount of c-di-GMP prevents YciR from inhibiting YdaM, since YciR binds c-di-GMP and starts to degrade it, acting as a trigger enzyme [LKPH13]. In turn, unbound (active) YdaM activates MlrA and induces the signalling cascade leading to the expression of the curli operon csgBAC.

Previously, the activity of the csgB promoter was measured in all possible single, double, triple and quadruple knockout strains of the set of genes yegE, yhjH, ydaM and yciR [LKPH13]. As depicted in Figure 4.3A, despite 15 different genetic backgrounds, only four main expression levels of the csgB gene were observed. E. coli strains containing a ydaM-knockout exhibit a completely absent expression of the curli gene csgB (hyperrepressed) or a very low (basal) level expression, as compared to wild type (Figure 4.3A). Furthermore, strains containing a yciR-knockout reveal an increased (hyperactivated) expression of csgB if ydaM is present. This knockout background is “blind” with respect to knockouts of module I genes (yegE and yhjH). Genetic and biochemical analyses allowed Lindenberg *et al.* [LKPH13] to derive a model of the regulatory network, which is depicted in Figure 4.3B. In this model different functional states are assigned to YciR (I/II) and YdaM (active/inactive).

In order to find evidence for the bistability of the curli signalling system we assigned each of the four different expression levels observed in the genetic knockout experiments at population level (Figure 4.3A) a certain single-cell phenotype. To this end, we used the complementary single-cell measurements of the wild type expression of the CsgB protein in Serra *et al.* [SRK$^+$13]. These single-cell data suggest that the intermediate expression level of the csgB gene at population level corresponds to a heterogeneous mix of curli-on and curli-off cells in the bacterial population. This indicates that the basal and the hyperrepressed level of csgB reflects a mixture of cellular phenotypes where the fraction of curli-on cells is significantly lower as compared to the wild type, or it becomes undetectable. In contrast, the hyperactivated phenotype may be due to two different types of single-cell expression levels. Either all cells in the population are in the curli-on mode, or there is a subset of cells in the curli-off mode but their frequency is significantly lower than the frequency of curli-off cells in the wild type population.

4.3.2 Derivation of the Logical Model

For deriving the model we applied a time-scale separation. Thus we assumed that the total levels of involved proteins do not significantly change in contrast to their activity states. We based our assumption on results indicating that protein transcription and translation are significantly slower than post-translational interaction dynamics [Alo06].

Having described the components by means of logical states, logical rules were added to characterise the principles of mutual regulation by the components. In particular, all activating or inhibitory effects indicated by experimental data were incorporated, resulting in the logical model depicted in Figure 4.3C. Most of the components were interpreted as Boolean. An exception was made for the protein YciR, which was assigned a knockout state (0-state) and two different observable activities that are mutually exclusive (PDE activity and YdaM/MlrA-inhibition activity) [LKPH13]. Note that YdaM has also two different activity levels. However, since one of them does not have any observable effect, we identified it with the 0-state.

In the model we require all the regulations to be observable. In the case of YciR we distinguished between alternative regulations depending on their corresponding effect. A regulation is observable only when YciR occurs in the respective state, including the positive and negative effects (Figure 4.3C). Most of the included regulations correspond to the experimentally derived network topology in Figure 4.3B, with a few exceptions. The first difference is given by the negative feedback induced by the allosteric product inhibition of c-di-GMP on the catalytic activity of YegE. When molecular numbers are considered, product inhibition gives rise to an upper limit on the synthesis rate and thus contributes to setting up a homeostatic steady-state level of c-di-GMP [CCP$^+$06]. However the semantics of this negative feedback loop do not transfer to the discrete logical model *i.e.* it is not possible for c-di-GMP to completely inhibit YegE. Since in the logical model, only one discrete value is used as an abstraction for the concentrations of c-di-GMP that are considered as the 1-state, this negative feedback inhibition has no observable logical effect. Therefore we eliminated the corresponding edge from the logical interaction graph (see Figure 4.3C).

Furthermore, the DGC/PDE-pair YhjH and YegE constitute the inputs of the network, which means that they maintain the values that they were initially set to. Finally, we used the system property that the expression of csgB is induced if its transcription factor CsgD is expressed. This is the case if MlrA is at the level 1, since it is in turn the functional activator of csgD transcription (Figure 4.3B). Therefore, in order to reduce the model, we removed csgD and used the Boolean component MlrA as the output component of the network. Thus, the 1-state of MlrA represents the induction of curli expression in the logical model.

At this point we incorporated a sufficient amount of information into the model for deriving all regulatory functions except the functions of c-di-GMP and MlrA. These two components are influenced by multiple regulators, giving rise to a set of alternative logical functions describing their effect on c-di-GMP or MlrA, full list of which is in the respective qualitative report. After resolving the constraints derived from the interaction effects, we identified 114 possible regulatory functions for c-di-GMP and 2 for MlrA. Finally, by generating all possible combinations of the functions of the two components, we obtained 228 alternative models for the whole network.

4.3.3 Formalisation of the Experimental Data

After specifying all 228 possible alternative logical models, we assessed for each model whether it fulfils all constraints given by the observations of the 15 genetic knockout experiments (depicted in Figure 4.3A, the 16$^{\text{th}}$ is without any knockout). We modelled a genetic knockout by forcing the respective component to remain in the 0-state for the whole course of the simulation. Our main focus was on stabilising behaviour, thus we assessed the reachability of stable states with a particular model configuration. A system state is considered as stable if it is not possible to leave it during simulation. As shown in Figure 4.3A, there are four distinctive phenotypes, each of which we addressed individually:

- The intermediate (bistable) phenotype was related to the ability to reach a stable state with MlrA = 1 and a stable state with MlrA = 0 in the logical modelling framework.
- In the hyperrepressed phenotype no curli is being expressed, therefore we required that the stable state with MlrA = 0 is reachable, while the MlrA = 1 state is not.
- The hyperactivated phenotype exhibits strong expression of curli and therefore we stated that the stable state with MlrA = 1 must be reachable. However, unlike the hyperrepressed phenotype, we did not prohibit the inactive stable state, since we could not rule out that this phenotype is generated by a mix cell that do produce the curli fimbrae and those that do not.
- Lastly, the basal phenotype exhibits only little, but still measurable expression of curli. However, very weak expression of curli was also observed in the hyperrepressed phenotype (see Figure 4.3A). It would therefore be possible to identify the basal with the hyperrepressed phenotype, but this constraint could be spurious. We have therefore decided not to include the basal phenotype in the main analysis.

Since the measurements were conducted in the early stationary phase of the growth cycle (see Figure 4.3A), we expected that YegE = 1 and YhjH = 1

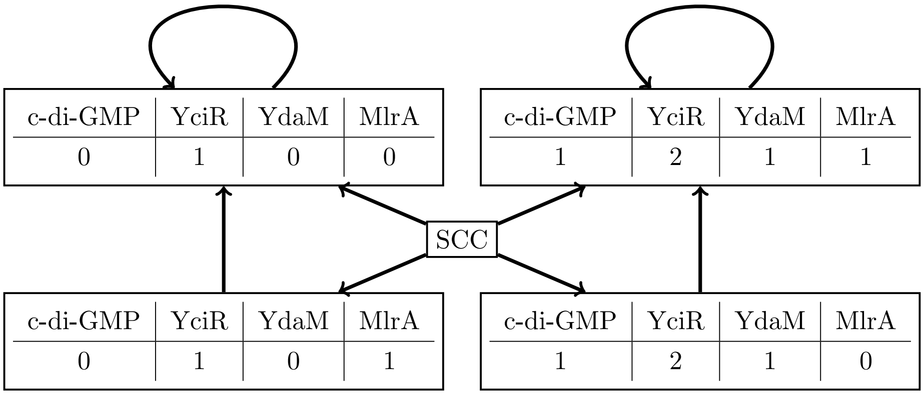

c-di-GMP | YciR | YdaM | MlrA
0 | 1 | 0 | 0
c-di-GMP | YciR | YdaM | MlrA
1 | 2 | 1 | 1
SCC
c-di-GMP | YciR | YdaM | MlrA
0 | 1 | 0 | 1
c-di-GMP | YciR | YdaM | MlrA
1 | 2 | 1 | 0

Figure 4.4: A reduced graph of behaviour of the 10 parametrizations that satisfy the *relaxed* set of properties. Each of these has a strongly connected component (SCC) allowing to oscillate indefinitelly. However once it leaves the SCC, it stabilizes on one of the two stable states, either with all the core compoenents active, or with all the core components inactive.

unless they are knocked out. Using the stability constraints resulting from experimental data (the knockout data), we formulated 15 properties, which are listed in the *relaxed* selection, as referred in Appendix B.6.

In addition, the observations from the knock-out experiments can also be interpreted in a less conservative sense *i.e.* interpreting the data such that no curli is expressed in both the basal- and hyperrepressed phenotype (the stable state MlrA $= 0$ is reachable, while the MlrA $= 1$ state is not) and that the hyperactivated phenotype corresponds to the scenario where all cells express curli (the stable state MlrA $= 1$ is reachable, but not the MlrA $= 0$ state). Such interpretation yields 17 additional properties, making it 32 in total. We call the selection that matches all the 32 properties *strict*.

4.3.4 Results of the MC Procedure

As a first analysis step, we tested the consistency of the candidate models with the 15 *relaxed* constraints. As a result, we found that 10 out of 228 possible models were in agreement with all constraints. These are represented by the *relaxed* selection in the attached data (Appendix B.6). Looking into the 10 models we see that for MlrA there are two possible regulatory functions, and 5 possible functions for c-di-GMP. The 10 valid models are then obtained as all combinations of these.

Since our focus is on stabilizing behaviour, we distinguish parametrizations according to the stable states that they generate. If the network is to stabilize in a state where YciR exhibits its inhibiting effect on YdaM then YdaM needs to be 0, since there are no other regulators influencing its activity. For the same reason, YciR switching to PDE activity must result in a change of activity level for YdaM, namely YdaM must become active. Consequently, in a stable state we either observe that YciR acts as an inhibitor and c-di-GMP = 0, or YciR exhibits its PDE activity and YdaM = 1. Since MlrA is only regulated by YdaM and YciR, we know that in the first scenario MlrA needs to be 0, since its inhibitor is active and its activator is inactive, and in the second, opposing scenario MlrA = 1. That is, in a stable state the values of YdaM and MlrA always coincide and we can use YdaM as a marker for the curli production.

In Figure 4.4 we illustrate the dynamical behaviour that is shared by all 10 models. As ensured by the constraints, all models support bistability given that both inputs (YegE and YhjH) are active. This means that the model behaviour is non-deterministic and can asymptotically end in either one of the two stable states distinguished by the presence of MlrA. Interestingly, for all models these stable states do not only coincide in that they represent *expression* or *no expression* of curli, but also in the activity of all other model components. That is, we can completely characterise the two possible equilibrium states when the input components are activating, *i.e.* at the level 1. Furthermore, an inspection of the regulatory functions of the 10 remaining models indicates that there are five possible regulation mechanisms for c-di-GMP, which exhibit noteworthy commonalities. The most prominent pattern is that c-di-GMP is in most cases adopting the value of YdaM, whereas the value of YciR is almost uncorrelated with the update of c-di-GMP. This observation allowed us to identify YdaM as the most influential regulator of c-di-GMP.

In the *strict* scenario, there is exactly one parameterization that satisfies all the 32 properties. Thus, irrespective of the elaborated interpretation of the experimental observations, there exists at least one logical model that is consistent with all the properties. Furthermore, note that only one function for MlrA is possible if we consider the constraint set relating to the *strict* interpretation of the knock-out experiments. This is due to the interpretation of the basal phenotype as a biological state with no curli production, which is also the case when YdaM and YciR are both knocked out. In particular, this means that the absence of an inhibitory effect of YciR is not sufficient to initiate curli production and therefore eliminates one of the two previously valid MlrA regulatory functions. Since the logical framework is too coarse-grained to distinguish between differences in data that are of rather quantitative nature (as might be the case when considering the basal and the hyperrepressed phenotype), the underlying assumptions and consequently the result of this analysis have to be carefully evaluated in the biological context.

4.3.5 Translating Parameter Constraints

Above we have stated that YdaM is the most prominent regulator of c-di-GMP. We show this statement formally and translate it into a constraint on the sampling space of an continuous model in the form of ordinary differential equations (ODE). The aim of this section is to serve as a demonstration of how the logical modelling can be coupled with other frameworks.

Let us recall the assumption that the pool of c-di-GMP is shared between YegE, YciR, YdaM, and YhjH. We assume that there are no competitive reactions between the above listed four proteins on any single molecule of c-di-GMP, meaning that c-di-GMP is active in the system if and only if the production/degradation ratio stabilizes in a state with a sufficiently high concentration of c-di-GMP. From the final parametrization set, we can see that the conditions for c-di-GMP = 1 are either that the activating effect exhibited by YdaM is already sufficient on its own , or only one other activating influence (YegE = 1, YhjH = 0 or YciR being at its MlrA/YdaM inhibitor activity) is needed. Conversely, if the activating influence of YdaM is absent, this cannot be compensated or only if one activating influence is combined with the absence of at least one inhibiting influence on c-di-GMP.

Under the standing assumption that the regulators of c-di-GMP do not regulate each other, we can qualitatively compare the pairwise effects of competitive regulators of c-di-GMP. Two observations appear:

1. YdaM is stronger than YciR: in all parametrizations we observe that when YdaM = 1, YciR is at its PDE activity, YhjH = 0, and YegE = 0, then c-di-GMP = 1.
2. YhjH is stronger than YegE: in all parametrizations we observe that when only YhjH = 1 and YegE = 1, and the remaining regulators of c-di-GMP are 0, then c-di-GMP = 0.

Such observations can now be exploited to derive constraints for the parameter sampling for the ODE model.

In the following, we will only focus on the observation 1). As a first step we translate the logical scenario for our observation concerning c-di-GMP regulation into the continuous setting. In the ODE model the pool of c-di-GMP is described by the equation:

$$\begin{aligned}\frac{d}{dt}x_1 = {} & \frac{V_{\max 1}}{1 + x_1/K_{\mathrm{i}}^{\mathrm{YegE}}} - \frac{V_{\max 2}x_1}{x_1 + K_{\mathrm{m}}^{\mathrm{YhjH}}} \\ & - (YciRtot - x_2)\frac{k_{YciRact}x_1}{x_1 + K_{\mathrm{m}}^{\mathrm{YciR}}} + \frac{k_{YdaMact}x_3^n}{(K_{\mathrm{d_{polymer}}}^{\mathrm{ydaM}})^n + x_3^n},\end{aligned}$$

where x_2 is the amount of YciR, x_3 is the amount of YdaM and the remaining values are either parameters or constants. For the whole model with details on the parameters and constants please refer to [YSS$^+$15].

Since in the observation 1) it is said that YegE = 0 and YhjH = 0, we can drop the first two terms in the ODE describing c-di-GMP behaviour, meaning that the above equation simplifies to:

$$\frac{d}{dt}x_1 = -(\mathit{YciRtot} - x_2)\frac{k_{\mathrm{YciRact}}x_1}{x_1 + K_{\mathrm{m}}^{\mathrm{YciR}}} + \frac{k_{\mathrm{YdaMact}}x_3^n}{(K_{\mathrm{d_{polymer}}}^{\mathrm{ydaM}})^n + x_3^n}.$$

Much of the difficulty in relating a logical to an ODE model comes from the necessity to put *real-valued* and *discretized* values into relation, implying that we need to decide which ranges of *real* values correspond to the logical states of YdaM = 1 and c-di-GMP = 1 and YciR being in its PDE configuration. A variety of discretization methods are available, but the results often need to be carefully evaluated w.r.t. the application, see *e.g.* [DLML10]. However, for our purposes we do not need to know the exact discretization thresholds, we just need to make sure that we evaluate the ODE at values for x_1, x_2 and x_3 that represent the logical values of c-di-GMP = 1, YciR being in its PDE configuration (interpreted as YciR not acting as YdaM inhibitor) and YdaM = 1. We achieve this by considering the limits $x_1 \to \infty, x_2 \to 0$, and $x_3 \to \infty$, which ensures that we crossed the respective discretization thresholds regardless of their actual value. Note that these limits can be interpreted as YciR being fully committed to PDE activity and YdaM and c-di-GMP fully saturating the corresponding rate functions. This results in the terms $x_1/(x_1 + K_{\mathrm{m}}^{\mathrm{YciR}})$ and $x_3^n/((K_{\mathrm{d_{polymer}}}^{\mathrm{ydaM}})^n + x_3^n)$ tending to the limit value 1. We obtain:

$$\begin{aligned}
\lim_{x_1 \to \infty} \frac{k_{\mathrm{YciRact}}x_1}{x_1 + K_{\mathrm{m}}^{\mathrm{YciR}}} &= k_{\mathrm{YciRact}}, \\
\lim_{x_2 \to 0} (YciRtot - x_2) &= YciRtot, \\
\lim_{x_3 \to \infty} \frac{k_{YdaMact}x_3^n}{(K_{\mathrm{d_{polymer}}}^{\mathrm{ydaM}})^n + x_3^n} &= k_{YdaMact},
\end{aligned}$$

which we apply to our simplified ODE, obtaining the limit form:

$$\frac{d}{dt}x_1 = -\mathit{YciRtot} \cdot k_{\mathrm{YciRact}} + k_{\mathrm{YdaMact}}.$$

The target parameter for c-di-GMP in the corresponding logical states indicates that c-di-GMP remains active in this scenario. We now make the assumption that we can translate this into a constraint saying that the value of x_1 should not decrease (represented by $\frac{d}{dt}x_1 \geq 0$) whenever the system is in a biological state corresponding to the state of our logical scenario. In particular, this holds for our limit considerations. In general, this assumption will not always be true, since it is feasible that for large values inhibiting effects come into play that are again counteracted when approaching the discretization threshold. However, the logical model is set up in such a way that all qualitative regulation effects that could lead to such a behaviour are captured and represented by the different activity levels, so that we exclude the possibility of the observations not holding asymptotically. This is also in agreement with the biological interpretation of those limits mentioned above. Inserting the condition for $\frac{d}{dt}x_1$ into the ODE then yields the inequality

$$0 \leq -\mathit{YciRtot} \cdot k_{\text{YciRact}} + k_{\text{YdaMact}},$$

giving us the constraint:

$$\mathit{YciRtot} \cdot k_{\text{YciRact}} \leq k_{\text{YdaMact}}.$$

To further ensure that our assumptions are proper, we relax this constraint based on the following observation. $YciRtot$ represents the total number of YciR molecules in the system. We know that YciR is present, therefore $YciRtot \geq 1$ and from there:

$$k_{\text{YciRact}} \leq \mathit{YciRtot} \cdot k_{\text{YciRact}} \leq k_{\text{YdaMact}}.$$

Since the value for $YciRtot$ is expected to be much larger than 1, we can utilize the much weaker constraint:

$$k_{\text{YciRact}} \leq k_{\text{YdaMact}}$$

with high confidence.

In the original study, this assumption is then translated into a constraint for the sampling of the parameters for the ODE equations, which however no longer relates to the topic of this thesis. From the logical perspective, we have contributed to this study in three main points:

- Confirming that the network satisfies the experimental data (model validation).
- Identification of the optimal logical parameters for the network.
- Derivation of regulatory semantics from the resulting model pool for the purposes of quantitative modelling.

The network in this study is quite small, so the performance improvements from adding sample constraints to the ODE model are not so significant. In general we however believe that the fast, abstract logical modelling can be very well coupled with the more precise but slower quantitative methods, as illustrated also for example in the article pair [AJOK09, OAJK10].

CHAPTER 5

Algorithms and Proofs

In this section we provide detailed description of the algorithms and mathematical constructs that constitute the methodology of this contribution and that are utilized in TREMPPI. The contents of this chapter foster a deeper understanding of the methods presented in Chapter 3, however it is not necessary for their comprehension. The two chapters are ordered in the same fashion.

All the computationl methods are evaluated in terms of their computational complexity. As TREMPPI in general produces large quantities of data, we discuss not only the usual TIME and SPACE complexity [Sip96], but also take into account the size of the ouput. In computer terms, we discern the requirements for secondary and tertiary storage [PH13]. For an algorithm *Algo* we denote:

- TIME($Algo(n)$) the usual time complexity of *Algo* on the input n,
- SPACE($Algo(n)$) the computational memory complexity of *Algo*,
- $[\![Algo(n)]\!]$ the amount of space taken by the output of *Algo*.

All the algorithms have their names in italics with the first letter being upper-case.

For each of the statements we provide the relevant proofs. However, only the purely mathematical statements are proven in a rigorous way. For the computational methods, which have been implemented and tested in practice, we provide only a structure or a summary for how a rigorous proof would be done.

