Algorithmic Reduction of Biochemical Reaction Networks

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)

Vorgelegt im Fachbereich Mathematik und Naturwissenschaften
der Universität Kassel

von

Christoph Lüders

aus Sankt Augustin

Kassel, 25. Februar 2022

Tag der Disputation:

02. Juni 2022

Betreuer:

Prof. Dr. Werner M. Seiler
PD Dr. Thomas Sturm
Prof. Dr. Sebastian Walcher
Prof. Dr. Andreas Weber †

Erstgutachter:

Prof. Dr. Werner M. Seiler
Universität Kassel

Zweitgutachter:

PD Dr. Thomas Sturm
CNRS, Frankreich

For Andreas

Summary

The dynamics of species concentrations of chemical reaction networks are given by autonomous first-order ordinary differential equations. Singular perturbation methods allow the computation of approximate reduced systems that make explicit several time scales with corresponding invariant manifolds. This thesis presents:

1. An algorithmic approach for the computation of such reductions on solid analytical grounds. Required scalings are derived using tropical geometry. The existence of invariant manifolds is subject to certain hyperbolicity conditions. These conditions are reduced to Hurwitz criteria and discrete combinatorial conditions on degrees, which are technically solved using SMT over nonlinear real and linear integer arithmetic, respectively. The approach is implemented in Python and applied to a large body of known biochemical models.

2. ODEbase, a repository of biochemical models that has been created to provide real-world input data in a form that is suitable for symbolic computation software. ODEbase has been populated with semi-automatic conversions of several hundred SBML models from the BioModels database and is available to everyone at `https://odebase.org`.

3. A calculus for model-driven computation of disjunctive normal forms of real constraints from conjunctions of such disjunctive normal forms, which is required for the tropicalization in 1. The calculus technically once more builds on SMT solving, here over linear real arithmetic. Compared to existing software like Redlog, its implementation generally shows significant speedups, and a number of otherwise infeasible computations finish within seconds.

Contents

List of Figures

List of Tables

List of Algorithms

Symbols & Notation

$A \subseteq B$	$\forall a(a \in A \longrightarrow a \in B)$
$A \subset B$	$A \subseteq B \wedge B \nsubseteq A$
$\mathbb{N}$	Set of nonnegative integers: $\{0, 1, 2, \ldots\}$
$\mathbb{Z}$	Ring of all integers: $\{0, \pm 1, \pm 2, \ldots\}$
$\mathbb{R}$	Field of real numbers
$(0, \infty)$	$\{\, x \in \mathbb{R} \mid x > 0 \,\}$
$[0, \infty)$	$\{\, x \in \mathbb{R} \mid x \geq 0 \,\}$
$[a : b]$	$\{\, x \in \mathbb{Z} \mid a \leq x \leq b \,\}$, a, $b \in \mathbb{Z}$
$g(n) \in O(f(n))$	$\exists c \geq 0 \exists n_0 > 0 \forall n \geq n_0 : \lvert g(n) \rvert \leq c \cdot f(n)$
$g(n) \in \Theta(f(n))$	$\exists c_1 \geq 0 \exists c_2 \geq 0 \exists n_0 > 0 \forall n \geq n_0 : c_1 f(n) \leq \lvert g(n) \rvert \leq c_2 f(n)$
$\dot{x}$	$\mathrm{d}x/\mathrm{d}t$
x'	$\mathrm{d}x/\mathrm{d}\tau$
$\langle x, y \rangle$	Inner product, $\sum_{i=1}^{n} x_i y_i$, x, $y \in \mathbb{R}^n$
$D_{x_1,\ldots,x_s} f$	$\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_s} \\ \multicolumn{3}{c}{\dotfill} \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_s} \end{pmatrix}$
y^c	$y_i^{c_i} \cdots y_n^{c_n}$, $y \in \mathbb{R}^n$, $c \in [0, \infty)^n$
“term”	A product of powers of variables
“monomial”	A term times a coefficient

Preface

The evolution in time of the species concentrations of many *chemical reaction networks* can be modeled and analyzed with autonomous first-order *ordinary differential equations*. Such systems of differential equations can answer questions analytically about the qualitative behavior of the described system, like stability, periodicity, or multistationarity. However, the analysis of high-dimensional differential equations is often computationally infeasible. Through *model reduction* one aims to find models that—in certain regions of the phase space—behave approximately like the original model, yet are smaller, which makes them easier to handle. *Singular perturbation methods* have proven to be an effective tool to find such reduced systems. However, systems of differential equations must be already partitioned into several *time scales* to be accessible to singular perturbation methods. Time scales are a coarse-grained measure of the speed at which differential variables change. For each time scale, variables in even faster time scales are assumed to be already in equilibrium and variables in slower time scales are assumed to be constant. Building on mathematical theory by Tikhonov and Fenichel for two time scales, recent advances by Cardin and Teixeira allow to expand this to three and more time scales.

The algorithmic approach described in this thesis rests on rigorous mathematical methods to compute invariant manifolds and corresponding reduced systems. At the same time, this approach provides a thorough algorithmic description for computations of such reductions in practice. The existence of invariant manifolds is subject to hyperbolicity conditions, for which an algorithmic test based on Hurwitz criteria is used. Verification of the Hurwitz criteria and a technical issue of verification of smoothness are accomplished with the help of Satisfiability Modulo Theories (SMT) solving.

The algorithms are implemented and tested against a large body of known biochemical models. For that purpose, ODEbase, a repository of biochemical models, has been created and filled with 662 models from the curated branch of the BioModels database. Furthermore, a Web front-end has been built to allow free public access to the repository.