5.1 Constructing a Parametrization Space

In this section we describe the methods that are involved in the initial step of the framework—enumeration of the feasible parametrizations. To describe the complexity of the algorithms for constructing the parametrization space we first need to establish bounds on its size in general. To this end we establish several variables denoting the boundaries of size for a graph G as:

- $size(\rho^G) = \max\{\rho^G(v) \mid v \in V^G\}+1$—the maximal number of activity levels of any component.
- $size(\Omega^G) = (size(\rho^G))^{[\![V^G]\!]}$—the maximal number of regulatory contexts of any component and therefore also for any partial parametrization. Note that this term stems form the fact that there are at most $size(\rho)$ regulatory intervals for each of the up to $[\![V]\!]$ regulators. This is also the maximal size of a partial parametrization.
- $size(K) = size(\Omega^{[K]}) \cdot [\![V^{[K]}]\!]$—the maximal size of a single parametrization.
- $size(\mathcal{K}^G) = size(\rho^G)^{size(K)}$—the maximal size of the parametrization space in the number of different parametrizations.

We then use the enumeration algorithm $Enumerate(G) \in \mathcal{K}^G$. If no constraints are used, the time complexity is equal to the time it takes to write out all the parametrizations, i.e.:

$$\text{TIME}(Enumerate(G)) = [\![Enumerate(G)]\!] \in \mathcal{O}(size(\mathcal{K}^G) \cdot size(K))$$

as we simply enumerate and write all the possible values. For evaluation of a single parametrization we need only $\text{SPACE}(Enumerate(G)) \in \mathcal{O}(size(K))$.

5.1.1 Computing Regulation Constraints

To evaluate the regulation constraints over a parametrization, we must compare for each regulation the parameters for the regulatory context that differ in the presence of the said regulation. To improve performance, we do not evaluate the full parametrizations, but only the partial ones. We then construct the full parametrization space as the product of those partial parametrizations that satisfy the regulation constraints for each of the components.

For a single partial parametrization the evaluation of one regulation constraint requires $size(\Omega^G)$ time (compare all the values). There are only two such constraints for every regulation, which is a constant factor, and there are at most $[\![V^G]\!]$ (all other components) times $size(\rho^G)$ (all possible thresholds) regulations for a single component. In summary, evaluation of a single partial parametrization is bounded by $\mathcal{O}(size(\rho^G) \cdot [\![V^G]\!] \cdot size(\Omega^G))$.

The size of a partial parametrization space (the maximal number of possible partial parametrizations) is equal to $size(\rho^G)^{size(\Omega^G)}$ for each of the $[\![V^G]\!]$ components. We now show that the time complexity of evaluating all the constraints belongs to the same class as the unconstrained enumeration, which was bound by $size(\mathcal{K}^G)$. This subsequently means that adding the constraints does not add on complexity of the procedure.

Lemma 5.1.1. *It holds that for any network G s.t. $\llbracket V^G \rrbracket > 4$ we have that $size(\mathcal{K}^G) \geq (size(\rho^G)^{size(\Omega^G)} \cdot \llbracket V^G \rrbracket) \cdot (size(\rho^G) \cdot \llbracket V^G \rrbracket \cdot size(\Omega^G))$.*

Proof. We do so by giving the target inequality and replacing each side by equal or stronger inequality until we reach a set of terms whose validity is apparent. We start by expanding the left side and then considering the logarithm of both:

$$\begin{aligned}
size(\rho^G)^{size(\Omega^G) \cdot \llbracket V^G \rrbracket} &\geq size(\rho^G)^{size(\Omega^G)+1} \cdot \llbracket V^G \rrbracket^2 \cdot size(\Omega^G) \\
\log size(\rho^G) \cdot size(\Omega^G) \cdot \llbracket V^G \rrbracket &\geq \log size(\rho^G) \cdot (size(\Omega^G) + 1) \\
&\quad + 2 \cdot \log \llbracket V^G \rrbracket \\
&\quad + \log size(\Omega^G)
\end{aligned}$$

This inequality can be split into three sub-inequality s.t. if each of them holds, then the inequality also holds:

$$\begin{aligned}
&\frac{1}{3} \cdot \log size(\rho^G) \cdot size(\Omega^G) \cdot \llbracket V^G \rrbracket \geq \log size(\rho^G) \cdot (size(\Omega^G) + 1) \\
&\frac{1}{3} \cdot \log size(\rho^G) \cdot size(\Omega^G) \cdot \llbracket V^G \rrbracket \geq 2 \cdot \log \llbracket V^G \rrbracket \\
&\frac{1}{3} \cdot \log size(\rho^G) \cdot size(\Omega^G) \cdot \llbracket V^G \rrbracket \geq \log size(\Omega^G)
\end{aligned}$$

We now simplify each sub-inequality. Note that each *size* term is a non-zero integer, and that $size(\rho^G) \geq 2$ which implies that $\log size(\rho^G) \geq 1$ and $size(\rho^G)^{\llbracket V^G \rrbracket} \geq 2^{\llbracket V^G \rrbracket}$. We start with the first sub-inequality:

$$\begin{aligned}
\frac{1}{3} \cdot \log size(\rho^G) \cdot size(\Omega^G) \cdot \llbracket V^G \rrbracket &\geq \log size(\rho^G) \cdot (size(\Omega^G) + 1) \\
\frac{1}{3} \cdot size(\Omega^G) \cdot \llbracket V^G \rrbracket &\geq (size(\Omega^G) + 1) \\
size(\Omega^G) \cdot \llbracket V^G \rrbracket &\geq 3 \cdot size(\Omega^G) + 3 \\
size(\Omega^G) \cdot (\llbracket V^G \rrbracket - 1) + 2^{\llbracket V^G \rrbracket} &\geq size(\Omega^G) \cdot 3 + 3
\end{aligned}$$

which apparently holds for $\llbracket V^G \rrbracket \geq 4$, as given in the assumptions. Also for the second sub-inequality we get that:

$$\begin{aligned}
\frac{1}{3} \cdot \log size(\rho^G) \cdot size(\Omega^G) \cdot \llbracket V^G \rrbracket &\geq 2 \cdot \log \llbracket V^G \rrbracket \\
\log size(\rho^G) \cdot size(\Omega^G) \cdot \llbracket V^G \rrbracket &\geq 6 \cdot \log \llbracket V^G \rrbracket \\
\log size(\rho^G) \cdot 2^{\llbracket V^G \rrbracket} \cdot \llbracket V^G \rrbracket &\geq 6 \cdot \log \llbracket V^G \rrbracket
\end{aligned}$$

which is again apparent, since for $\llbracket V^G \rrbracket \geq 4$ it holds that $2^{\llbracket V^G \rrbracket} \geq 6$ and that $\llbracket V^G \rrbracket \geq \log \llbracket V^G \rrbracket$. The last inequality:

$$\frac{1}{3} \cdot \log size(\rho^G) \cdot size(\Omega^G) \cdot \llbracket V^G \rrbracket \geq \log size(\Omega^G)$$

is quite clear directly. □

As each partial parametrization is evaluated individually, no additional space is necessary. The size of the result is understandably at most as big as the unconstrained space.

However, rather than evaluating the individual constraints, we are using a constraint solver Gecode [STL10], which usually provides for much better practical performance.

5.1.2 Computing Direct Constraints

Using direct constraints means all-round performance improvement by reducing the size of the space of partial parametrizations. Let k be the number of parameters that are set to a fixed value by some direct constraints, then:

$$size(\mathcal{K}^G) = size(\rho^G)^{size(K)-k},$$

meaning that both the computation time and the output size are reduced by some exponential factor.

5.1.3 Computing Normalization Constraints

Like the regulations constraints, the normalization constraints can be evaluated on the space of the partial parametrizations. As we show in Theorem 5.3.9, evaluation of the normalization constraints on a single partial parametrization is bounded by $\mathcal{O}(size(\Omega^G))$, which is strictly smaller than the complexity of the regulation constraint, therefore the overall complexity again remains unchanged.

The recipe for normalization is given in Algorithm 1, and the following two sections iteratively build the mathematical theory necessary for deriving the algorithm.

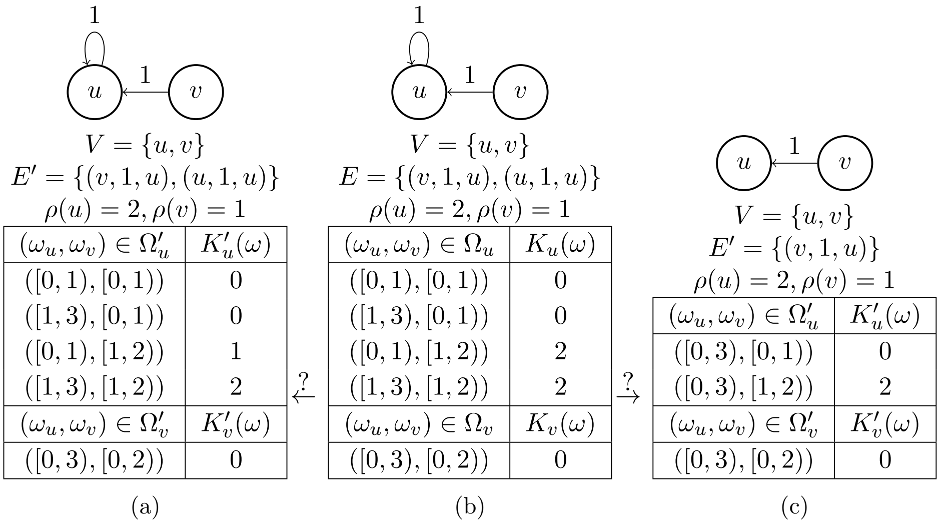
5.2 Conservative Graph Manipulations

This section will provide an introduction into the mathematical nature of dynamical behaviour and its conservation in the multi-valued networks.

5.2.1 Canonical Parametrizations

First, we focus on the notion of *dynamical equivalence*, *i.e.* the case where different parametrizations of a single graph generate the same TS. From (2.3) it is clear that two functions $F^K \neq F^{K'}$ will lead to distinct TSs, while coinciding functions $F^K, F^{K'}$ lead to the same dynamics. We therefore focus on describing the situations where for $K \neq K'$ we still have $F^K = F^{K'}$.

Consider the simple example in Figure 5.1. We see that having the parameter value $K'_v([0,1),[1,2)) = 1$ instead $K_v([0,1),[1,2)) = 2$ still yields $F_v^{K'}(0,1) = F_v^K(0,1) = 1$ and therefore the TS remains the same. This



(a)

$V = \{u, v\}$
$E' = \{(v, 1, u), (u, 1, u)\}$
$\rho(u) = 2, \rho(v) = 1$

$(\omega_u, \omega_v) \in \Omega'_u$	$K'_u(\omega)$
$([0,1),[0,1))$	0
$([1,3),[0,1))$	0
$([0,1),[1,2))$	1
$([1,3),[1,2))$	2
$(\omega_u, \omega_v) \in \Omega'_v$	$K'_v(\omega)$
$([0,3),[0,2))$	0

(b)

$V = \{u, v\}$
$E = \{(v, 1, u), (u, 1, u)\}$
$\rho(u) = 2, \rho(v) = 1$

$(\omega_u, \omega_v) \in \Omega_u$	$K_u(\omega)$
$([0,1),[0,1))$	0
$([1,3),[0,1))$	0
$([0,1),[1,2))$	2
$([1,3),[1,2))$	2
$(\omega_u, \omega_v) \in \Omega_v$	$K_v(\omega)$
$([0,3),[0,2))$	0

(c)

$V = \{u, v\}$
$E' = \{(v, 1, u)\}$
$\rho(u) = 2, \rho(v) = 1$

$(\omega_u, \omega_v) \in \Omega'_u$	$K'_u(\omega)$
$([0,3),[0,1))$	0
$([0,3),[1,2))$	2
$(\omega_u, \omega_v) \in \Omega'_v$	$K'_v(\omega)$
$([0,3),[0,2))$	0

Figure 5.1: An example of an uncertain reduction. The network **(b)** is in a non-canonical form. The *canonization* **(a)** however makes the edge $(u, 1, u)$ observable, even though it is superfluous as illustrated by **(c)**.

illustrates that, other than in the Boolean case, information on parameter values may get lost when deriving the update function. In Figure 5.1b we have a case where the parameter value lies two steps outside its context and by incremental change we leave the context even before the value can be attained.

We now define the notion of a *canonical* parametrization that prohibits such effects. Observe that a value change in v can cause the change of context only if v regulates itself. Therefore we say that $K \in \mathcal{K}^G$ is *canonical* if and only if

$$\forall v \in V, \forall \omega \in \Omega_v^G, \omega_v = [j, k) : (K_v(\omega) \geq j - 1) \wedge (K_v(\omega) \leq k). \qquad (5.1)$$

We also denote $\mathcal{C}^G \subseteq \mathcal{K}^G$ the subset of canonical parametrizations in $\mathcal{K}^G$.

We can obtain a clear correspondence between K and F^K if all the contexts contain just a single state, so that no ambiguities are introduced in (2.2). This partition is achieved when only considering *complete* graphs. For clarity we add that $(V, E, \rho) \in \mathcal{G}$ is complete if and only if for all $u, v \in V$ and every $n \in [1, \rho(u)]$ the edge (u, n, v) is in E. This gives us the following theorem:

Theorem 5.2.1. *For each $G = (V, E, \rho) \in \mathcal{G}$ and $S^{[K]} = \prod_{v \in V}[0, \rho(v)]$ it holds that if G is complete then (2.3) defines a bijection between $\mathcal{C}^G$ and $\mathcal{T}^G$.*

Proof. Let $T^K = (S^{[K]}, \to^T)$ for some complete $G = (V, E, \rho) \in \mathcal{G}$ such that $K \in \mathcal{C}^G$. The one-to-one correspondence between F^K and $(S^{[K]}, \to^T)$

immediately follows from (2.3). We therefore need to show that there is also such a correspondence between K and F^K.

First, it is important to note that if the graph is complete, each regulatory context depicts only a single configuration. This is because if G is complete then by (2.1) we have:

$$\forall v \in V : \Omega_v^G = \prod_{u \in V} \{[0,1),[1,2),\dots,[\rho(u),\rho(u)+1)\}.$$

Since each context is a singleton, each component has its value fixed. In that case, canonicity requires that the parameter in the context differs from the value only by 1. Therefore we have only three options for the parameter value. More precisely, by substituting (5.1) we have

$$\forall v \in V, \forall s \in S \,:\, s_v - 1 \leq K_v(\{s\}) \leq s_v + 1.$$

Then in such a case, (2.2) can be written as

$$F_v^K(s) = \begin{cases} s_v + 1 & \text{if } s_v + 1 = K_v(\{s\}), \\ s_v & \text{if } s_v = K_v(\{s\}), \\ s_v - 1 & \text{if } s_v - 1 = K_v(\{s\}), \end{cases}$$

from which we immediately get $F_v^K(s) = K_v(\{s\})$. Thus (2.3) corresponds to:

$$\forall v \in V, \forall s \in S^{[K]} : (K_v(\{s\}) = n) \wedge (K_v(\{s\}) \neq s_v) \iff s \to s_{v \leftarrow n}.$$

□

Based on this theorem, we can consider a complete graph with canonical parametrization as a representative of a class of models with the same behaviour. Now we show that it is possible to convert any graph with some parametrization into a complete graph with canonical parametrization, while keeping the dynamics unchanged.

First, we focus on the *canonization* function $Can : \mathcal{K}^G \to \mathcal{C}^G$. For each component $v \in V$ and for each regulatory context $\omega \in \Omega_v^G$ with $\omega_v = [j,k)$ we construct C as follows:

$$C_v(\omega) = \begin{cases} j - 1 & \text{if } K_v(\omega) < j - 1, \\ k & \text{if } K_v(\omega) > k, \\ K_v(\omega) & \text{otherwise.} \end{cases}$$

The goal is to avoid that the parameter value cannot be reached in one transition from any state in the context. The procedure is illustrated in the conversion from the parametrization K in Figure 5.1b to the parametrization K' in Figure 5.1a. We now prove that this procedure indeed yields a canonical parametrization for any RN that shares the TS with the original one.

Lemma 5.2.2. *Canonization is correct. For all $G \in \mathcal{G}$ and all $K \in \mathcal{K}^G$ it holds that $Can(K) = C$ is canonical.*

Proof. There are two options for K not to be canonical. The first option is that

$$\exists v \in V, \exists \omega \in \Omega_v^G, \omega_v = [j, k) : K_v(\omega) < j - 1$$

but then $C_v(\omega) = j - 1$, so C is canonical. The second case

$$\exists v \in V, \exists \omega \in \Omega_v^G, \omega_v = [j, k) : K_v(\omega) > k$$

can be treated analogously.

□

Lemma 5.2.3. *Canonization is conservative. For all $G \in \mathcal{G}$ and all $K \in \mathcal{K}^G$ it holds that if $Can(K) = C$ then $T^K = T^C$.*

Proof. Recall that the TSs T^K and T^C are fully defined by F^K and F^C, respectively. We therefore need to show that $F^K = F^C$.

For all $v \in V$ and for all $\omega \in \Omega_v^G$ the value $C_v(\omega)$ is set based on one of the three following cases.

First consider the case that $K_v(\omega) < \omega_v$. Denote $\omega_v = [j, k)$. For all $s \in \omega$ it holds that $s_v > j - 1$ and then $C_v(\omega) = j - 1$. This means that for all $s \in \omega$ both $K_v(\omega)$ and $C_v(\omega)$ are smaller than s_v and therefore for each $s \in \omega$ we have $F_v^K(s) = s_v - 1 = F_v^C(s)$.

The case that $K_v(\omega_v) > \omega_v$ can be treated analogously.

The third case is that we have $K_v(\omega_v) = C_v(\omega_v)$ and thus by definition $F_v^K(s) = F_v^C(s)$ for any $s \in \omega_v$.

□

Corollary 5.2.4. *For any K holds: If $j = K_v(\omega) < \omega_v$ (resp. $j = K_v(\omega) > \omega_v$) then replacing j with j', $0 \leq j' < \omega_v$ (resp. $\omega_v < j' \leq \rho(v)$) is conservative.*

5.2.2 Completion

To compare two networks, we want them to be isomorphic, which we achieve by making both into complete graphs. To this end we extend the structure of a graph using the *completion* function $Comp : \mathcal{K}^G \to \mathcal{K}^{\hat{G}}$. If $[K]$ is complete, then we map K to itself. For an incomplete $[K] = (V, E, \rho)$ we consider the non-empty set of missing edges $\hat{E} = \{(u, n, v) \mid u, v \in V, n \in [1, \rho(u)], (u, n, v) \notin E\}$. Assume that the set of all possible edges has some ordering. We extend a K to $\hat{K}$ such that $\hat{G} = [\hat{K}] = (V, E \cup \{min(\hat{E})\}, \rho)$. As an example you can see that the RNs in Figure 5.2a, 5.2c differ from each other only by the presence of a non-observable edge. Likewise for the GRs in Figure 5.1b, 5.1c.

To obtain the new parametrization $\hat{K}$ we first observe that $[\hat{K}]$ gives rise to new contexts that were obtained by partitioning some context of the component regulated by $\{min(\hat{E})\}$ into two. To preserve the dynamical behaviour we simply assign the parameter value of the original context to both resulting new contexts. Have $(u, n, v) = min(\hat{E})$. For each $v' \in V$ and for each $\hat{\omega} \in \Omega_v^{\hat{G}}$ we then create $\hat{K}$ as:

$$\hat{K}_v(\hat{\omega}) = \begin{cases} K_{v'}(\hat{\omega}) & \text{if } v' \neq v \vee (\hat{\omega} = [j,k) \wedge (j \neq n_- \vee k \neq n_+)), \\ K_{v'}(\hat{\omega}_{u \leftarrow [n_-, n_+)}) & \text{otherwise.} \end{cases}$$

We now prove that for an incomplete RN we can use the completion procedure to add a new edge while retaining the dynamics.

Lemma 5.2.5. *Completion is sound. For all $G \in \mathcal{G}$ and all $K \in \mathcal{K}^G$ it holds that if Comp$(K) = \hat{K}$ then there is $\hat{G} \in \mathcal{G}$ s.t. $\hat{K} \in \mathcal{K}^{\hat{G}}$.*

Proof. We have $u, v \in V$ and $n \in \rho(u)$, therefore by definition of $\mathcal{G}$ we have that if $(V, E, \rho) \in \mathcal{G}$ then $(V, E \cup \{(u, n, v)\}, \rho) \in \mathcal{G}$.

The variables n_-, n_+ exist since 0 is always a possible choice for n_- and $\rho(u) + 1$ for n_+. For any $m \in \theta(u, v)$ we know that $(0 < m < \rho(u) + 1)$.

From (2.1) we know that the only change occurs in the interval I_v^u. We therefore only need to show that $\hat{K}_v$ is extended to the affected contexts. Since n_-, n_+ exist, we have that $\omega_u \in \{i_1, \dots, [n_-, n_+), \dots, i_k\}$ for any $\omega \in \Omega_v^G$. Thus for any $\hat{\omega} \in \Omega_v^{\hat{G}}$ we also have $\hat{\omega}_{u \leftarrow [n_-, n_+)} \in \Omega_v^G$. Therefore $\hat{K}_v$ is defined on the whole $\Omega_v^{\hat{G}}$ for each $v \in V$.

□

Lemma 5.2.6. *Completion is conservative. For each $G \in \mathcal{G}$ and for each $K \in \mathcal{K}^G$ it holds that if Comp$(K) = \hat{K}$ then $T^K = T^{\hat{K}}$.*

Proof. We have that K differs from $\hat{K}$ only in a context $\hat{\omega} \in \Omega_v^{\hat{G}}$ with $\hat{\omega} = [j, k)$ where either $j = n_-$ or $k = n_+$.

Assume there is some $s \in \hat{\omega}$ for which $F_v^K(s) \neq F_v^{\hat{K}}(s)$. But we know that $[j, k) \subset [n_-, n_+)$ and therefore $s \in \hat{\omega}_{u \leftarrow [n_-, n_+)}$. This would however imply that also $\hat{K}_v(\hat{\omega}_v) \neq K_v(\hat{\omega}_{u \leftarrow [n_-, n_+)})$, which contradicts the definition of $\hat{K}$. We therefore have that $F^K(s) = F^{\hat{K}}(s)$.

□

Since the completion procedure adds only one edge at a time, we need to repeat the procedure. This is captured in the following Lemma.

Lemma 5.2.7. *For $G \in \mathcal{G}$, $K \in \mathcal{K}^G$, there is $n \in \mathbb{N}_1$ s.t. Comp$^n(K) =$ Comp$^{n+1}(K)$ and $[$Comp$^n(K)]$ is complete.*

Proof. The set $\hat{E}$ of missing edges in G is finite as V is finite. For each $v \in V$ also $[1, \rho(v)]$ is finite. In each iterative application of *Comp* the size of $\hat{E}$ is decremented by one. The recursive sequence becomes constant when $\hat{E}$ is empty, indicating a fixpoint $\mathit{Comp}^n(K)$ of *Comp*. By definition, $\mathit{Comp}^n(K)$ is complete.

□

5.2.3 Equivalence of Complete Networks

Combining all the statements above, we arrive at our final theorem:

Theorem 5.2.8. *Let $G, G' \in \mathcal{G}$, $K \in \mathcal{K}^G$, $K' \in \mathcal{K}_{G'}$ and denote Comp^* the fixpoint of Comp. Then $T^K = T^{K'}$ iff $\mathit{Can}(\mathit{Comp}^*(K)) = \mathit{Can}(\mathit{Comp}^*(K'))$.*

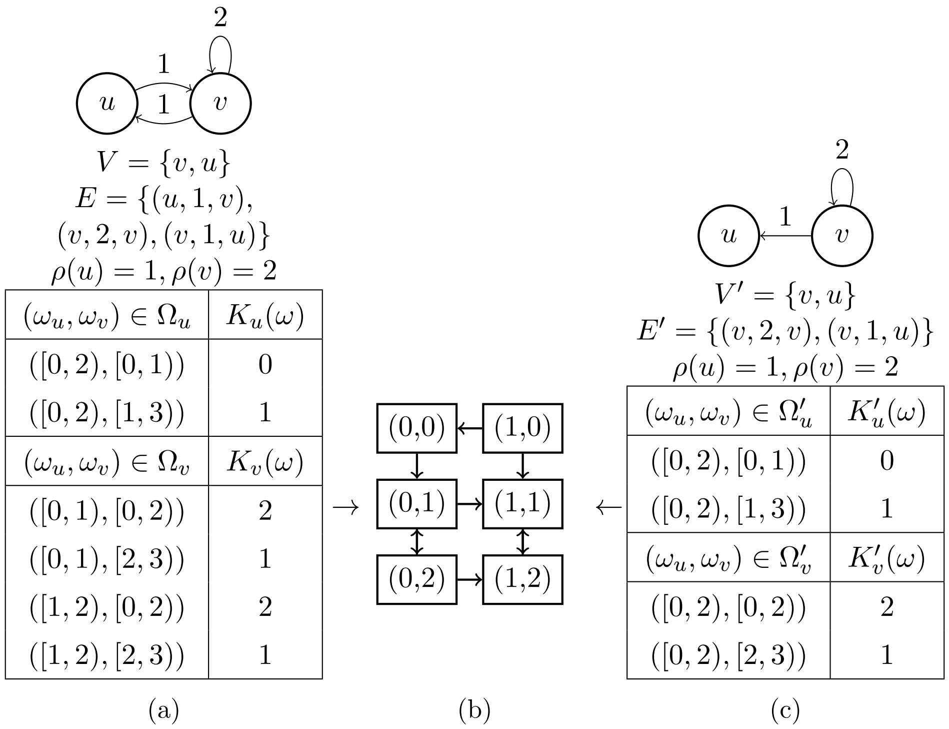
Proof. We now know that $\mathit{Can}(\mathit{Comp}^*(K))$ and $\mathit{Can}(\mathit{Comp}^*(K'))$ are canonical and complete. The equivalence follows from the bijection in Theorem 5.2.1.

□

5.3 Network Minimization

Canonization and completion provide mathematical insights into the problem of dynamical equivalence. However in application it is more useful to have a minimal, rather than maximal structure. Also, while the canonization provides a good intuition about what the actual behaviour for each context is, it can have side-effects like converting a non-observable edge to observable, as is illustrated in Figure 5.1. We therefore introduce another form of parametrization, named *normalized* parametrization, which prevents such effects but is more involved. Using a *normalization* procedure we then obtain a parametrization which is amenable to minimization. This section is divided into five consecutive steps:

1. We introduce a notion of *observability* in the TS which allows us to see whether an edge is observable based on the transitions in the TS.
2. We introduce a notion of a *monotone target value* (MTV) of a component. This value keeps the observability properties of TS, but is shared for a whole context.
3. We show how to compute the MTV from a parametrization and consequently how to compute a *normalized* parametrization.
4. We show that every edge that is not observable in the TS is not observable in the respective normalized parametrization.
5. We introduce the minimization function for RNs based on normalized parametrization and explain how to test equivalence via minimization.



$V = \{v, u\}$
$E = \{(u, 1, v),$
$(v, 2, v), (v, 1, u)\}$
$\rho(u) = 1, \rho(v) = 2$

$(\omega_u, \omega_v) \in \Omega_u$	$K_u(\omega)$
$([0, 2), [0, 1))$	0
$([0, 2), [1, 3))$	1

$(\omega_u, \omega_v) \in \Omega_v$	$K_v(\omega)$
$([0, 1), [0, 2))$	2
$([0, 1), [2, 3))$	1
$([1, 2), [0, 2))$	2
$([1, 2), [2, 3))$	1

(a)

(b)

$V' = \{v, u\}$
$E' = \{(v, 2, v), (v, 1, u)\}$
$\rho(u) = 1, \rho(v) = 2$

$(\omega_u, \omega_v) \in \Omega'_u$	$K'_u(\omega)$
$([0, 2), [0, 1))$	0
$([0, 2), [1, 3))$	1

$(\omega_u, \omega_v) \in \Omega'_v$	$K'_v(\omega)$
$([0, 2), [0, 2))$	2
$([0, 2), [2, 3))$	1

(c)

Figure 5.2: Reductions on the running example network. **(a)** The network from Figure 2.1 and a reducible parametrization $K \in \mathcal{K}^G$. **(c)** The reduced toy network G' with the parametrization K'. Note that for any $x \in [0, \rho(u)]$ and $y \in [0, \rho(v)]$ it holds that $F_v^K(x, y) = F_v^{K'}(x, y)$ and $F_u^K(x, y) = F_u^{K'}(x, y)$. Therefore we have **(b)** the transition system $T^K = T^{K'}$.

5.3.1 Observability in Transition Systems

We have already defined the notion of observability in the parametrization. However we are mostly interested in the observability since it has implications on the dynamics. We now show how it can be evaluated in the TS.

Intuitively, for an edge to be not observable there must be only a single value towards which the component evolves, no matter whether the regulator is above the thresholds of the said edge or below it. Formally for the RN $G = (V, E, \rho)$ the edge $(u, n, v) \in E$ is not observable in $(S, \to) = T^K$ *iff*:

$$\begin{aligned} &\forall s \in S^G, s_u \in [n_-, n_+), \exists k \in [0, \rho(v)], \forall j \in [n_-, n_+) : \\ &\delta(F_v^K)(s_{u \leftarrow j}) = sgn(k - (s_{u \leftarrow j})_v)\,, \end{aligned} \tag{5.2}$$

where $sgn : \mathbb{Z} \to \{-1, 0, +1\}$ is the usual signum function and $\delta(F_v^K) : S^{[K]} \to \{-1, 0, +1\}$ is the partial derivative in $s \in S$, in the dimension v, i.e.:

$$\delta(F_v^K)(s) = F_v^K(s) - s_v.$$

Consider the example in Figure 5.2. It is easy to see that the regulation $(u, 1, v)$ does not have any effect, since the left and right half of the TS are identical. Take in particular the example of the state $(u, v) = (0, 0)$. We choose $k = 2$. Then for $j = 1$ it holds that $\delta(F_v^K)(0, 1) = +1 = sgn(2 - 0)$ and for $j = 0$ it holds that $\delta(F_v^K)(0, 0) = +1 = sgn(2 - 0)$.

From the definition of the function F (2.2) and its derivative we easily see that:

$$\forall s \in S^G, s_v \in \omega \in \Omega_v^G : \delta(F_v^K)(s) = sgn(K_v(\omega) - s_v). \tag{5.3}$$

This illustrates that the TS non-observability is the kind that we are interested in, since it relies on the actual dynamics of the network as captured in F^K. It is a stronger notion than the corresponding parametrization based one, since observability in the TS implies observability in the parametrization. We show the contraposition of this statement in the following lemma:

Lemma 5.3.1. *Have $(u, n, v) \in E$ not observable in $K \in \mathcal{K}^G$. Then (u, n, v) is not observable in T^K.*

Proof. By definition of observability (Section 3.1.1) we have that for each $\omega \in \Omega_v^G$ such that $\omega_v = [n, n_+)$ it holds that $K_v(\omega) = K_v(\omega_{u \leftarrow [n_-, n)})$. We therefore can set $k = K_v(\omega) = K_v(\omega_{u \leftarrow [n_-, n)})$ and from (5.3) we immediately see that (5.2) is satisfied for the whole range $[n_-, n_+)$.

□

5.3.2 Monotone Target Value

To relate the dynamics captured in a TS with a parametrization value, we introduce the notion of monotone target value. Intuitively, for a state $s \in S$ an MTV is a value towards which the component value s_v evolves if we traverse only in the dimension of v until s_v either stabilizes or an opposing effect takes place. This idea is strongly linked to the derivative of the update function.

Consider the example in Figure 5.2b. Under the influence of the edge $(u, 1, v)$ we see the trace $(1, 0) \rightarrow^T (1, 1) \leftrightarrow^T (1, 2)$. Here the MTV for component v in the state $(v, u) = (0, 0)$ is 2 since $\delta(F_v^K)(1, 0) = +1 = \delta(F_v^K(1, 1)) \neq \delta(F_v^K(1, 2)) = -1$, *i.e.*, at the level $s_v = 2$ an opposing effect takes place.

For a $T^K = (S^{[K]}, \rightarrow)$, $v \in V$, $\omega \in \Omega_v^G$ and any $s \in \omega$ we denote the MTV by $(F_v^K)^{mon}(s)$ defined as:

$$(F_v^K)^{mon}(s) = \begin{cases} s_v & \text{if } \delta(F_v^K)(s) = 0, \\ min\{j > s_v \mid \delta(F_v^K)(s_{v \leftarrow j}) \neq +1\} & \text{if } \delta(F_v^K)(s) = +1, \\ max\{j < s_v \mid \delta(F_v^K)(s_{v \leftarrow j}) \neq -1\} & \text{if } \delta(F_v^K)(s) = -1. \end{cases} \tag{5.4}$$

Also note that:

$$\forall s \in S^{[K]}, \forall v \in V : \delta(F_v^K)(s) = sgn((F_v^K)^{mon}(s) - s_v). \tag{5.5}$$

The MTV can now be related to the observability in the TS. In particular, we can rewrite (5.2) as:

$$\begin{aligned} &\forall s \in S^{[K]}, s_u \in [n_-, n_+), \forall j \in [n_-, n_+) : \\ &\delta(F_v^K)(s_{u\leftarrow j}) = sgn((F_v^K)^{mon}(s) - (s_{u\leftarrow j})_v). \end{aligned} \tag{5.6}$$

Lemma 5.3.2. *The non-observability conditions (5.2) and (5.6) are equivalent.*

Proof. Clearly, (5.6) implies (5.2) since we can set $k = (F_v^K)^{mon}(s)$. For the other direction we show that if (5.6) does not hold, then (5.2) cannot hold either.

If (5.6) does not hold, then there is a state $s \in S^{[K]}$ and some $j \in [n_-, n_+)$ such that $\delta(F_v^K)(s') \neq sgn((F_v^K)^{mon}(s) - s'_v)$ where $s' = s_{u\leftarrow j}$. We distinguish three cases based on the value of $\delta(F_v^K)(s')$ and then again three cases based on the difference between s_v and s'_v:

Case $\delta(F_v^K)(s') = 0$ and $\boldsymbol{sgn}((F_v^K)^{mon}(s) - s'_v) \neq 0$:

- If $s_v = s'_v$, then for any k certainly $sgn(k - s'_v) = sgn(k - s_v)$. Also since $sgn((F_v^K)^{mon}(s) - s'_v) \neq 0$ we get $(F_v^K)^{mon}(s) \neq s'_v = s_v$, and thus $\delta(F_v^K(s)) \neq 0$ according to (5.5). Then if there is a k s.t. $\delta(F_v^K)(s') = sgn(k - s'_v)$ then also $sgn(k - s'_v) \neq \delta(F_v^K(s))$ and therefore (5.2) does not hold.
- If $s_v > s'_v$, it follows that $u = v$. From (5.4) we get $(F_v^K)^{mon}(s) \geq s'_v$ since $\delta(F_v^K)(s') = 0$ indicating an effect change in the only problematic case that $\delta(F_v^K)(s) = -1$. Since $sgn((F_v^K)^{mon}(s) - s'_v) \neq 0$ we have strict inequality $((F_v^K)^{mon}(s)) > s'_v$. Subsequently by (5.4) there exists l such that $s_v \geq l > s'_v$ and $\delta(F_v^K)(s_{v\leftarrow l}) \geq 0$. We have that $s'_v \in [n_-, n_+)$ and $\delta(F_v^K)(s') = 0$, so to fulfil (5.2) the k must be chosen as $k = s'_v$. But $\delta(F_v^K)(s_{v\leftarrow l}) \geq 0$ and $l \in (s'_v, s_v] \subseteq [n_-, n_+)$ and therefore k needs to satisfy $k > (s_{v\leftarrow l})_v = l$. Together we get $k = s'_v < l \leq k$ which is a contradiction.
- The case $s_v < s'_v$ can be treated analogously to $s_v > s'_v$.

Case $\delta(F_v^K)(s') = +1$ and $\boldsymbol{sgn}((F_v^K)^{mon}(s) - s'_v) \leq 0$:

- If $s_v = s'_v$, then from $sgn((F_v^K)^{mon}(s) - s'_v) \leq 0$ it follows that $(F_v^K)^{mon}(s) \leq s'_v = s_v$ and therefore we have that $\delta(F_v^K)(s) \leq 0$. Then k in (5.2) needs to satisfy $k \leq s_v$. Also $\delta(F_v^K)(s') = +1 = sgn(k - s'_v)$ and thus k needs to satisfy $k > s'_v$ leading to a contradiction.

- The case $s_v > s'_v$ is impossible, since $s_v > s'_v \geq (F_v^K)^{mon}(s)$, so $\delta(F_v^K)(s) = -1$ and therefore by (5.4) also $\delta(F_v^K)(s') = -1$ which is a contradiction.
- If $s_v < s'_v$, then by (5.4) there is l such that $s_v \leq l < s'_v$ and $\delta(F_v^K)(s_{v \leftarrow l}) \leq 0$. As $l \in [n_-, n_+)$, a k satisfying condition (5.2) must be such that $sgn(k - (s_{v \leftarrow l})_v) \leq 0$ and thus $k \leq l$. But $\delta(F_v^K)(s') = +1$ and a suitable k must also satisfy $s'_v < k$. Together we again have the contradiction $k \leq l < s'_v < k$.

Case $\delta(F_v^K)(s') = -1$ and $\boldsymbol{sgn}((F_v^K)^{mon}(s) - s'_v) \geq 0$:
This case can be treated like the previous one.

□

5.3.3 Normalization Algorithm

We can see that the MTVs $(F_v^K)^{mon}(s)$ allow for a straightforward test of observability in the TS. Obtaining $(F_v^K)^{mon}(s)$ is however quite tedious. The size of $S^{[K]}$ is exponential w.r.t. the set V and we have to unfold the TS to find the monotone paths characterizing the MTVs. In this section we show that all states of a context share their MTV and additionally that we can obtain the MTV from a context directly.

We introduce the *normalization* function, described in Algorithm 1, that computes for each component and for each regulatory context of that component the MTV shared between the states of the context.

In the algorithm we traverse through the contexts, rather than through states of a system, when looking for a monotone trace. As an example consider the K'_u in Figure 5.1a and $\omega = ([0,1),[1,2))$. Then $Norm(K', v, \omega) = Norm(K', v, \omega_{v \leftarrow [n_{i+1}, n_{i+2})}) = K'_v(\omega_{v \leftarrow [n_{i+1}, n_{i+2})}) = 2$. Note that this coincides with the value in Figure 5.1b, which actually has a normalized parametrization.

The correctness of the approach is quite intuitive since a regulatory context is a subspace of the state space with uniquely determined target value. This means that either the behaviour is monotone or there is exactly one stable state which breaks monotonicity in both directions. Considering the definitions of $(F_v^K)^{mon}(s)$ and the derivative. Easy calculations for the three cases $K_v(\omega) < \omega_v$, $K_v(\omega) > \omega_v$ and $K_v(\omega) \in \omega_v$ for a context ω immediately prove the following lemma.

Lemma 5.3.3. *For any $v \in V$, any $\omega \in \Omega_v^G$ and $K \in \mathcal{K}^G$ we have $(F_v^K)^{mon}(s) = (F_v^K)^{mon}(s')$ for all $s, s' \in \omega$.*

Due to this lemma we can extend the notion of MTV to regulatory contexts ω so that $(F_v^K)^{mon}(\omega) = (F_v^K)^{mon}(s)$ for any $s \in \omega$. Having this extension, we now prove the correctness of Algorithm 1.

Algorithm 1 Calculate $\mathit{Norm}(K, v, \omega)$ where $\Theta(v, v) = \{n_0, ..., n_k\}$.

```
$[n_i, n_{i+1}) = \omega_v$
**if** $K_v(\omega) \in \omega_v$ **then**
    $\mathit{Norm}(K, v, \omega) = K_v(\omega)$
**else if** $K_v(\omega) < \omega_v$ **then**
    $\omega' = \omega_{v \leftarrow [n_{i-1}, n_i)}$
    **if** $K_v(\omega') \geq n_i - 1$ **then**
        $\mathit{Norm}(K, v, \omega) = n_i - 1$
    **else**
        $\mathit{Norm}(K, v, \omega) = \mathit{Norm}(K, v, \omega')$
    **end if**
**else**
    $\omega' = \omega_{v \leftarrow [n_{i+1}, n_{i+2})}$
    **if** $K_v(\omega') \leq n_{i+1}$ **then**
        $\mathit{Norm}(K, v, \omega) = n_{i+1}$
    **else**
        $\mathit{Norm}(K, v, \omega) = \mathit{Norm}(K, v, \omega')$
    **end if**
**end if**
```

Theorem 5.3.4. *For each $v \in V^G$, each $\omega \in \Omega_v^G$ it holds that $(F_v^K)^{mon}(\omega)$ is equal to $\mathit{Norm}(K, v, \omega)$.*

Proof. Linking back to the importance of self-regulation already seen in the canonization, we lead the proof by induction w.r.t. the distance in number of activity intervals of self-regulation of v between the context and its target value. We want to show that $\mathit{Norm}(K, v, \omega)$ returns the correct value after at most as many recursive calls as is the distance between the context and its MTV. The notion of distance needed for this is defined as a function $\mathit{dist}_v^K : \Omega_v^{[K]} \to \mathbb{N}_0$ with

$$\mathit{dist}_v^K(\omega) = \mathit{max}([\![\{A \in I_v^v \mid \omega_v < A \leq A_\omega\}]\!], [\![\{A \in I_v^v \mid \omega_v > A \geq A_\omega\}]\!]),$$

where $A_\omega \in I_v^v$ is the activity interval where it holds that $(F_v^K)^{mon}(\omega) \in A_\omega$.

Now we prove the theorem by the means of induction. Note that the proof does not constitute an invariant of the algorithm, as it proceeds in the other direction than the algorithm itself. In particular, we start by showing that for all the contexts that have their MTV within them or on their boundaries, the algorithm ends immediately with the correct value. Then we proceed to show that if the normalized parameter of any context whose distance is m is correct and known, then the normalized parameter of a context whose distance is $m + 1$ can be correctly determined by calling Algorithm 1 once.

Base of induction (distance 0 and 1):
If $dist_v^K(\omega) = 0$ then we know that $(F_v^K)^{mon}(\omega) \in \omega_v$. This implies that there is a state $s \in \omega$ such that $K_v(\omega) = s_v$ and $\delta(F_v^K)(s) = 0$ according to (2.2) and (5.4). Therefore $(F_v^K)^{mon}(\omega) = K_v(\omega) = Norm(K, v, \omega)$, as set on the lines 2, 3. The recursion depth is 0.

If $dist_v^K(\omega) = 1$, denote $\omega' = \omega_{v \leftarrow A}$ such that $(F_v^K)^{mon}(\omega) \in A$. It follows from $dist_v^K(\omega) = 1$ that $dist_v^K(\omega') \leq 1$. We distinguish the two options:

- $dist_v^K(\omega') = 0$: Then the above argument repeats and there is $s \in \omega$ such that $K_v(\omega') = s_v = (F_v^K)^{mon}(\omega')$. By the definition of the MTV we get $(F_v^K)^{mon}(\omega') = (F_v^K)^{mon}(\omega)$. Based on the ordering of ω, ω' we arrive either on the line 9 or 16 of the algorithm and state correctly that $(F_v^K)^{mon}(\omega) = Norm(K, v, \omega) = Norm(K, v, \omega') = s_v$. The recursion depth is 1.

- $dist_v^K(\omega') = 1$: Since $dist_v^K(\omega) = 1$ it follows that $(F_v^K)^{mon}(\omega') \in \omega_v$. In this case the respective MTVs take the adjacent values of the boundary between ω_v and ω'_v. In the case that $\omega > \omega'$ we arrive on line 7 in the algorithm and assign $Norm(K, v, \omega) = n_i - 1$. This is correct as in any state of ω we monotonously update towards ω' and as we enter ω' by crossing the boundary value n, we change the direction back towards ω, breaking the monotonicity. Analogously the correct value is assigned for the case $\omega < \omega'$. The recursion depth is 0.

Induction step (distance over 1):
The induction assumption is that in at most recursion depth $m \geq 1$ the value of any $\omega' \in \Omega_v^G$ such that $dist_v^K(\omega') \leq m$ is correctly set and consider now $dist_v^K(\omega) = m + 1$.

In case $\omega_v > (F_v^K)^{mon}(\omega)$, since $dist_v^K(\omega) > 1$, we have an ω' such that $dist_v^K(\omega') = m \geq 1$ and $\omega_v > \omega'_v > (F_v^K)^{mon}(\omega)$. From the definition of the MTV it follows that all states in both ω and ω' monotonously decrease under F_v^K, therefore also $\omega'_v > (F_v^K)^{mon}(\omega')$ which gives us $(F_v^K)^{mon}(\omega) = (F_v^K)^{mon}(\omega')$. In the algorithm this is assured on line 9 and by induction hypothesis $(F_v^K)^{mon}(\omega')$ is correctly determined by the algorithm in at most m recursions, giving us the desired result for $(F_v^K)^{mon}(\omega)$. The case that $\omega_v < (F_v^K)^{mon}(\omega)$ is again analogous, leading to line 14 instead of 9. The recursion depth is now $m + 1$.

Since the set of intervals is finite and the recursion traverses monotonously, we terminate in the recursion depth of at most $max(\{\rho(v) \mid v \in V\})$. □

Using the normalization function we can, similarly to canonization, create a conservative and sound transformer on parametrizations. We extend $Norm$ to a function $Norm : \mathcal{K}^G \to \mathcal{N}^G$ where $\mathcal{N}^G \subseteq \mathcal{K}^G$ is the set of normalized parametrizations of $G = (V, E, \rho)$ and $Norm(K) = N$ where N is defined by

$$\forall v \in V, \forall \omega \in \Omega_v^G : N_v(\omega) = \mathit{Norm}(K, v, \omega).$$

We have proven correctness of normalization already in Theorem 5.3.4, so it only remains to prove that normalization is conservative.

Lemma 5.3.5. *Normalization is conservative. For all $G \in \mathcal{G}$ and every $K \in \mathcal{K}^G$ it holds that if $\mathit{Norm}(K) = N$ then $T^K = T^N$.*

Proof. Observe that if $K_v(\omega) \in \omega_v$ then $\mathit{Norm}(K, v, \omega) = K_v(\omega)$. In Corollary 5.2.4 we have shown that if $K_v(\omega) < \omega_v$, then it is conservative to replace $K_v(\omega)$ with any $l \in \mathbb{N}_0$ such that $l < \omega_v$, which is also the case in Algorithm 1. The same holds for the case that $K_v(\omega) > \omega_v$. □

5.3.4 Observability in Normalized Parametrization

We have seen now that observability in the sense of an actual dynamical effect should not be evaluated based on the parametrization but rather on the TS. All information needed to construct a TS is captured in the MTVs due to its relation to the derivative and thus the update function. At the same time, an important aspect of parametrizations is shared, namely that the MTV stays fixed within a context. This allows us to link observability in parametrization and TS, as is shown in the following theorem that complements Lemma 5.3.1.

Theorem 5.3.6. *For every $G \in \mathcal{G}$ and every $K \in \mathcal{K}^G$ it holds that if $\mathit{Norm}(K) = N$ then every edge that is not observable in T^K is not observable in N.*

Proof. Assume that the above does not hold, *i.e.* there exists an edge $(u, n, v) \in E$ s.t. (5.6) holds, but also it holds that:

$$\exists \omega \in \Omega_v^G, \omega_u = [n, n_+), \omega^{\downarrow} = \omega_{u \leftarrow [n_-, n)} : N_v(\omega) \neq N_v(\omega^{\downarrow}). \tag{5.7}$$

Case $u \neq v$:
Note that in this case we have $\omega_v = \omega_v^{\downarrow}$.

We have $\delta(F_v^N)(s) = \delta(F_v^K)(s)$ for all states $s \in \omega$ as can be easily deduced from Lemma 5.3.5. For all $s \in \omega, s' \in \omega^{\downarrow}$ we have $(F_v^K)^{\mathit{mon}}(s) = N_v(\omega) \neq N_v(\omega^{\downarrow}) = (F_v^K)^{\mathit{mon}}(s')$. It follows that we can only meet the condition $\mathit{sgn}((F_v^K)^{\mathit{mon}}(s) - s_v) = \mathit{sgn}((F_v^K)^{\mathit{mon}}(s) - s'_v) = \delta(F_v^K)(s')$ for all $s \in \omega, s' = s_{u \leftarrow j}, j \in \omega^{\downarrow}$ as demanded in (5.6) if and only if either $N_v(\omega) < \omega_v \wedge N_v(\omega^{\downarrow}) < \omega_v^{\downarrow}$ or $N_v(\omega) > \omega_v \wedge N_v(\omega^{\downarrow}) > \omega_v^{\downarrow}$.

First consider that $(N_v(\omega) < \omega_v) \wedge (N_v(\omega^{\downarrow}) < \omega_v^{\downarrow})$. If the condition on the line 7 is satisfied for both ω and $\omega^{\downarrow}$ we immediately see that $N_v(\omega) = N_v(\omega^{\downarrow})$. If it is satisfied for exactly one, then apparently we break the requirement (5.6) as for $s \in \omega'_v$ we have $(s_{u \leftarrow n_-})_v = s_v$ but $\delta(F_v^K)(s_{u \leftarrow n_-}) \neq \delta(F_v^K)(s)$.

We therefore meet the condition on the line 8 and from the line 9 we know that for ω' as defined there $N_v(\omega') = N_v(\omega) \neq N_v(\omega^{\downarrow}) = N_v((\omega^{\downarrow})')$. Since $N_v(\omega') \neq N_v((\omega^{\downarrow})')$ we have again that $N_v(\omega') \notin \omega'_v$ and the same for $(\omega^{\downarrow})'$. Therefore it again must hold that $N_v(\omega') < \omega'_v$ and $N_v((\omega^{\downarrow})') < (\omega^{\downarrow})'_v$. Apparently, the argument is recursive, requiring that for each $\omega' \in \Omega^G_v$ such that $\omega'_v < \omega_v$ it holds that $N_v(\omega') < \omega'_v$. But then ultimately $N_v(\omega) < 0$ which contradicts the definition of K.

For the case that $N_v(\omega) > \omega_v$ a similar argument holds using the upper boundary $N_v(\omega) \leq \rho(v)$.

Case $u = v$:

First note that in this case we have $\omega' = \omega^{\downarrow}$ for ω' as defined in the algorithm. In case that $k = (F^K_v)^{mon}(s) < n$ for any $s \in \omega$ we execute $\mathit{Norm}(K, v, \omega)$ in the algorithm and the condition on the line 4 is met. Then depending on the value $K(\omega')$:

- $K(\omega') \geq n - 1$: then $k = n - 1$, meaning that for any $s' \in \omega'$ we have $(F^K_v)^{mon}(s')_v = n - 1$, otherwise the edge would be observable. But then also $N_v(\omega) = N_v(\omega')$ which contradicts (5.7).
- $K(\omega') < n - 1$: then according to the line 9 we have $\mathit{Norm}(K, v, \omega) = \mathit{Norm}(K, v, \omega')$, again contradicting (5.7).

The case that $k \geq n$ can be treated similarly.

□

5.3.5 Minimizing the Model

Since we know that after normalization, all non-observable edges can be directly detected, constructing a reduction algorithm is rather straightforward. Our minimization process is closely related to the completion process in Section 5.2.

We use the *minimization* function $\mathit{Minim} : \mathcal{N}^G \to \mathcal{N}^{\check{G}}$ to eliminate the non-observable edges. If $[N]$ is minimal, *i.e.* there are no non-observable edges, then we map N to itself. Otherwise denote $\check{E}$ the set of non-observable edges in $[N] = (V, E, \rho)$ and an arbitrary total ordering on $\check{E}$ and $(u, n, v) = \mathit{min}(\check{E})$, then:

1. $[\check{N}] = \check{G} = (V, E \setminus \{(u, n, v)\}, \rho)$,
2. $\check{N}_v(\omega_{u \leftarrow [n_-, n_+)}) = N_v(\omega_{u \leftarrow [n_-, n)}) = N_v(\omega_{u \leftarrow [n, n_+)})$.

The nature of MTVs ensures that $\check{N}$ is again in $\mathcal{N}^{\check{G}}$. As an example consider the network in Figure 5.1b. The edge $(v, 1, v)$ is apparently not observable. We therefore remove it from E and with that we set

$$\check{N}_v([0,3),[0,1)) = N_v([0,1),[0,1)) = N_v([1,3),[0,1)) = 0$$
$$\check{N}_v([0,3),[1,2)) = N_v([0,1),[1,2)) = N_v([1,3),[1,2)) = 2.$$

Note that this coincides with the network in Figure 5.1c, which is in fact minimized and normalized.

This procedure can be seen as an inversion to the completion as defined in Section 5.2, where we created two new contexts by splitting one, keeping the values, whereas here we merge two contexts with the same value into one. Note that the fixpoints of *Comp* and *Minim* are not dependent on the order on $V \times \mathbb{N}_1 \times V$. We can therefore take arbitrary, but fixed, order and execute both *Comp* and *Minim* according to this order. Then for $\mathit{Minim}(N) = \check{N}$ with $N \in \mathcal{N}$ and $G \neq \check{G}$ we have $\mathit{Comp}(\check{N}) = N$. This can be easily verified by applying the two operations successively. Since completion is sound and conservative, minimization in such corresponding cases is also sound and conservative. The only remaining case is that $\mathit{Minim}(N) = N$, but this is obviously sound and conservative too.

Iteration of the *Minim* function will then lead to a minimal structure w.r.t. the number of regulations, as demonstrated by the following lemma:

Lemma 5.3.7. *For $G \in \mathcal{G}$, $K \in \mathcal{K}^G$ and $\mathit{Norm}(K) = N$ exists $n \in \mathbb{N}_1$ s.t. $\mathit{Minim}^n(N) = \mathit{Minim}^{n+1}(N)$. Then $\mathit{Minim}^n(N)$ is minimal,* i.e. *there are no other K' s.t. $T^K = T^{K'}$ and $[K']$ has less edges than $[\mathit{Minim}^n(N)]$.*

Proof. Since we change N only if we remove an edge and the edge set is finite, the existence of a fixpoint is trivial. If $[\mathit{Minim}^n(N)]$ was not minimal, then inevitably there would have to be an edge which is not observable in T^K but observable in N which contradicts Theorem 5.3.6.

□

Now we can conclude the section with a theorem about equivalence checking through minimization, complementing the result of Section 5.2 and showing that the set of dynamically equivalent RNs has both maximal and minimal elements w.r.t. set inclusion on the regulators.

Theorem 5.3.8. *Let $G, G' \in \mathcal{G}$, $K \in \mathcal{K}^G$, $K' \in \mathcal{K}^{G'}$ and denote Minim^* the fixpoint of Minim. Then $T^K = T^{K'}$ if and only if $\mathit{Minim}^*(\mathit{Norm}(K)) = \mathit{Minim}^*(\mathit{Norm}(K'))$.*

Proof. Similarly to the proof of Theorem 5.2.8, we now know that both $\mathit{Minim}^*(\mathit{Norm}(K))$ and $\mathit{Minim}^*(\mathit{Norm}(K'))$ are normalized and minimal. Due to Theorem 5.3.6 we know that in the minimization exactly the non-observable edges in the TS are removed. So if the two TSs are equivalent, the set of remaining edges is the same in both G and G'.

Additionally we know that the parameter values are not changed during the minimization, only in the normalization where the value is set to the MTV of the states covered by the context. The only way how we could have a difference between $\mathit{Minim}^*(\mathit{Norm}(K))$ and $\mathit{Minim}^*(\mathit{Norm}(K'))$ is that there is a state $s \in T^K$ and a component v such that $(F_v^K)^{mon}(s) \neq (F_v^{K'})^{mon}(s)$.

But then by (5.4) we have some $r \in T^K$ such that $\delta(F_v^K)(r) \neq \delta(F_v^{K'})(r)$ and $T^K \neq T^{K'}$.

□

5.3.6 Complexity

It has been shown that testing equivalence for two expressions over n variables is co-NP complete [BHR84]. Since in a Boolean network a component $v \in V$ can be regulated by up to $[\![V]\!]$ nodes, the update function can be an expression over $[\![V]\!]$ variables and deciding whether two parametrizations are equivalent is therefore necessarily exponential w.r.t. the number of components. However as already stated in Section 5.1.3, the complexity is still favourable. In particular:

Theorem 5.3.9. *$TIME(Minim^*(Norm(K))) \in \mathcal{O}([\![V^{[K]}]\!] \cdot size(\Omega^{[K]}))$.*

Proof. As we call $Norm(K, v, \omega)$ for each $\omega \in \Omega_v$ no recursion is needed—either we set the value directly or we set it equal to some other value that will eventually be known. We therefore call $Norm(K, v, \omega)$ at most $size(\Omega^{[K]})$ times for each $v \in V$.

In the minimization part we need to obtain the set $\check{E}$. For an edge $(u, n, v) \in E$ to detect whether $(u, n, v) \in \check{E}$ we need to consider all $\omega \in \Omega_v^{[K]}$ with $\omega_u = [n, n_+)$ and compare them to the respective $\omega_{u \leftarrow [n_-, n)}$. At most we need to do $(size(\Omega^{[K]})/2) \in \mathcal{O}(size(\Omega^{[K]}))$ pair-wise comparisons. Since we have to consider at most k regulators of $[\![V]\!]$ components, we obtain in total $\mathcal{O}([\![V^{[K]}]\!] \cdot size(\Omega^{[K]}))$. To remove the edge we remove the set of tested contexts, which again leads to $\mathcal{O}([\![V^{[K]}]\!] \cdot size(\Omega^{[K]}))$.

□

Intuitively, since we only change the values in place or remove them, no additional space is needed. The spatial complexity is thus equal to the size of the input.

5.4 Encoding

In this section we focus on encoding the properties and their features as described in Section 3.2 into BA in order to be able to conduct the model checking procedure.

5.4.1 Basic Measurements Automaton

As the knowledge about a system is usually obtained by measuring concentration or activity of a component, we use *measurements* as a basic unit of our encoding of data. Intuitively, a *measurement* describes the activity of some of the components at one time point.

Formally we describe a measurement M in a TS $T^K = (S^{[K]}, \rightarrow^T)$ as predicate over $S^{[K]}$, *i.e.* $M : S^{[K]} \rightarrow \{true, false\}$. Interpreted as a set, we also intuitively have that $M \subseteq S^{[K]}$, meaning that a measurement is the set of states that match the data.

A sequence of measurements $\overrightarrow{M}^G = (M^1, \ldots, M^n), n \in \mathbb{N}_1$ can be encoded via an automaton that loops in its current state until its respective measurement is matched and then it proceeds to a next state. After the last measurement, an accepting state with a loop is added. However note that reaching the accepting state already proves that there is a path that matches the sequence of measurement, therefore using a TBA is sufficient. Formally we use a TBA $B^G = (S^B, \xrightarrow{\mathcal{L}(G)}, I^B, A^B)$ where:

$$\begin{aligned}
S^B &= \{b^1, \ldots, b^{n+1}\},\\
\xrightarrow{\mathcal{L}(G)} &= \bigcup_{j \in [1,n]} \{b^j \xrightarrow{\neg M^j} b^j, b^j \xrightarrow{M^j} b^{j+1}\} \cup \{b^{n+1} \xrightarrow{true} b^{n+1}\},\\
I^B &= \{b^1\},\\
A^B &= \{b^{n+1}\}.
\end{aligned}$$

An example of such an automaton is the one in Figure 2.2a with $\overrightarrow{M}^G = (M^1, M^2), M^1 \equiv (CompA = 0), M^2 \equiv (CompB = 1)$. Clearly the size of an automaton encoding a measurement series is linear w.r.t. to the number of measurements. This is advantageous as for an arbitrary property the resulting automaton can be exponential in the worst case [BK08]. However the automaton there also shows that there is a slight inconsistency w.r.t. the definition in Section 3.2.1. In particular, while there is a single state $s^T = (1, 0)$ s.t. $s^T, T^K \models (M^1, M^2)$, there must be at least three steps in the product for the automaton to be traversed, therefore the witness is at least three states long, even though one should be sufficient. To tackle the problem, we are introducing a new version of Büchi automata.

5.4.2 State-conditional Büchi Automata

To improve the performance of the methods, we create a class of automata that have condition on whether a state is initial or accepting. As this is a strict superset of the standard BA, we keep the notation, meaning that Büchi automata are from now on given as $B^G = (S^B, \xrightarrow{\mathcal{L}(G)}, I^B, A^B)$, where:

- S^B is a set of states,
- $\xrightarrow{\mathcal{L}(G)}$ is a transition relation with propositions where $\mathcal{L}(G) = \mathbb{P}(\{v * n \mid v \in V^G, * \in \{\leq, \geq, <, >, =\}, n \in [0, \rho(v)]\})$,
- $I^B : S^B \rightarrow \mathcal{L}(G)$ is an initial condition function,

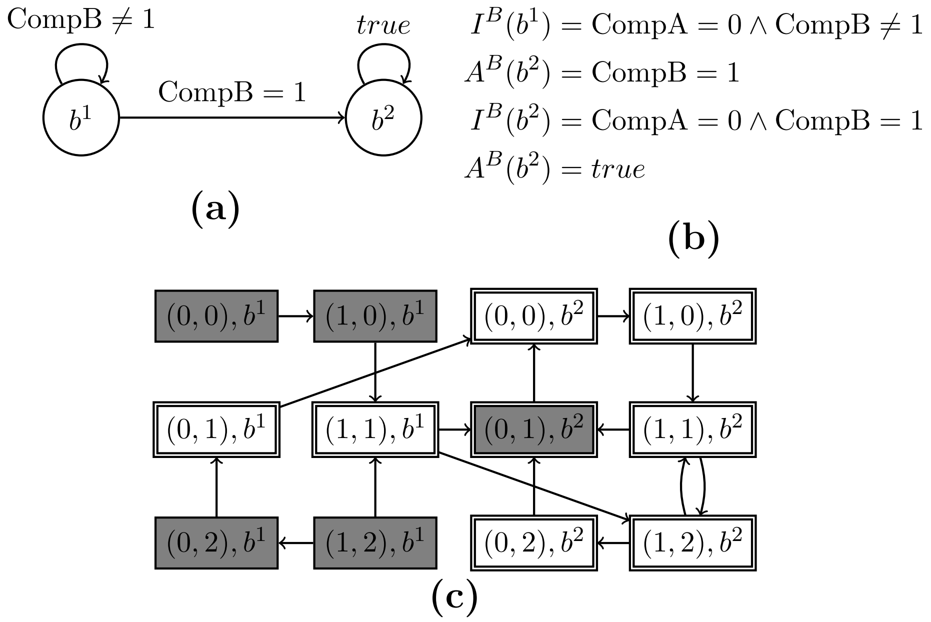


Figure 5.3: **(a)** A new encoding of the property in Figure 2.2, i.e. a TBA($\overrightarrow{M}^G$) with $M^1 \equiv \text{CompA} = 0$ and $M^2 \equiv \text{CompB} = 1$. **(b)** The respective accepting conditions. Note that a conjunction over an empty set is equal to *true*.**(c)** A product with the T^K from Figure 2.1. Note that the product accepts immediately in the state $((0,1), b^2)$, meaning that $cost^X = 1$.

- $A^B : S^B \to \mathcal{L}(G)$ is an accepting condition function.

The synchronous product is then constructed as $X^K = (S^X, \to^X, I^X, A^X)$ where:

- $S^X = S^{[K]} \times S^B$,
- $I^X = \{(s,b) \in S^X \mid s \models I^B(b)\}$,
- $A^X = \{(s,b) \in S^X \mid s \models A^B(b)\}$,
- $(s,b) \to^X (s',b') \iff (s \to^T s') \wedge (b \xrightarrow{\phi} b') \wedge (s \models \phi)$.

Note that if we restrict the state conditions only to $\{true, false\}$ we obtain the normal BA.

5.4.3 Encoding Measurements

There are multiple changes we are applying to the basic encoding both to accommodate the semantics and to optimize the size.

First, as we are looking only for the shortest witnesses in general, we can focus only on searching paths that start in a state that matches at least the first measurement. This fully removes the first state of the automaton in Section 5.4.1, since it is responsible for verifying exactly the first measurement.

Second, if there is a state that matches multiple measurements at once, we start even further in the automaton. Conversely, if the automaton has not reached its final state, however the current state matches all the remaining measurements, we can already accept.

At last, if we match multiple measurements while traversing the automaton, we can potentially skip the states that would be responsible for verifying the respective measurements.

We therefore encode a measurements series $\overrightarrow{M}^G = (M^1, \dots, M^n), n > 0$ as an automaton $B^G = \text{TBA}(\overrightarrow{M}^G)$:

$$\begin{aligned}
S^B =& \{b^1, \dots, b^n\}, \\
\xrightarrow{\mathcal{L}(G)} =& \bigcup_{j \in [1,n]} (\bigcup_{l \in [0,n-j]} \{b^j \xrightarrow{\Phi(j,l)} b^{j+l}\}), \\
\Phi(j,l) =& (\bigwedge_{k \in (j,j+l]} M^k) \wedge (\neg M^{j+l+1} \vee (j+l=n)), \\
\forall j \in [1,n] : I^B(b^j) =& \bigwedge_{l \in [1,j]} M^l \wedge (\neg M^{j+1} \vee n = j), \\
\forall j \in [1,n] : A^B(b^j) =& \bigwedge_{l \in (j,n]} M^l.
\end{aligned}$$

To show that the encoding is correct, we need to show that there is an equivalence between the existence of a set of indices that map measurements to a path w^T and the existence of a path from initial state to an accepting state in the encoding automaton. We show a stronger statement, which also states that the lengths of the path in the TS and in the product are the same and, moreover, the sequence of states of the TS is preserved. Recall that we use BA that is deterministic and total, therefore for each $w^T \in (\rightarrow^T)^{k-1}$ there is exactly one $w^B \in (\xrightarrow{\mathcal{L}(G)})^{k-1}$ s.t. $w^X = w^T \times w^B = ((w^T_l, w^B_l))_{l \in [1,k]} \in (\rightarrow^X)^{k-1}$. For brevity, we will use w^X to denote the pre-image of w^T in X^K as suggested above.

Theorem 5.4.1. *For $B^{[K]} = TBA(\overrightarrow{M}^{[K]})$, $X^K = T^K \times B^{[K]}$, and $w^T \in (\rightarrow^T)^{k-1}$:*

$$\exists I(w^T, \overrightarrow{M}^{[K]}) \iff (w^X_1 \in I^X) \wedge (w^X_k \in A^X).$$

Proof. First observe that the transition function of our encoding is total, therefore any path in the BA combined with any path in the TS will result in a path in the product and *vice versa.* The proof is divided into three parts, each showing a correspondence between w^X and w^T:
Part 1:

$$\forall l \in [1, n] : i^l = 1 \wedge i^{l+1} \neq 1 \iff w^X_1 = (w^T_1, w^B_l).$$

Since we require in the definition of $I(w^T, \overrightarrow{M}^{[K]})$ that at least the first measurement is matched, then this statement follows from the definition of the initial condition—the index of the initial BA state is simply equal to the number of measurements satisfied in the initial state.
Part 2:

$$\forall l \in [1, n], i^l = j : i^{l+1} \neq j \iff w^X_{j+1} = (w^T_{j+1}, w^B_l).$$

The initial state matches as many measurements as possible. Have M^{l+1} the first measurement not matched by w^T_1. Then always $(w^T_1, w^B_l) \to (w^T_2, w^B_l)$. This serves as a base of induction.

When walking through the product, two possibilities can occur for each state $w^X_j = (w^T_j, w^B_l)$:

- The measurement M^{l+1} is not matched by w^T_j and therefore we have that $(w^T_j, w^B_l) \to^X (w^T_{j+1}, w^B_l)$, keeping l unchanged. From $j > 1$ is therefore l always the index of the last measurement that has been satisfied before the current state.

- There is a $q \in \mathbb{N}_1$ such that $w^T_j \models M^{l+1} \wedge \cdots \wedge M^{l+q}$, meaning that we move from w^B_l to s^B_{l+q}, therefore $(w^T_j, w^B_l) \to^X (w^T_{j+1}, w^B_{l+q})$ and for $l+q$ it holds $i^{l+q} = j, i^{l+q+1} \neq j$, meaning the equivalence stands.

Part 3:

$$i^l \neq k \wedge i^{l+1} = k \iff w^X_k = (w^T_k, w^B_l) \in A^X.$$

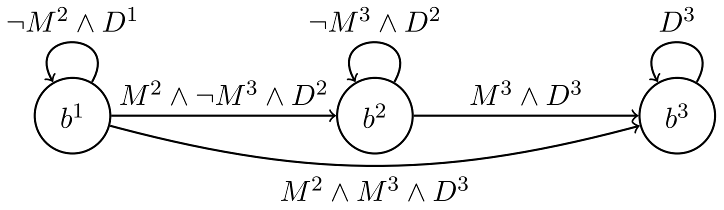
From the previous part we have that $l = i^l$ Since all the measurements with index higher than l are satisfied by s^k we exactly match the accepting condition.

□

5.4.4 Encoding Deltas

Recall that the delta constraint states that for a path $w^T = (s^1, \dots, s^k) \in (\to^T)^{k-1}$ it holds that $w^T \models D^G$, *iff* for each $v \in V^G$ and all the pairs (s^i, s^{i+1}) s.t. $i \in [1, k)$ we have that:

- if $D^G_v = up$ then $s^i_v \leq s^{i+1}_v$,



$$\begin{aligned} I^B(b^1) &= M^1 \wedge \neg M^2 & A^B(b^1) &= M^2 \wedge M^3 \\ I^B(b^2) &= M^1 \wedge M^2 \wedge \neg M^3 & A^B(b^2) &= M^3 \\ I^B(b^3) &= M^1 \wedge M^2 \wedge M^3 & A^B(b^3) &= \mathit{true} \end{aligned}$$

Figure 5.4: A generic automaton for encoding a delta constrained measurement sequence of length 3, i.e. a TBA($\overrightarrow{M}^G, \overrightarrow{D}^G$), s.t. $[\![\overrightarrow{M}^G]\!] = 3$ for any $G \in \mathcal{G}$.

- if $D^G_v = \mathit{down}$ then $s^i_v \geq s^{i+1}_v$,
- if $D^G_v = \mathit{stay}$ then $s^i_v = s^{i+1}_v$.

To apply the delta constraints, the transition relation of the product X^K is extended s.t.:

$$(s,b) \rightarrow^X (s',b') \iff (s \rightarrow^T s') \wedge (b \xrightarrow{\phi,D} b') \wedge (s \models \phi) \wedge ((s,s') \models D).$$

We then apply this constraint to the encoding of measurements. In particular, consider a measurements vector $\overrightarrow{M}^G, [\![\overrightarrow{M}^G]\!] = n$ and a path constraint vector $\overrightarrow{D}^G, [\![\overrightarrow{D}^G]\!] = n$. Then the condition for the automaton TBA($\overrightarrow{M}^G, \overrightarrow{D}^G$) is created as :

$$\Phi(j,l) = (\bigwedge_{k \in (j,j+l]} M^k) \wedge D^{j+l} \wedge (\neg M^{j+l+1} \vee (j+l=n)).$$

An example of such an automaton is in Figure 5.4. This slightly involved encoding follows from the fact that for each $j \in [1,n)$ the state s^B_j is left only after M^j was satisfied, therefore we already require D^j when leaving s^B_j and the requirement is kept until M^{j+1} is satisfied.

In the first proof we depend on the transition function in the BA being total. This is no longer true, since the transitions that do not match the monotonicity constraints are removed, however for the purposes of the proof we can fix it by adding a non-initial, non-accepting state *sink* s.t. $\{sink \xrightarrow{true}$

sink} and for each $j \in [1, n]$ and $l \in [0, n - j)$ adding a transition with the condition:

$$\Phi(j, l) = (\bigwedge_{k \in (j, j+l]} M^k) \land \neg D^{j+l} \land (\neg M^{j+l+1} \lor (j + l = n)).$$

Understandably, reaching the *sink* means the property can no longer be satisfied, so in practice it is equivalent to not having it at all.

In addition note that removing transitions affects (increases) the robustness value in general, since some of the transitions that would not lead to a witness are removed. While it remains true to its definition, its interpretation may be less intuitive. If that would be a problem, it is again sufficient to make the transition function total as described above. However we opted not to do so for performance reasons.

To show correctness of the encoding, we follow on the assumptions in Theorem 5.4.1, since for a witness with delta constraints to exists, there must be a witness without the constraints in the first place. Therefore we already have that $\exists I(w^T, \overrightarrow{M}^{[K]})$. What remains is to show that w^T satisfies the delta constraints if and only if the encoding automaton accepts.

Theorem 5.4.2. *For* $B^{[K]} = TBA(\overrightarrow{M}^{[K]}, \overrightarrow{D}^{[K]})$*,* $X^K = T^K \times B^{[K]}$*, and* $w^T \in (\rightarrow^T)^{k-1}$*:*

$$\exists I(w^T, \overrightarrow{M}^{[K]}) \land \forall l \in [1, n] : (w^T_{i^l}, \dots, w^T_{i^{l+1}}) \models D^l \iff$$
$$(w^X_1 \in I^X) \land (w^X_k \in A^X).$$

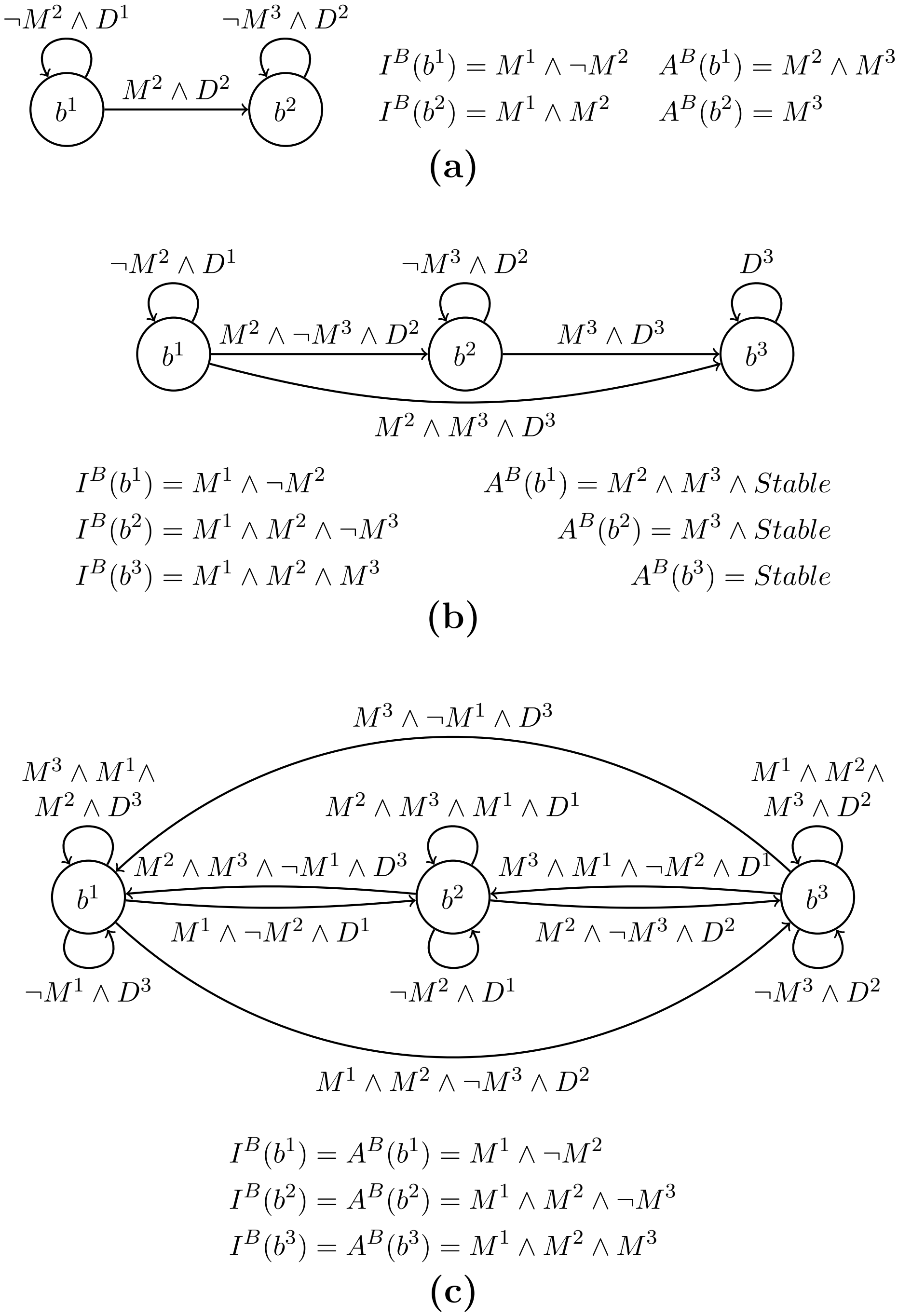
Proof. We follow from the proof of Theorem 5.4.1. In particular we know that the state of the automaton does not change until a new measurement is satisfied. Formally for each $l \in [1, n]$ if $i^l \neq i^{l+1}$ then for each $j \in (i^l, i^{l+1}]$ it holds that exists $b \in S^B$ such that $w^X_j = (w^T_j, b)$. That means that we have to enforce D^l in two positions:

- On b, which covers all the states form $w^T_{i^l+1}$ to $w^T_{i^{l+1}}$, which is guaranteed by the self-loop condition of the automaton.
- On each predecessor of b, i.e. on each b' s.t. $(w^T_{i^l}, b') \rightarrow^X (w^T_{i^l+1}, b)$. This is guaranteed by the fact that each other transition in B^G entering to b has also D^l on its label.

□

5.4.5 Open Ending

First we focus on the *open ending*. Since there is no additional condition, we can actually already use the automaton we have. However there is also space for improvement. In particular we know that the automaton is terminal and


$\neg M^2 \wedge D^1$
$\neg M^3 \wedge D^2$
b^1
$M^2 \wedge D^2$
b^2
$\neg M^2 \wedge D^1$
$\neg M^3 \wedge D^2$
D^3
b^1
$M^2 \wedge \neg M^3 \wedge D^2$
b^2
$M^3 \wedge D^3$
b^3
$M^2 \wedge M^3 \wedge D^3$
$M^3 \wedge \neg M^1 \wedge D^3$
$M^3 \wedge M^1 \wedge$
$M^2 \wedge D^3$
$M^2 \wedge M^3 \wedge M^1 \wedge D^1$
$M^1 \wedge M^2 \wedge$
$M^3 \wedge D^2$
$M^2 \wedge M^3 \wedge \neg M^1 \wedge D^3$
$M^3 \wedge M^1 \wedge \neg M^2 \wedge D^1$
b^1
b^2
b^3
$M^1 \wedge \neg M^2 \wedge D^1$
$M^2 \wedge \neg M^3 \wedge D^2$
$\neg M^1 \wedge D^3$
$\neg M^2 \wedge D^1$
$\neg M^3 \wedge D^2$
$M^1 \wedge M^2 \wedge \neg M^3 \wedge D^2$

$$I^B(b^1) = M^1 \wedge \neg M^2 \quad A^B(b^1) = M^2 \wedge M^3$$
$$I^B(b^2) = M^1 \wedge M^2 \quad A^B(b^2) = M^3$$

(a)

$$I^B(b^1) = M^1 \wedge \neg M^2 \qquad A^B(b^1) = M^2 \wedge M^3 \wedge \mathit{Stable}$$
$$I^B(b^2) = M^1 \wedge M^2 \wedge \neg M^3 \qquad A^B(b^2) = M^3 \wedge \mathit{Stable}$$
$$I^B(b^3) = M^1 \wedge M^2 \wedge M^3 \qquad A^B(b^3) = \mathit{Stable}$$

(b)

$$I^B(b^1) = A^B(b^1) = M^1 \wedge \neg M^2$$
$$I^B(b^2) = A^B(b^2) = M^1 \wedge M^2 \wedge \neg M^3$$
$$I^B(b^3) = A^B(b^3) = M^1 \wedge M^2 \wedge M^3$$

(c)

Figure 5.5: The automaton from Figure 5.4 extended with the *ending* constraint. **(a)** TBA($\overrightarrow{M}^G, \overrightarrow{D}^G, \mathit{open}$). **(b)** TBA($\overrightarrow{M}^G, \overrightarrow{D}^G, \mathit{stable}$). **(c)** DBA($\overrightarrow{M}^G, \overrightarrow{D}^G, \mathit{cyclic}$).

therefore we stop searching as soon as the accepting condition is reached. This implies that the accepting state b^n can never be reached from any other state, since the respective edges are labelled exactly with the accepting condition. The only way how to reach the state b^n is if there exists a state that matches all the measurements at once. In such an event, we know that any state of the automaton is also accepting and we can add this condition to the initial labelling of any of the automata states. We therefore add it to the state b^{n-1}. Note that this reduction is only possible if there are at least two measurements, otherwise the automaton would be empty. For $[\![\overrightarrow{M}^G]\!] > 1$ we then construct TBA$(\overrightarrow{M}^G, \overrightarrow{D}^G, \mathit{open})$ as:

$$\begin{aligned}
S^B =&\{b^1, \dots, b^{n-1}\},\\
\xrightarrow{\mathcal{L}(G)} =& \bigcup_{j\in[1,n)} (\bigcup_{l\in[0,n-j)} \{b^j \xrightarrow{\Phi(j,l)} b^{j+l}\}),\\
\Phi(j,l) =& (\bigwedge_{k\in(j,j+l]} M^k) \wedge D^{j+l} \wedge (\neg M^{j+l+1} \vee (j+l = n-1)),\\
\forall j \in [1,n) : I^B(b^j) =& \bigwedge_{l\in[1,j]} M^l \wedge (\neg M^{j+1} \vee (j = n-1)),\\
\forall j \in [1,n) : A^B(b^j) =& \bigwedge_{l\in(j,n]} M^l.
\end{aligned}$$

An example of such an automaton is in Figure 5.5a.

Above we stated that the construction applies for at least 2 measurements. If there is only one measurement, there would be no states. In such a case we fall back on the construction from Section 5.4.4 and put

$$\text{TBA}((M^1), (D^1), \mathit{open}) = (\{b^1\}, \{b^1 \xrightarrow{D^1} b^1\}, \{b^1 \mapsto M^1\}, \{b^1 \mapsto \mathit{true}\}).$$

However such situation is not expected, as such an automaton accepts *iff* there is any initial state, *i.e.* only controls whether $M^1 \neq \emptyset$.

We now show that due to the condition on the *open ending* that $i^n = i^{n+1}$ our reduction is correct.

Theorem 5.4.3. *For* $B^{[K]} = TBA(\overrightarrow{M}^{[K]}, \overrightarrow{D}^{[K]}, \mathit{open})$*,* $X^K = T^K \times B^{[K]}$*, and* $w^T \in (\rightarrow^T)^{k-1}$*:*

$$\exists I(w^T, \overrightarrow{M}^{[K]}) \wedge \forall l \in [1,n] : (w^T_{i^l}, \dots, w^T_{i^{l+1}}) \models D^l \wedge (i^n = i^{n+1}) \iff (w^X_1 \in I^X) \wedge (w^X_k \in A^X).$$

Proof. The requirement $i^n = i^{n+1}$ means that we accept as soon as the last measurement was matched and therefore do not undergo a transition that would in the normal encoding lead to b^n. b^n still however remains reachable if there is $s \in S^{[K]}$ s.t. for each $l \in [1, n]$ it holds that $s \models M^l$, i.e. if all the measurements overlap. We add this condition to b^{n-1}, which automatically accepts under the M^n, which is satisfied. All the other cases are covered by Theorem 5.4.1. □

5.4.6 Stable Ending

The *stable ending* adds a simple condition that the accepting state must be stable. We can not use the reduction from the *open ending* here, because the state matching the last measurement does not have to be stable. For TBA($\overrightarrow{M}^G, \overrightarrow{D}^G, stable$) we use the same structure as for TBA($\overrightarrow{M}^G, \overrightarrow{D}^G$), only with the accepting condition:

$$\forall j \in [1, n] : A^B(b^j) = \bigwedge_{l \in (j,n]} M^l \wedge Stable.$$

where $s \models Stable \iff s \to s$. An example of such an automaton is in Figure 5.5b. Note that this is the first time the last delta constraint actually has any effect, since now we require the constraints to be applied between the state matching the last measurement and the stable state.

Like in the case of the *open ending*, correctness quite directly follows from the previous theorems.

Theorem 5.4.4. *For $B^{[K]} = TBA(\overrightarrow{M}^{[K]}, \overrightarrow{D}^{[K]}, stable)$, $X^K = T^K \times B^{[K]}$, and $w^T \in (\to^T)^{k-1}$:*

$$\exists I(w^T, \overrightarrow{M}^{[K]}) \wedge \forall l \in [1, n] : (w^T_{i^l}, \ldots, w^T_{i^{l+1}}) \models D^l \wedge (w_{i^{n+1}} \to^T w_{i^{n+1}}) \iff (w^X_1 \in I^X) \wedge (w^X_k \in A^X).$$

Proof. Since the stability requirement is directly placed on the accepting states, the equivalence directly follows. □

5.4.7 Cyclic Ending

The cyclic ending implies that the model is able to return to its initial state after the last measurement was matched. This means that there also has to be a cycle on the verifying automaton. Such a behaviour can no longer be verified by a TBA, since there is no way of making sure that the initial state matches the accepting state and we need to employ a DBA. This means in general a diversion from the structure of the automaton used up till now. There are two major changes:

1. There must be a cycle in the controlling automaton, which moreover must enforce satisfaction of all the measurements in a sequence.

2. Because we want for each witness that the first state is the same as the last one, we need to make the initial condition equal to the accepting one.

To make sure that the automaton conducts a cycle, we need to construct the accepting condition so that it is not possible to accept on a simple loop, therefore the accepting condition must be a subset of the leaving condition for each state. We therefore no longer put the initial condition as the longest sequence of measurements that is satisfied by the state, but we also add the measurement that is controlled by the given state, meaning that after the initial step we leave immediately.

To achieve the cyclical behaviour, we construct the transitions roughly as if we took an automaton for the measurements sequence, composed it so the last state of the first one is the first state of the next one, and then merged the states that control the same measurement. In the formal description we achieve this merging by indexing the states that are in a distance of n. In particular for $[\![\overrightarrow{M}^G]\!] = n$ we put for each $l \in (n, 2n]$ that $b^l \equiv b^{l-n}, M^l \equiv M^{l-n}, D^l \equiv D^{l-n}$ and $D^0 \equiv D^n$.

Formally we use the automaton $B^K = \mathrm{DBA}(\overrightarrow{M}^G, \overrightarrow{D}^G, \mathit{cyclic})$:

$$\begin{aligned}
S^B =& \{b^1, \dots, b^n\}, \\
\xrightarrow{\mathcal{L}(G)} =& \bigcup_{j\in[1,n]} (\bigcup_{l\in[0,n]} \{b^j \xrightarrow{\Phi(j,l)} b^{j+l}\}), \\
\Phi(j,l) =& (\bigwedge_{k\in[j,j+l)} M^k) \wedge ((D^{j+l-1} \wedge \neg M^{j+l}) \vee l = n), \\
\forall j \in [1,n] : I^B(b^j) =& \bigwedge_{l\in[1,j]} M^l \wedge (\neg M^{j+1} \vee j = n), \\
\forall j \in [1,n] : A^B(b^j) =& \bigwedge_{l\in[1,j]} M^l \wedge (\neg M^{j+1} \vee j = n).
\end{aligned}$$

Example of such an automaton is in Figure 5.5c. To show the correctness of this encoding, we have to show an equivalence between the existence of a required cycle in a TS and in the respective SP. Since we are now using a DBA, we have to show that the measurements are again matched in a sequence.

Theorem 5.4.5. *For $B^{[K]} = DBA(\overrightarrow{M}^{[K]}, \overrightarrow{D}^{[K]}, cyclic)$, $X^K = T^K \times B^{[K]}$, and $w^T \in (\rightarrow^T)^{k-1}$:*

$$(\exists I(w^T, \overrightarrow{M}^K)) \wedge (\forall l \in [1,n] : (w^T_{il}, \dots, w^T_{il+1}) \models D^j) \wedge (w^T_1 = w^T_k) \iff (w^X_1 \in I^X) \wedge (w^X_k \in A^X) \wedge (w^X_1 = w^X_k).$$

Proof. Part 1 of the proof of Theorem 5.4.1 remains unchanged, as the initial condition is the same. Moreover, the initial condition implies the accepting condition, meaning that the first state of the witness is also accepting, replacing Part 3 of the original proof.

We now present a modified version of Part 2 of the proof of Theorem 5.4.1. This condition says that after reaching a state w^T_j that satisfies $q = \mu^I(j)$ measurements, we move the BA by q states. The $\mu^I(j)$ is used to denote multiplicity of j in $I(w^T, \overrightarrow{M})$. Formally:

$$\forall j \in [1,k], \exists l \in [1,n], j = i^l : w^X_j = (w^T_j, w^B_q) \to (w^T_j, w^B_{q+\mu^I(j)}).$$

This statement can be easily drawn from the definition of the automaton, as we move always by as many steps as there are satisfied measurements. Also, we do not change the state unless a measurement is satisfied, and conversely we move as soon as there is a satisfied measurement, meaning that the assumption about the correctness of the delta constraints holds.

Lastly, from the above statement we have that after matching n measurements, we move by n steps, but because for each $l \in [1,n]$ we have $b^l = b^{l+n}$ this means we are in the original state, making a cycle. □

5.4.8 Encoding Experiments

For any type of automaton $B^{[K]} = {}^*\mathrm{BA}(\overrightarrow{M}^{[K]}, \overrightarrow{D}^{[K]}, \mathit{End}^{[K]}, \mathit{Exp}^{[K]})$ and a parametrization K, to encode the experimental setup we have to do two changes:

- alter its parametrization so that its target values are bounded by the experiment, meaning that we use the reduced parametrization $K' = \mathit{Reduce}^G(K, \mathit{Exp}^{[K]})$, as explained in Section 3.2.4,
- remove the states outside the experimental setup, *i.e.* put:

$$S^{T'} = \{s \in S^{[K]} \mid \forall v \in V^{[K]} : s_v \in \mathit{Exp}^{[K]}_v\}.$$

The reduced transition system is then $T^{K'} = (S^{T'}, \to^{T'})$ where $\to^{T'}$ is derived from K' as usual. A SP is then constructed as $X = T^{K'} \times B^{[K]}$.

Note that this is the only part of the property that technically changes the original transition system instead of the BA and therefore there is no need to show correspondence between the property in the TS and SP.

5.5 Model Checking with Parameter Uncertainty

In this section we detail the algorithms for model checking under parameter uncertainty, meaning that for a set of parametrization $\mathcal{K}$ and a property $P^{[K]}, K \in \mathcal{K}$ we decide for each $K \in \mathcal{K}$ whether $T^K \models P^{[K]}$. Since this is a computationally difficult procedure, we employ techniques that exploit the structure of the problem for performance gains. These techniques have been published in [SS15].

The core of our approach is the $succ^X$ function (2.5). The successor function, together with X^K itself, is constructed explicitly and provided to each of our algorithms on the output. However we construct it only once for the whole set of parametrizations. Once a product X^K has been checked for some K we can proceed to check $X^{K'}$ for some $K' \neq K$ usually by conducting just a few changes. In particular, the state space $S^{[K]}$ understandably remains always the same. This provides us with a particular advantage in comparison to standard MC tools, which usually do not expect any structural similarity between individual MC instances.

In this section we present three algorithms for computation of cost, trace, and robustness. At first we show how to employ each of these algorithms with a property encoded by TBA. Later the extension to DBA is made. Lastly, we argue that the same extension can also be used with NBA and discuss the implications of non-determinism in terms of complexity and semantics.

The main feature of our algorithms is that we exploit the cost value for practical performance. First please note several aspects of the problem:

- For a network with $[\![V^{[K]}]\!]$ components, the out-degree of any state in T^K is at most $2 \cdot [\![V^{[K]}]\!]$ since we can only increment or decrement by one in each dimension.

- Each component has at least 2 values, so $[\![S^{[K]}]\!] \geq 2^{[\![V^{[K]}]\!]}$ and $[\![V^{[K]}]\!] \leq \log_2([\![S^{[K]}]\!])$.

- The DBA guarantees that for each state of T^K there is only one edge allowed in the automaton, therefore for an $X^K = (S^X, \rightarrow^X, I^X, A^X)$ we have $[\![\rightarrow^X]\!] \leq 2 \cdot [\![S^X]\!] \cdot \log_2([\![S^X]\!])$. For brevity we will further use $size(X) = 2 \cdot [\![S^X]\!] \cdot \log_2([\![S^X]\!])$.

- The product is fully constructed, *i.e.* the time complexity of any algorithm is at least $\mathcal{O}(size(X))$. Also in practice we need $\mathcal{O}(size(X))$ of memory to search through the graph.

- The transition system, if seen as a random process, has a Markov property. If we simulate the model by taking a random walk in T^K (or X^K), the choice of the following state depends only on the current state.

5.5.1 Property Checking

First we present the Algorithm 2 for model checking. Recall that we assume that the property was encoded by TBA, *i.e.* for a product $X^K = (S^X, \rightarrow^X, I^X, A^X)$ we only need to decide whether there is a path from I^X to A^X

Algorithm 2 Calculate $Cost(succ, I, A, full)$.

```
$R \leftarrow I, R' \leftarrow \emptyset, cost \leftarrow 0, depth \leftarrow 1$
for $s \in S$ do
    $visit[s] \leftarrow 0$
end for
while $R \neq 0$ do
    for $r \in R$ do
        $visit[r] = depth$
        if $(r \in A) \wedge (cost = 0)$ then
            $cost \leftarrow depth$
        end if
        $R' \leftarrow R' \cup succ(r)$
    end for
    if $(cost \neq 0) \wedge (\neg full)$ then
        $R \leftarrow \emptyset$
    else
        $R \leftarrow R' \cap \{r \mid visit[r] = 0\}$
    end if
    $R' \leftarrow \emptyset, depth \leftarrow depth + 1$
end while
return $(cost, visit)$
```

using the product successor function, $succ^X$. The algorithm is in its core a simple breadth-first-search (BFS), however it is modified so we obtain the cost value and also a labelling $visit : S^X \rightarrow \mathbb{N}_0$, that stores for each state the depth at which we visited the state. If a state was not visited yet, the *cost* is 0. Note that if the shortest path has length k then $cost = k$ and for any $s \in S$ we have either $visit[s] \in [1, k]$ or $visit[s] = 0$. Also note that we use an additional input parameter, called $full$. Currently we set it to $false$, however it will become useful in Section 5.5.4 for checking with a DBA.

Proposition 5.5.1. *Algorithm 2 is correct. For* $X^K = T^K \times TBA(P^{[K]}) = (S^X, \rightarrow^X, I^X, A^X)$ *it holds that* $(Cost(succ^X, I^X, A^X, false))_1 \neq 0 \iff \neg(T^K \models \neg P^{[K]})$.

Proof. The algorithm is very similar to other implementations of BFS, for exampled the one of [Ski08], and consequently each state is visited at most once. The correctness of the labelling is then trivial. □

Proposition 5.5.2. $SPACE(Cost(succ^X, I^X, A^X, false)) \in \mathcal{O}(size(X))$, $TIME(Cost(succ^X, I^X, A^X, false)) \in \mathcal{O}(size(X))$.

Proof. For the space we only use the labelling *visit* for each state and store of size at most $[\![S^X]\!]$, which can be done in $\log_2([\![S^X]\!])$ space.

The time is given by the fact that we search from each state only once, looping through all the outgoing edges, so again at most $size(X)$. Note however that if the cost is low we can exit the procedure early, examining only a subset of states, which is usually the case in practice. □

5.5.2 Trace

To obtain a witness, we use a recursive depth-first-search (DFS), again modified for our purposes. Recall from Section 2.2.4 that we are looking for all the shortest paths. However, due to the strong non-determinism, the number of shortest paths grows exponentially w.r.t. the cost. To prevent the exponential explosion in space and time complexity, we only store individual transitions that are on some $w \in SW^X$—the trace—as explained in Section 3.4.5. For brevity we will use ST^X instead of $trace^K(P^{[K]})$ for the respective $P^{[K]}$.

To keep the complexity low and avoid searching through the paths we have already visited, we use altogether three distinct state labels. The *visit* label is already provided by Algorithm 2. The $found$ label notes if a state lies on a known witness path and where. The *used* label marks states that we already visited in DFS. Additionally we use the value *branch* which points to the state where we branched from the last path that was found to be a shortest witness path. When a state is known to be part of a shortest witness, either by being a final state or by lying on some already known shortest witness path, we just store the transitions from the last *branch* to the current *depth*, avoiding duplicities.

Proposition 5.5.3. *Algorithm 4 is correct. If* $(cost^X, visit^X) \leftarrow Cost$ $(succ^X, I^X, A^X false)$ *then* $Trace(succ^X, I^X, A^X, visit^X, cost^X) = ST^X$.

Proof. There are two parts to be proven. First, we need to show that the algorithm traverses through all the acyclic paths of length up to cost. In the algorithm we stop traversing in three cases. If the condition on Line 1 is met, then we found the state in BFS sooner than now in DFS and therefore there must exist a shorter path to that state. If the condition on Line 5 is met, then we found a witness. Lastly, if the condition on Line 11 is met, then we either are at maximal depth or we already traversed from the state.

Second, we need to show that each transition is stored exactly once. When a new path is found we see that all transitions are stored on Lines 6-9. At this point we set *branch* to the current *depth* and only decrement by one with each backtracking step. Therefore when storing transitions, we know that those in between 1 and *branch* have been stored already. Also, when we hit a $found$ state, we know that all transitions up from that state have been stored already. □

Algorithm 3 Calculate $DFS(r, depth, branch)$. The labellings $visit$, $found$, $used$, the sets Tr, A, the sequence $Path$, and the function $succ$ are shared between the recursive calls.

```
if $visit[r] < depth$ then
    return $branch$
end if
$Path[depth] \leftarrow r$
if $(r \in A) \vee (found[r] \leq depth)$ then
    for $d \in [branch, depth)$ do
        $Tr \leftarrow Tr \cup (Path[d], Path[d+1])$
        $found[r] \leftarrow depth$
    end for
    $branch \leftarrow depth$
else if $(depth < cost) \wedge (\neg used[r])$ then
    for $r' \in succ(r)$ do
        $branch \leftarrow min(DFS(r', depth+1, branch), depth)$
    end for
end if
$used[r] \leftarrow true$
return $branch$
```

Proposition 5.5.4. *$TIME(Trace(succ^X, I^X, A^X, visit^X, cost^X)) \in \mathcal{O}(size(X))$, $SPACE(Trace(succ^X, I^X, A^X, visit^X, cost^X)) \in \mathcal{O}(size(X))$.*

Proof. The space is again simple—we only use space for states labelling, this time twice. For time complexity we again know that we do traverse any edge twice, since we label a state (*used*) after conducting a search from it and never search from it again. □

Algorithm 4 Calculate $Trace(succ, I, A, visit, cost)$.

```
for $s \in S$ do
    $found[s] \leftarrow 0, used[s] \leftarrow false$
end for
$Tr \leftarrow \emptyset, Path \leftarrow (\bot)_{cost}$
for $i \in I$ do
    $DFS(i, 1, 1)$
end for
return $Tr$
```

5.5.3 Robustness

Lastly we focus on the robustness metric. For computation of robustness we utilize the set ST^X as we know that only the transitions from the set lie on the shortest witness paths. Additionally we also know that since we use the shortest paths, there is no state that would be repeated on any of those paths. We can therefore simply descend through the set of shortest paths in a BFS manner, as we are sure that each state appears in only one iteration of the algorithm. Consequently the probability of reaching a state $s \in S$ in any of the shortest paths is equal to the probability of reaching it in $visit[s]$ steps, which is the invariant of the algorithm.

Proposition 5.5.5. *Algorithm 5 is correct. If* $(cost^X, visit^X) \leftarrow Cost$ $(succ^X, I^X, A^X, false)$ *and* $Tr^X \leftarrow Trace(succ^X, I^X, A^X, visit^X, cost^X)$, *then* $Robustness(succ^X, I^X, A^X, cost^X, Tr^X) \approx robustness^X$.

Proof. The invariant of the proof is that after k iterations, all the states up to the depth k are labelled with the reaching probability by any shortest witness path. Thus, after cost steps we have a labelling for all the final states at distance cost from I^X. Note that we state that the values are possibly only similar since we consider a possibility of having a slight rounding error for fractions for the sake of storage space. □

Proposition 5.5.6. $SPACE(Robustness(succ^X, I^X, A^X, cost^X, Tr^X)) \in \mathcal{O}(size(X))$ *and* $TIME(Robustness(succ^X, I^X, A^X, cost^X, Tr^X)) \in \mathcal{O}(size(X))$.

Proof. Again we use just one label for all the states—in this case a fraction, which we store in at most $\log_2([\![S^X]\!])$ space, having a possible rounding error in practice. Since we have no loops in SW^X, we certainly propagate from each state at most once through each edge, providing the bound of $[\![ST^X]\!] \leq Space^X$. □

5.5.4 Extending to Deterministic BA

Up till now we have discussed usage of the algorithm for properties encoded by TBA. We can however extend the algorithms also to DBA, by stacking multiple calls of each of the algorithms. As explained in Section 2.2, to check for a property encoded by DBA, we are looking not only for a path from some $i \in I^X$ to some $a \in A^X$, but we also need a cycle containing a. We therefore need to first obtain the set $reach(A^X) \subseteq A^X$ of all reachable final states and then we need to decide whether there is a cycle on any $a \in reach(A^X)$. Also, previously we indicated that some of the advantages of the algorithm stem from the witnesses being acyclic, which does not hold any more as we are looking for a cycle on a. However we can break this

Algorithm 5 Calculate $Robustness(succ, I, A, cost, Tr)$.

```
 1: $R \leftarrow I, R' \leftarrow \emptyset, rob \leftarrow 0$
 2: for $s \in S$ do
 3:     if $s \in I$ then
 4:         $prob[s] \leftarrow \frac{1}{[\![I]\!]}$
 5:     else
 6:         $prob[s] \leftarrow 0$
 7:     end if
 8: end for
 9: for $d \in [0, cost]$ do
10:     for $r \in R$ do
11:         $sw[r] \leftarrow \{r' \mid (r, r') \in Tr\}$
12:         $R' \leftarrow R' \cup sw[r]$
13:         for  $r' \in sw[r]$ do
14:             $prob[r'] \leftarrow prob[r'] + \frac{prob[r]}{[\![Succ(r)]\!]}$
15:         end for
16:     end for
17:     $R \leftarrow R'$
18:     $R' \leftarrow \emptyset$
19: end for
20: for $a \in A$ do
21:     $rob \leftarrow rob + prob[a]$
22: end for
23: return $rob$
```

cycle by creating a copy of a—a new state that has the same successors as a, but does not share its labels:

$$\forall s \in S^X : succ^X(s^{copy}) = succ^X(s) \wedge s^{copy} \notin S^X.$$

Lastly, in Algorithm 6 we denote $Tr_{a^{copy} \leftarrow a}$ the set of transitions where a^{copy} was replaced by a.

Proposition 5.5.7. *Algorithm 6 is correct.* $Analyze(X) = (cost^X, ST^X, robustness^X)$.

Proof. First note that we determine the cost already at Lines 3-6. Later we therefore only search for paths that we already know are minimal. This is then done by joining shortest paths from I^X to $a \in A^X$ and from a to itself. Also for such a we know that its initial probability is given as probability of reaching it from I^X. Since in Algorithm 5 we set on Line 4 the probability $prob[a] = \frac{1}{[\![\{a\}]\!]} = 1$, we gain the final probability by multiplying the two. □

Concerning the time complexity of the algorithm we can see that there is a stark increase w.r.t. the size of the set $reach(A)$. In the worst case

Algorithm 6 Calculate $Analyse(X)$ such that $X = T^K \times \text{DBA}(P^{[K]})$.

```
$(cost\_reach, visit\_reach) \leftarrow Cost(succ^X, I^X, A^X, true)$
$cost \leftarrow 0$
for $a \in (A^X \cap \{s \in S^X \mid visit\_reach[s] \neq 0\})$ do
    $(cost\_loop, visit\_loop) \leftarrow Cost(succ^X, a^{copy}, a, false)$
    $cost \leftarrow min(cost, visit\_reach[a] + cost\_loop)$
end for
$Tr \leftarrow \emptyset, Rob \leftarrow 0$
for $a \in (A^X \cap \{s \in S^X \mid visit\_reach[s] \neq 0\})$ do
    $(cost\_loop, visit\_loop) \leftarrow Cost(succ^X, a^{copy}, a, false)$
    if $visit\_reach[a] + cost\_loop = cost$ then
        $Tr\_reach \leftarrow Wintess(succ, I^X, a, visit\_reach, cost\_reach)$
        $Tr\_loop \leftarrow Wintess(succ, a^{copy}, a, visit\_loop, cost\_loop)$
        $Rob\_reach \leftarrow Robustness(succ, I^X, a, cost\_reach, Tr\_reach)$
        $Rob\_loop \leftarrow Robustness(succ, a^{copy}, a, cost\_loop, Tr\_loop)$
        $Tr \leftarrow Tr \cup Tr\_reach \cup Tr\_loop_{a^{copy} \leftarrow a}$
        $Rob \leftarrow Rob + Rob\_reach \cdot Rob\_loop$
    end if
end for
return $(cost, Tr, Rob)$
```

the time is a square of what we had for TBA. Therefore if we expect a big $reach(A)$ set, one may probably want to trade the results provided by our analyses for performance gain of traditional model checking algorithms.

Proposition 5.5.8. *$SPACE(Analyse(X)) \in \mathcal{O}(size(X))$ and $TIME(Analyze(X)) \in \mathcal{O}(size(X) \cdot [\![A^X]\!])$.*

Proof. For the space we see that we keep results of at most two executions of $Cost, Trace$, and $Robustness$ which is only a constant increase. Concerning the time we have two $Cost$ executions for each of the $reach(A^X)$ members, with two executions of $Reach$ and $Trace$. These have again a time bound of $\mathcal{O}(size(X))$, together $\mathcal{O}(size(X) \cdot [\![A^X]\!])$. □

5.5.5 Extending to Non-deterministic BA

Lastly we give a short comment on usage of NBA. There are two main practical differences between DBA and NBA. First, the complexity bound of each algorithm in terms of both space and time is dependent on the term $size(X)$. However, for $X^K = T^K \times \text{NBA}(P^{[K]})$ it no longer holds that $size(X) \leq 2 \cdot [\![S^X]\!] \cdot \log_2([\![S^X]\!])$. Nevertheless, we can easily adjust the complexity bounds by considering the maximal out-degree of any state in the automaton. More precisely, have an NBA $B^G = (S^B, \xrightarrow{\mathcal{L}(V)}, I^B, A^B)$

Table 5.1: The complexity of computing an individual label for a single parametrization for a graph G. For a selection the TIME and the size of the output is multiplied by the size of the selection. In the table we use $X = T \times \mathrm{TBA}(P^G)$, $X' = T \times \mathrm{DBA}(P^G)$, to denote property encoded by a terminal and deterministic BA respectively, and $size(PI^G) = (size(\rho^G))^{size(\Omega^G)}$, as explained in Section 5.6.1.

$L(x)$	$TIME(L(x))$	$SPACE(L(x))$	$[\![L(x)]\!]$
$Bias(V^G)$	$[\![V^G]\!] \cdot size(\Omega^G)$	$size(\Omega^G)$	$[\![V^G]\!]$
$Cost(X)$	$size(X)$	$size(X)$	1
$Cost(X')$	$size(X') \cdot A^{X'}$	$size(X')$	1
$RF(V^G)$	$[\![V^G]\!] \cdot size(PI^G)$	$size(PI^G)$	$[\![V^G]\!] \cdot size(PI^G)$
$Impact(E^G)$	$[\![E^G]\!] \cdot size(\Omega^G)$	$size(\Omega^G)$	$[\![E^G]\!]$
$Indgeree(V^G)$	$[\![E^G]\!] \cdot size(\Omega^G)$	$size(\Omega^G)$	$[\![V^G]\!]$
$Robutness(X)$	$size(X)$	$size(X)$	1
$Robutness(X')$	$size(X') \cdot A^{X'}$	$size(X')$	1
$Sign(E^G)$	$[\![E^G]\!] \cdot size(\Omega^G)$	$size(\Omega^G)$	$[\![E^G]\!]$
$Trace(X)$	$size(X)$	$size(X)$	$size(X)$
$Trace(X')$	$size(X') \cdot A^{X'}$	$size(X')$	$size(X')$

and denote $out(b) = \{b' \mid \exists \phi \in \mathcal{L}(V^{[K]}) : b \xrightarrow{\phi} b'\}$. Then $size(X) \leq 2 \cdot [\![S^X]\!] \cdot \log_2([\![S^X]\!]) \cdot max\{[\![out(b)]\!] \mid b \in S^B\}$, which we can readily use in all the previous complexity statements. The second, more elusive, difference is in the nature of witness and robustness analysis. While we still are correct w.r.t. to the definitions of Section 2.2.4, we do not have any longer one-to-one correspondence between transitions in S^X and $S^{[K]}$. Consequently, different encodings of one property can yield different robustness. Unlike in the case of the encoding of the delta constraints, we can not place any guarantees on the relation between $robustness^K$ and $robustness^X$, which then fully depends on the structure on the automaton. For this problem we do not have any formal solution, and it should be taken into consideration by the user.

5.6 Labels and Reports

In this section we shortly discuss the computational aspects of the individual tools that are available in TREMPPI.

5.6.1 Storage

All the data are stored in a single SQLite3 table [OA10], where each parametrization and its labels occupies a single row. Each of the algorithms, be it for enumeration, labels, or reports is always manipulating only one row of the database at a time to assure low memory requirements.

A size of a single row is then limited by the maximum possible size for any label, as described by the Table 5.1.

A selection $\mathcal{K}^{\phi}$ for some ϕ is then evaluated as an SQlite3 query. From Section 3.4 we see that ϕ is only composed as a conjunction of linear constrains. Therefore for each parametrization we only have to evaluate as many inequalities as there are constraints, which is bound by the number of labels present. The trace and regulatory function are not allowed as constraints for a selection, therefore the TIME of evaluating a selection depends on the remaining labels times the size of the pool, *i.e.* belongs to $\mathcal{O}([\![\mathcal{K}^G]\!] \cdot \mathit{label\_count})$. The number of labels is either bound by the size of the parametrization or by the number of properties, whichever is bigger, therefore $\mathit{label\_count} \in \mathcal{O}(max(size(\Omega^G) \cdot [\![V^G]\!], [\![\overrightarrow{P}^{[K]}]\!]))$.

Note that due to optimizations made by the database [Hip15], the process of evaluating a selection is usually quite quick.

Computing Labels

The computation of the dynamic labels is evaluated in Section 5.5. For the static ones, it is in almost all cases as straightforward as evaluating the mathematical function they are expressing. The complexity of them then in most cases stems from the fact that all the parameter values need to be evaluated, and is therefore equal to the length of a partial parametrization.

The only exception to this is the regulatory function label. To obtain this label we progress by the following recipe, which corresponds to enumeration of prime implicants of the Quine-McCluskey algorithm [McC56]:

1. For $v \in V$, and for each $k \in [1, \rho(v)]$ enumerate all the configuration of regulators of v (states) that lead to the target value k.

2. For each k merge these configurations. A merge is possible if there are two configuration that match in all but one component. Create all possible merges.

3. Store those configurations that were not merged. For those that were obtained by merging, repeat the step 2.

4. After $[\![V]\!]$ repetitions we have the prime implicants for each k. These we write out in the Post Algebra form [MT07].

Note that a prime implicant is a combination of subsets of values for each component, therefore their number is bound by $(size(\rho^G))^{size(\Omega^G)}$. As this is

Table 5.2: The computational demands for each of the reports. As the reports depend on the labels, some of the reports require computation of the respective label prior to the computation (the Precondition column). Since the reports mostly only read and store the respective labels, the TIME, SPACE, and size of the result are equal to the size of the labels, as reflected in the Complexity column. The complexity is again given for a single parametrization and must be multiplied by the size of the selection. The symbol L is used to denote the set of all labels that can be read by the given report.

$Report(x)$	Precondition	Complexity
$Quantitative(L)$	none, but all quantitative labels will be used if available	$\llbracket L \rrbracket$
$Qualitative(L)$	none, but all qualitative labels will be used if available	$max\{\llbracket l^K \rrbracket \mid l^K \in L\} \cdot \llbracket L \rrbracket$
$Regulations(G)$	sign and impact	$\llbracket E^G \rrbracket$
$Correlations(G)$	bias	$\llbracket V^G \rrbracket^2$
$Witness(\overrightarrow{P}^G)$	trace for each $P \in \overrightarrow{P}^G$	$max\{cost^K(P) \mid P \in \overrightarrow{P}^G\} \cdot S^{[K]}$

potentially a very big number, the functions should be in general computed as the last step only for the selections that are of the real interest to the user.

Computing Reports

The computation of the individual reports is in most cases quite straightforward, as it usually only requires to sum the values of the individual labels.

The only exception is the correlation report, where to compute the corelation beween the bias values, we must first computate the mean and the standard deviation [Bar13] of bias of the individual components. For this purpose we conduct three passes over a database, in which we compute in succession:

1. the mean of the individual biases,
2. the standard deviation of the individual biases, using the obtained mean,
3. the covariance and subsequently the correlation of all pairs of biases.

Thus, for all the reports, the TIME, SPACE and the output size is therefore linear in the size of the input data, as shown in Table 5.2.

CHAPTER 6

Conclusion

To conclude this thesis we provide a comparison of TREMPPI to the existing tools, give a summary of the presented research, and finish with an informal evaluation of this work and its possible future.

6.1 Related Work

There exist multiple other tools allowing for enumeration or evaluation of parametrizations based on dynamical properties.

The most immediate comparison is to the LogicModelClassifier tool [Kla15]. This is a Python based tool which uses the state-of-the-art model checker NuSMV [CCG$^+$02] to evaluate each parametrization separately against an LTL or a *computational tree logic* (CTL) formula. These can be derived from a discretized dataset, in a fashion similar to the measurements encoding of TREMPPI. LogicModelClassifier offers a wider spectrum of enumeration constraints than TREMPPI and also provides access to a more extensive property descriptive language. Additionally, due to symbolic manipulation with the state space, the memory usage scales in the average case better with the size of a network. The most significant drawback is lower performance—the LogicModelClassifier was used for comparison in the EGFR study, exhibiting over a thousand times lower performance—which stems from the fact NuSMV does not expect to check many similar systems in a row and does not provide any ”warm-start” capabilities.

The probably oldest tool for evaluation of parameter families using CTL is called SMBioNet [KCRB09]. Like LogicModelClassifier, SMBioNet uses NuSMV for the model checking procedure, therefore the advantages and drawbacks are shared between the two. The tool is currently not available for download and is most likely discontinued. In addition, the CTL language is difficult to understand and use and the tool provides no visual interface, causing it to have a high entry threshold.

Antelope [AAA^{+}11] is a web-based tool that uses its own hybrid CTL model checker. This adds the capabilities to express features like stability or oscillation directly as parts of the formula, rather than encoding it into the CTL language using its operators. In a sense this is similar to the approach taken in this work, where we also define dynamical behaviour in terms of structure of a witness, rather than an encoding formula. In addition, this is the only tool that provides a web access and a visual interface, making it more accessible and being the only competitor of TREMPPI in this sense. Additionally, Antelope has the option to consider multiple parametrizations, but does so by overlaying the individual TS into a superstructure, without the ability to discern between the individual parametrizations.

The last MC tool we examine is called SPuTNIk [GMLGB14]. Like TREMPPI, SPuTNIk builds on LTL basics and deploys a custom model checker, however SPuTNIk in addition utilizes symbolic representation for performance gains. The method is quite promising, unfortunately the tool is not yet available and the authors [GMLGB14] do not evaluate the performance, therefore a comparison is unfortunately not possible.

All of the above are tools that depend on the model checking procedure, which sets rather fixed boundaries on the number of components a network can have due to both the time and the space complexity. A different possibility is to express the data in terms of formal constraints on a system. One of the tools in this category is called GNBox [CFT10]. GNBox depends on the *constraint logical programming* technique for expressing the regulatory constraints. GNBox exhibits similar performance to TREMPPI on the Bacteriophage Lambda model study [KSB12] and [CFT10]. GNBox also allows to evaluate properties from their structural perspective, *e.g.* for an unsatisfiable formula it is possible to ask, what is its maximal satisfiable sub-formula. We see such capability as certainly beneficial and it constitutes a possible path for future development of TREMPPI. GNBox itself is however most likely discontinued.

A similar approach is taken by Caspo [GVE^{+}13]. This tool depends on the *answer set programming* method which allows to enumerate models that fit certain dynamical constraints, like a time series. The fit here is given by an error between the measured data and a synchronous simulation of a model, which is simpler and less informative measure than the dynamical behaviour guaranteed by model checking. Under this scenario the tool however exhibits a better performance in terms of both the time and the space required by the method than TREMPPI. Recently [VGE^{+}15] the authors also removed the original limitation of the tool, which could only consider networks without feedback. With this improvement, Caspo became the probably most promising tool from all those discussed in this section.

From the above listed tools, Caspo and LogicModelClassifier are the only existing tools that attempt any classification or analysis of the parametrization pool after the model checking procedure. Caspo simply groups the models by the fit error, which is roughly equal to creating a selection for each qualitatively different value. LogicModelClassifier allows to draw several features from a parametrization selection, *e.g.* a parameter value or the most strict label for each regulation. To the best of our knowledge, the visual representation of the statistical features, as well as the report comparison workflow of TREMPPI are new features, previously not available in the field.

6.2 Summary

The aim of this work was to develop a new methodology for analysis of molecular pathways using top-down modelling approaches. To this end we have created a software under the name TREMPPI—Toolkit for Reverse Engineering of Molecular Pathways via Parameter Identification. This software is tailored to help with the task of construction of holistic biological models, and as we have demonstrated on distinct biological studies, it is sufficiently well-equipped and powerful to handle real-world applications. Throughout this thesis we have described numerous methods that constitute the building blocks of TREMPPI. When these methods are collected in a single tool, a modelling and evaluation workflow emerges, allowing for TREMPPI to pose as a completely stand-alone and easy-to-use tool in the field of systems biology, with no existing alternative. We therefore conclude that the goal of this work was fulfilled.

Each of the methods is examined in all the possible aspects—we always provide an intuitive description, formal formulation, an example, a demonstration on a real system, and an algorithm together with an evaluation of its performance. In this sense this thesis serves as complete and comprehensive documentation of the theoretical side of TREMPPI. The practical side—a usage manual—is present as an attachment to this work. The same holds also for the technical side: the code.

The tool, its documentation, and multiple binary distributions for various platforms are available under an open-source licence.

Apart from serving as a reference book for the TREMPPI tool, this thesis also examines two formal aspects of the qualitative model checking:

- We showed that in the multi-valued networks there is an equivalence relation on the dynamics of models. We showed that this is an issue directly related to this framework and describe a procedure for efficient evaluation of this equivalence.

- We showed that by changing the format of the BA-based model checking to the state-conditional Büchi automaton, we can achieve a direct correspondence between traces in a transition system and its product, which feature is otherwise not present in the standard procedure.

6.3 Outlook

It is our belief that TREMPPI offers a unique and valuable service to the researchers trying to untangle the immense complexity of biological systems. The tool was designed to provide a low enough threshold for those who already have knowledge of the biological side of the problem to be able to use it, however there is no doubt that further improvements can be made to the user experience. The most important next step is to make TREMPPI available as a web service, allowing for even an easier access, without a platform dependency, as we have already suggested in [SKSv13]. Since TREMPPI provides an HTML interface, this goal is already half achieved, however tasks like user and data management still need to be tackled.

From the perspective of scale, TREMPPI has only very limited application. As we have discussed, the problem at hand is double-exponential, making it very difficult to move beyond the simplest instances. Using TREMPPI for analysis of comprehensive models, *e.g.* modelling all genes of a single organism, or all the signalling molecules that can potentially interact, is far beyond its capabilities. Arguably though, models over a hundred of components are no longer human-readable and thus suggest that the interest of the modeller is in executing the model with a computer, rather than in making the model in order to understand the modelled system. TREMPPI still does not even reach the hundred component boundary with the limit being realistically around thirty-forty components. There are however many pressing problems in biology requesting attention of researchers, in systems with up to forty components, sometimes even with just a handful.

In short, TREMPPI does not promise to be the "Swiss knife" for reverse engineering of molecular pathways, but offers unparalleled features in many practical scenarios. It is our hope that it will establish itself as one of the tools in the toolbox of systems biologist, tackling many of the biological problems, both those that exist now and those to come.

Bibliography

[AAA$^+$11] Gustavo Arellano, Julián Argil, Eugenio Azpeitia, Mariana Benítez, Miguel A Carrillo, Pedro Arturo Góngora, David A Rosenblueth, Elena R Alvarez-Buylla, et al. "Antelope": a hybrid-logic model checker for branching-time Boolean GRN analysis. *BMC Bioinformatics*, 12(1):490, 2011.

[AJL$^+$08] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell.* Garland Science, 5th edition, November 2008.

[AJOK09] Wassim Abou-Jaoudé, Djomangan A Ouattara, and Marcelle Kaufman. From structure to dynamics: frequency tuning in the p53-Mdm2 network: I. logical approach. *Journal of theoretical biology*, 258(4):561–577, 2009.

[AJV09] Markus Arndt, Thomas Juffmann, and Vlatko Vedral. Quantum physics meets biology. *HFSP journal*, 3(6):386–400, 2009.

[Alo06] Uri Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits (Chapman & Hall/CRC Mathematical and Computational Biology).* Chapman and Hall/CRC, 1 edition, July 2006.

[ASRC$^+$10] Leonidas G Alexopoulos, Julio Saez-Rodriguez, Benjamin D Cosgrove, Douglas A Lauffenburger, and Peter K Sorger. Networks inferred from biochemical data reveal profound differences in toll-like receptor and inflammatory signaling between normal and transformed hepatocytes. *Molecular & Cellular Proteomics*, 9(9):1849–1865, 2010.

[Bar13] Michael Baron. *Probability and statistics for computer scientists.* CRC Press, 2013.

[BBK$^+$12] J. Barnat, L. Brim, A. Krejčí, A. Streck, D. Šafránek, M. Vejnár, and T. Vejpustek. On Parameter Synthesis by Parallel Model Checking. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(3):693–705, 2012.

[BCRG04] Gilles Bernot, Jean-Paul Comet, Adrien Richard, and Janine Guespin. Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347, August 2004.

[BHR84] P. A. Bloniarz, H. B. Hunt, III, and D. J. Rosenkrantz. Algebraic structures with hard equivalence and minimization problems. *Journal of the ACM*, 31(4):879–904, September 1984.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[Bra14] Tim Bray. The javascript object notation (json) data interchange format. `http://tools.ietf.org/html/rfc7159.html`, 2014. Accessed: 15/12/2015.

[CBBC11] Oana-Teodora Chis, Julio R. Banga, and Eva Balsa-Canto. Structural identifiability of systems biology models: A critical comparison of methods. *PLoS ONE*, 6(11), 11 2011.

[CCG$^+$02] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *Computer Aided Verification*, pages 241–268. Springer, 2002.

[CCP$^+$06] B. Christen, M. Christen, R. Paul, F. Schmid, M. Folcher, P. Jenoe, M. Meuwly, and U. Jenal. Allosteric Control of Cyclic di-GMP Signaling. *Journal of Biological Chemistry*, 281(42):32015–32024, October 2006.

[CFT10] Fabien Corblin, Eric Fanchon, and Laurent Trilling. Applications of a formal approach to decipher discrete genetic networks. *BMC Bioinformatics*, 11(1):385, 2010.

[CGP99] Edmund Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, 1999.

[dJ02] H. de Jong. Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[DLML10] Elena S Dimitrova, M Paola Vera Licona, John McGee, and Reinhard Laubenbacher. Discretization of time series data. *Journal of Computational Biology*, 17(6):853–868, 2010.

[DSM$^+$15] Lorenza A. D'Alessandro, Regina Samaga, Tim Maiwald, Seong-Hwan Rho, Sandra Bonefas, Andreas Raue, Nao

Iwamoto, Alexandra Kienast, Katharina Waldow, Rene Meyer, Marcel Schilling, Jens Timmer, Steffen Klamt, and Ursula Klingmüller. Disentangling the complexity of HGF signaling by combining qualitative and quantitative modeling. *PLoS Comput Biology*, 11(4):e1004192, 04 2015.

[eP03] Ivana Černá and Radek Pelánek. Relating hierarchy of temporal properties to model checking. In Branislav Rovan and Peter Vojtáš, editors, *Mathematical Foundations of Computer Science 2003*, volume 2747 of *Lecture Notes in Computer Science*, pages 318–327. Springer Berlin Heidelberg, 2003.

[Fla06] David Flanagan. *JavaScript: the definitive guide.* O'Reilly Media, Inc., 2006.

[GCC+15] Cristian A. Gallo, Rocio L. Cecchini, Jessica A. Carballido, Sandra Micheletto, and Ignacio Ponzoni. Discretization of gene expression data revised. *Briefings in Bioinformatics*, 2015.

[Ger02] Carlos Gershenson. Classification of random boolean networks. *CoRR*, cs.CC/0208001, 2002.

[GMLGB14] Emmanuelle Gallet, Matthieu Manceny, Pascale Le Gall, and Paolo Ballarini. An LTL model checking approach for biological parameter inference. In *Formal Methods and Software Engineering*, pages 155–170. Springer, 2014.

[GVE+13] Carito Guziolowski, Santiago Videla, Federica Eduati, Sven Thiele, Thomas Cokelaer, Anne Siegel, and Julio Saez-Rodriguez. Exhaustively characterizing feasible logic models of a signaling network using answer set programming. *Bioinformatics*, 2013.

[Hen09] Regine Hengge. Principles of c-di-gmp signalling in bacteria. *Nature Reviews Microbiology*, 7(4):263–273, April 2009.

[Hip15] Dwayne Richard Hipp. The SQLite query planner. `https://www.sqlite.org/optoverview.html`, 2015. Accessed: 2015-11-05.

[HR04] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and reasoning about systems.* Cambridge University Press, 2004.

[Ing13] Brian Ingalls. *Mathematical Modelling in Systems Biology: An Introduction.* MIT press, 2013.

[Kau69] S.A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, 1969.

[KCRB09] Zohra Khalis, Jean-Paul Comet, Adrien Richard, and Gilles Bernot. The SMBioNet method for discovering models of gene regulatory networks. *Genes, Genomes and Genomics*, 3(1):15–22, 2009.

[Kit04] Hiroaki Kitano. Biological robustness. *Nature Reviews Genetics*, 5(11):826–837, 2004.

[Kla15] Hannes Klarner. *Contributions to the Analysis of Qualitative Models of Regulatory Networks.* PhD thesis, Freie Universität Berlin, Germany, 2015.

[KS08] Guy Karlebach and Ron Shamir. Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9(10):770–780, 2008.

[KSB12] Hannes Klarner, Heike Siebert, and Alexander Bockmayr. Time series dependent analysis of unparametrized Thomas networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 99, 2012.

[KSFG$^+$13] Bertram Klinger, Anja Sieber, Raphaela Fritsche-Guenther, Franziska Witzel, Leanne Berry, Dirk Schumacher, Yibing Yan, Pawel Durek, Mark Merchant, Reinhold Schäfer, et al. Network quantification of egfr signaling unveils potential for targeted combination therapy. *Molecular systems biology*, 9(1), 2013.

[KSv$^+$12] Hannes Klarner, Adam Streck, David Šafránek, Juraj Kolčák, and Heike Siebert. Parameter identification and model ranking of Thomas networks. In *CMSB*, pages 207–226, 2012.

[LKPH13] Sandra Lindenberg, Gisela Klauck, Eberhard Pesavento, Christina andKlauck, and Regine Hengge. The EAL domain protein YciR acts as a trigger enzyme in a c-di-G signalling cascade in E. coli biofilm control. *The EMBO Journal*, advance online publication, May 2013.

[McC56] Edward J McCluskey. Minimization of Boolean functions*. *Bell system technical Journal*, 35(6):1417–1444, 1956.

[Mic05] Bénédicte Michel. After 30 years of study, the bacterial sos response still surprises us. *PLoS Biol*, 3(7):e255, 07 2005.

[MT07] D Michael Miller and Mitchell A Thornton. *Multiple valued logic: concepts and representations*, volume 2. Morgan & Claypool Publishers, 2007.

[Nob08] Denis Noble. *The Music of Life: Biology Beyond Genes*. Oxford University Press, USA, April 2008.

[OA10] Mike Owens and Grant Allen. *SQLite*. Apress, 2010.

[OAJK10] Djomangan A Ouattara, Wassim Abou-Jaoudé, and Marcelle Kaufman. From structure to dynamics: Frequency tuning in the p53-Mdm2 network. II: Differential and stochastic approaches. *Journal of theoretical biology*, 264(4):1177–1189, 2010.

[PH13] David A Patterson and John L Hennessy. *Computer organization and design: the hardware/software interface*. Newnes, 2013.

[PSRA$^+$11] Robert J Prill, Julio Saez-Rodriguez, Leonidas G Alexopoulos, Peter K Sorger, and Gustavo Stolovitzky. Crowdsourcing network inference: the DREAM predictive signaling network challenge. *Science signaling*, 4(189):mr7, 2011.

[RA09] John Ross and Adam P Arkin. Complex systems: from chemistry to systems biology. *Proceedings of the National Academy of Sciences*, 106(16):6433–6434, 2009.

[RCB06] Adrien Richard, JP Comet, and Gilles Bernot. Formal methods for modeling biological regulatory networks. *Modern Formal Methods and Applications*, pages 1–33, 2006.

[SAR13] Assieh Saadatpour, Réka Albert, and Timothy C Reluga. A reduction method for boolean network models proven to conserve attractors. *SIAM Journal on Applied Dynamical Systems*, 12(4):1997–2011, 2013.

[Sip96] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.

[SK04] Ilya Shmulevich and Stuart A Kauffman. Activities and sensitivities in Boolean network models. *Physical review letters*, 93(4):048701, 2004.

[Ski08] Steven S. Skiena. *The Algorithm Design Manual*. Springer Publishing Company, Incorporated, 2nd edition, 2008.

[SKSv13] Adam Streck, Juraj Kolčák, Heike Siebert, and David Šafránek. Esther: Introducing an online platform for parameter identification of boolean networks. In *Computational Methods in Systems Biology: 11th International Conference, CMSB 2013, Klosterneuburg, Austria, September 22-24, 2013, Proceedings*, volume 8130, page 257. Springer, 2013.

[Sno89] El Houssine Snoussi. Qualitative dynamics of piecewise-linear differential equations: a discrete mapping approach. *Dynamics and stability of Systems*, 4(3-4):565–583, 1989.

[SRK$^+$13] Diego O. Serra, Anja M. Richter, Gisela Klauck, Franziska Mika, and Regine Hengge. Microanatomy at cellular resolution and spatial order of physiological differentiation in a bacterial biofilm. *mBio*, 4(2), May 2013.

[SS15] Adam Streck and Heike Siebert. Extensions for LTL model checking of thomas networks. In *advances is Systems and Synthetic Biology*, volume 14, pages 101–114. EDP Sciences, 2015.

[STL10] Christian Schulte, Guido Tack, and Mikael Z Lagerkvist. Modeling and programming with gecode, 2010.

[Str97] B. Stroustrup. *The C++ programming language.* Addison-Wesley Longman Publishing Co., Inc., 1997.

[STS15a] Adam Streck, Kirsten Thobe, and Heike Siebert. Analysing cell line specific EGFR signalling via optimized automata based model checking. In *Computational Methods in Systems Biology*, volume 9308 of *Lecture Notes in Computer Science*, pages 264–276. Springer International Publishing, 2015.

[STS15b] Adam Streck, Kirsten Thobe, and Heike Siebert. Comparative statistical analysis of qualitative parametrization set. In *Hybrid Systems Biology*, volume 9271 of *Theoretical Computer Science and General Issues.* Springer International Publishing, 2015.

[TA90] Louis C. Thomas and Richard d' Ari. *Biological feedback.* CRC Press, Boca Raton, 1990.

[Tho91] René Thomas. Regulatory networks seen as asynchronous automata: A logical description. *Journal of Theoretical Biology*, 153(1):1–23, 1991.

[Tho13] R. Thomas. Remarks on the respective roles of logical parameters and time delays in asynchronous logic: An homage to El Houssine Snoussi. *Bulletin of Mathematical Biology*, 75(6):896–904, 2013.

[TJ09] Jean-François Tanti and Jennifer Jager. Cellular mechanisms of insulin resistance: role of stress-regulated serine kinases and insulin receptor substrates (IRS) serine phosphorylation. *Current opinion in pharmacology*, 9(6):753–762, 2009.

[TSKS14] Kirsten Thobe, Adam Streck, Hannes Klarner, and Heike Siebert. Model integration and crosstalk analysis of logical regulatory networks. In *Computational Methods in Systems Biology*, pages 32–44. Springer, 2014.

[Tur52] Alan Mathison Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 237(641):37–72, 1952.

[van95] Guido van Rossum. Python reference manual. Report CS-R9525, Centrum voor Wiskunde en Informatica, P. O. Box 4079, 1009 AB Amsterdam, The Netherlands, April 1995.

[VGE$^+$15] Santiago Videla, Carito Guziolowski, Federica Eduati, Sven Thiele, Martin Gebser, Jacques Nicolas, Julio Saez-Rodriguez, Torsten Schaub, and Anne Siegel. Learning Boolean logic models of signaling networks with {ASP}. *Theoretical Computer Science*, 599:79–101, 2015. Advances in Computational Methods in Systems Biology.

[Wol01] Olaf Wolkenhauer. Systems biology: The reincarnation of systems theory applied in biology? *Briefings in Bioinformatics*, 2(3):258, 2001.

[WPP$^+$06] H. Weber, C. Pesavento, A. Possling, G. Tischendorf, and R. Hengge. Cyclic-di-GMP-mediated signalling within the σ^S network of *Escherichia coli*. *Molecular Microbiology*, 2006.

[WSA12] Rui-Sheng Wang, Assieh Saadatpour, and Réka Albert. Boolean modeling in systems biology: an overview of methodology and applications. *Physical Biology*, 9(5), 2012.

[WVD12] Marian Walhout, Marc Vidal, and Job Dekker. *Handbook of Systems Biology.* Elsevier, 2012.

[YSS$^+$15] Kaveh P Yousef, Adam Streck, Christof Schütte, Heike Siebert, Regine Hengge, and Max von Kleist. Logical-continuous modelling of post-translationally regulated bistability of curli fiber expression in Escherichia coli. *BMC systems biology*, 9(1):39, 2015.

APPENDIX A

Nomenclature

symbol	**meaning**
$Vec_{i\leftarrow n}$	the vector Vec where the element with the index $i \in [1, \llbracket Vec \rrbracket]$ is replaced by the value n
$\mathbb{N}_0$	the natural numbers with 0
$\mathbb{N}_1$	$\mathbb{N}_0 \setminus \{0\}$
$\mathbb{P}(S)$	the power set of S
G	(V, E, ρ)—a regulatory graph
V	a set of components
E	a set of regulations s.t. $(u, t, v) \in E \implies u, v \in V \wedge t \in [1, \rho(u)]$
$\rho(v)$	the maximum activity level of $v \in V$
$\theta(u, v)$	$\{t \mid (u, t, v) \in E\}$—the threshold function
$\Theta(u, v)$	$\theta(u, v) \cup \{0, \rho(u) + 1\}$—the extended threshold function
$\succ^{\Theta}$	the ordinal successor in the set Θ
t_-, t_+	the closes lower and higher element of t in Θ
I_v^u	$\{[t, \succ^{\Theta}(t)) \mid t \in \theta(u, v) \cup \{0\}\}$—the set of activity intervals of u in regulations of v
Ω_v	$\prod_{u \in V} I_v^u$ the regulatory contexts of $v \in V$
$K_v(\omega)$	the parameter of v in the context $\omega \in \Omega_v$
K	$(K_v)_{v \in V}$—a parametrization
$[K]$	G—a network that is parametrizable by K
$\mathcal{K}$	a set of parametrizations
S^G	$\prod_{v \in V^G} [0, \rho(v)]$—the state space of G
$F_v^K(s)$	update value for the component v in the state s under parametrization K

$\delta(F_v^K)(s)$	$F_v^K(s) - s_v$—a discrete partial derivative of F_v^K in s
T^K	$(S^{[K]}, \rightarrow^T)$— a transition system
$\rightarrow^T$	the transition relation in T
$\mathcal{T}^{[K]}$	the set of asynchronous transitions systems over the state space $S^{[K]}$
B^G	an automaton encoding a property for a network G
S^B	a set of BA states
$\xrightarrow{\mathcal{L}(G)}$	a transition relation with propositions
$\mathcal{L}(K)$	$\mathbb{P}(\{v * n \mid v \in V^G, * \in \{\leq, \geq, <, >, =\}, n \in [0, \rho(v)]\})$—the set of propositions for the network G
I^B	a set of initial states in S^B
A^B	a set of accepting states in S^B
X^K	$T^K \times B^{[K]}$—a product structure
W^X	$\{w \in (\rightarrow^X)^n \mid n \in \mathbb{N}_0, (X^K, w \models P^{[K]})\}$—the set of witnesses of $P^{[K]}$ in X^K
TBA,DBA,NBA	terminal, deterministic and non-deterministic Büchi automaton
$cost^X$	the length of some shortest w in W^X
$succ^X(s)$	$\{s' \mid s \rightarrow^X s'\}$—the successor function in X
$robustness^X$	the probability of finding a witness by taking a random walk in X^K of the length exactly equal to $cost^X$
$\uparrow(e)$	true *iff* e is activating
$\downarrow(e)$	true *iff* e is inhibiting
$\lambda(e)$	an regulation constraint for e
$\gamma(v)$	a parameter constraint on K_v
$\eta(v)$	the normalization constraint on K_v
P^G	$(\overrightarrow{M}^G, \overrightarrow{D}^G, End^G, Exp^G)$ a property of G
$\overrightarrow{M}^G$	$(M^1, \ldots, M^k)$—a sequence of measurements
M_v^G	a measurement, i.e. a sub-range in $[0, \rho(v)]$
$\overrightarrow{D}^G$	$(D^1, \ldots, D^k)$—a sequence of delta constraints
D_v^G	a delta constraint, one of $\{up, down, stay, none\}$
End^G	is an ending, either *open, stable,* or *cyclic*
Exp^G	is an experiment, which forms a rectangular subset of S^G
$I(\overrightarrow{M}^G, w^T)$	a vector of indices mapping some states the trace w^T to each of the measurements in $\overrightarrow{M}^G$, if such exists

$Reduce^G(K, Exp^{[K]})$	is a parametrization K that has been reduced by the experiment $Exp^{[K]}$
l^K	a label of parametrization K
$sign^K(e)$	an edge label, one of $\{0, +, -, 1\}$
$indegree^K(v)$	a number of observable regulators of v
$impact^K(e)$	the impact of the regulation e on its target
$bias^K(v)$	the tendency of the components v towards higher or lower values
$RF^K(v)$	a Post Algebra expression describing the partial parametrization of the component v
$qual(\mathcal{K}^\Phi, l, y)$	a qualitative report
$quan(\mathcal{K}^\Phi, l, y)$	a quantitative report
$frequency^{\mathcal{K}^\Phi}(e)$	how often is e observable in the selection
$correlation^{\mathcal{K}^\Phi}(v, u)$	Pearson correlation coefficient between biases of u and v in the selection
$trace^K(P^{[K]})$	transitions of the shortes witness of $P^{[K]}$ in T^K
$\overrightarrow{P}^G$	$(M^1, \ldots, M^k)$—a sequence of properties
$witness^K(\overrightarrow{P}^{[K]})$	a witness graph
$size(\rho^G)$	the maximal number of activity levels of any component
$size(\Omega^G)$	the maximal number of regulator contexts
$size(K)$	the maximal size of a single parametrization
$size(\mathcal{K}^G)$	the maximal size of the parametrization space
$Can(K)$	canonized form of the parametrization K
$\mathcal{C}^G$	the set of canonical parametrizations of G
$Comp(K)$	K with a single additional non-observable edge
$sgn(n)$	the sign of n, one of $\{-1, 0, +1\}$
$(F_v^K)^{mon}(s)$	the monotone target value of v under K in the state s
$Norm(K, v, \omega)$	the normalized parameter value of v from K in the context ω
$Norm(K)$	the normalized form of the parametrization K
$\mathcal{N}^G$	the set of normalized parametrizations of G
$Minim(K)$	K with a single non-observable edge removed
$size(X)$	the maximal size of a transition system X, in the number of transitions

APPENDIX B

Supplementary Files

There are multiple supplementary files supporting the results of this thesis. All of these are available on the medium that is attached to the thesis and on the web. The specific locations are detailed below.

B.1 TREMPPI

For the TREMPPI tool we are providing documentation, open-source, and binary distributions.

- **documentation:** is available as a single *HTML* page called `documentation.html` or at `http://dibimath.github.io/TREMPPI/`.
- **source:** is available in the folder `source` or in the on-line repository `https://github.com/xstreck1/TREMPPI`.
- **binaries:** are available for multiple platforms, each in a single archive. The local archives are stored in the folder `build`. The supported platforms are:
  - **Windows-32/64 bit:** file: `Windows_32bit.zip`, URL: `http://dibimath.github.io/TREMPPI/build/TREMPPI_Windows32.zip`
  - **Ubuntu-32/64 bit:** file: `Ubuntu_32bit.zip`, URL: `http://dibimath.github.io/TREMPPI/build/TREMPPI_Debian64.tar.gz`
  - **Debian-64 bit only:** file: `Debian_64bit.zip`, URL: `http://dibimath.github.io/TREMPPI/build/TREMPPI_Ubuntu32.tar.gz`

B.2 Understanding Case Studies

In the following sections we list the data for the individual case studies. These are always available for viewing either at the medium attached to the thesis in the folder `projects`, or on-line. Details are provided individually in each section.

Data for each of the case studies are stored as a TREMPPI project. This means that all the data are presented via respective HTML pages. The content of the individual pages is quite intuitive to understand. If needed, please refer to the documentation of TREMPPI, linked above.

To view any of the projects there are four options:

1. Navigate to the respective web address (recommended).

2. Copy the data from the medium and extract the build fitting your system. Then execute the binary called `tremppi`, which will load all the projects into TREMPPI.

3. Navigate to the `projects` folder on the medium and start a local server, for example by calling `python -m SimpleHTTPServer 8080`, if Python 2 is available, or `python -m http.server 8080`, if Python 3 is available.

4. Navigate to the directory of the specific project and open the required web-page directly in a browser. This way is discouraged, as some browsers may refuse to load data correctly due to security reasons.

B.3 Toy Network Data

First we present data for the running example in Figure 2.1, which provide a concise showcase for all the methods in this thesis. The project is available either at the URL: `http://dibimath.github.io/Toy_example` or in the folder `Toy_example` on the attached medium.

The network is constructed based on the description in Figure 2.1, with a single property as shown in Figure 3.1. There are three selections:

- *all:* is the set of all the 324 non-normalized parametrizations of the network. This set is included for illustration and cannot be produced by TREMPPI without disabling the normalization procedure.

- *normalized:* is the set of all the 144 normalized parametrizations of the network.

- *single_parametrization:* is the one specific parametrization selected for illustration of labels, as shown in Figure 3.2.

- *selected:* is the set of parametrizations used for comparison in Figure 3.3.

B.4 EGFR Data

The supporting data for the study in Section 4.1 are available either in the folder `EGFR` or at `http://dibimath.github.io/EGFR/`.

Altogether there are 160 properties we have used for model checking. The name of each property contains the cell line, the intervention strategy and possible letters `m` for monotonicity constraints and `s` for stability constraint. Recall that this study was mainly focused on the performance of the MC procedure, therefore the only reports available are the *qualitative* and *quantitative* comparison for each cell line and any of the 5 different types of selection from Figure 4.1c.

B.5 HGF Data

The data for the study in Section 4.2 are available either in the folder `HGF` or at `http://dibimath.github.io/HGF/`.

For this case study we have constructed a network with 5 properties as described in Section 4.2.1. Based on these properties we made three selections as detailed in Section 4.2.2. For each of these selections we have then constructed *qualitative, quantitative, regulations*, and *correlations* reports.

B.6 E. Coli Data

The last study—*Escherichia coli* biofilm production—is available either at the URL `http://dibimath.github.io/Ecoli_biofilm/` or in the folder `Ecoli_biofilm`.

For this study we have constructed the network as explained in Section 4.3.2, and derived 32 properties, as explained in Section 4.3.3. For these 32 there is the *relaxed* selection of only 15 of them and the *strict* selection with all the 32. We have created all the reports for this study, however only the regulatory functions are of interest for the analysis in Section 4.3.5, which are available in the *qualitative* report.

Erklärung zur Dissertation

Hiermit erkläre ich, dass ich alle Hilfsmittel und Hilfen angegeben habe und ich versiche, dass ich auf dieser Grundlage die Arbeit selbstständig verfasst habe. Die Arbeit wurde in einem früheren Promotionsverfahren nicht eingereicht.

Hereby I declare that I have listed all auxiliary means used and I assure that I have written this thesis myself. The thesis has not been submitted before.

Berlin, 11.4.2016

Adam Streck