To find appropriate scalings for singular perturbation theory, methods from *tropical geometry* are used. Since the pertinent computations of *tropical equilibrations* can be very time-consuming, special attention is given to efficiency. This leads to a novel method to compute a DNF of quantifier-free first-order formulas over the reals from an input of a conjunction of DNFs. This method is also based on SMT solving. It is implemented, tested, and benchmarked with all appropriate models from ODEbase. Also, it is compared against Redlog, the only established software that can perform this operation.

The original contribution of this thesis is, first and foremost, the algorithmic formulation of the scaling and reduction process. Furthermore, the ODEbase repository for biochemical systems of differential equations contains original work, since the systems need to be substantially pre-processed from their original BioModels sources. Lastly, the novel method to construct a DNF with the help of SMT solving is an original contribution.

This thesis is divided into seven chapters. The first chapter introduces chemical reaction networks and illustrates how their dynamics can be described by ordinary differential equations (ODEs). We explain mass action kinetics and provide examples of an ODE system describing a chemical reaction network with mass action kinetics. Some basic properties of ODEs are mentioned and singular perturbation theory in the sense of Tikhonov and Fenichel is illustrated for two time scales. For systems with polynomial vector fields, we state a general procedure for scaling that is the mathematical basis of our reductions. Lastly, we introduce tropical equilibrations which are used to provide exponent vectors required for the scaling procedure.

Chapter 2 explains singular perturbation methods for three and more time scales based on the theory of Cardin and Teixeira. A check for hyperbolic attractivity is formulated that, if affirmative, ensures that invariant manifolds exist. The theory of multiple invariant manifolds that are nested, each with a corresponding reduced system, is laid out. For the scaling of variables, we use tropical equilibrations, that is, the idea of the equilibration of dominant monomials. A rigorous mathematical connection from scaled systems to the theory of Cardin–Teixeira is provided.

In section 2.3, we present a precise formulation of algorithms for all steps from scaling to the computation of reduced systems, except the computation of the DNF that is discussed later in a separate chapter. Included is, among others, the verification of hyperbolic attractivity, a sufficient test for smoothness required by Cardin–Teixeira, and the computation of the tropical equilibration. Furthermore, algorithms for the simplification of the reduced systems and the defining equations of invariant manifolds as well as the back-transformation of the reduced systems are presented.

Chapter 3 contains a model-driven method to compute a DNF of quantifier-free first-order formulas from a conjunction of DNFs. The computation of a single DNF is an intermediate step needed to compute an explicit tropical equilibration. Previously, this computation was often very time-consuming, therefore a faster method was desirable. The new method is derived in the form of a calculus, complete with proofs of termination, soundness, and minimality. Additionally, the new method gives rise to an algorithm.

In Chapter 4, we present ODEbase, a repository of ODE systems for systems biology. We explain why commonly used formulations of models in SBML are not sufficient for symbolic computation and how SBML-formulated models are converted to become the content of ODEbase. Information about ODEbase data sets are outlined, and a freely available Web service is introduced, illustrated by several screenshots.

The calculus from Chapter 3 has been prototypically implemented in Python and is called SMTcut. Chapter 5 contains benchmarks of this implementation on suitable models from ODEbase from Chapter 4. This is compared against benchmarks done with Redlog.

In Chapter 6 we present several examples of reduced systems with details of the reduction and verification procedure. These systems have been reduced with a prototypical Python implementation of our algorithms as described in earlier chapters. We illustrate this with plots of the corresponding direction fields and a closer look at systems where the reduction procedure stopped early. Finally, Chapter 7 contains our conclusion and an outlook on possible future work.

Acknowledgements

First of all, I am grateful to my late advisor Andreas Weber at the University of Bonn, who sadly passed away in March 2020. His friendly and appreciative manner was most supportive and motivating for me while at the same time his scientific enthusiasm for tropical methods eventually gripped me as well. I was thankful and relieved to find that his colleagues and fellow participants in the SYMBIONT project Werner M. Seiler, Thomas Sturm, and Sebastian Walcher would divide among them the task of advising me for this thesis following Andreas' sudden death.

I am especially grateful to Thomas Sturm, with whom I had many long discussions about science, the universe, and everything. His guidance in general and his attitude towards mathematical rigor, scientific conduct, and linguistic subtleties have influenced me profoundly. I am also very grateful to Werner M. Seiler, who supported my project F_{22} and allowed me to find a new alma mater at the University of Kassel. Furthermore, I thank Sebastian Walcher for discussions and advice concerning the mathematical foundations of this thesis and for his encouraging words. Lastly, I want to express my thanks to Ovidiu Radulescu for his biological insight and discussions about tropicalization.

This thesis lies at a connection point of mathematics, computer science, and biology and this fact is mirrored in the areas of expertise of my advisors with Werner M. Seiler and Sebastian Walcher as mathematicians, Thomas Sturm as computer scientist, and Ovidiu Radulescu working in systems biology. The interdisciplinary character of my work and this thesis was challenging and at the same time inspiring and rewarding for me.

Over the course of the past seven years, there were several people that have helped me, each in their own way. Satya Swarup Samal was my predecessor as a doctoral student of Andreas Weber and I owe him gratitude for many discussions and explanations. I thank Jörg Zimmermann for discussions at the whiteboard, in the canteen, and in our office about science and more. I am thankful to Dima Grigoriev for his help with mathematical insight. Hassan Errami, Reinhard Klein, Matthias Neidhardt, and Anna Meschede were colleague, professor, and students, respectively, from the University of Bonn and helped me in various stages of my work. Ariadne Ullner provided help with proofreading. I also thank the other SYMBIONT project members that I exchanged ideas and had discussions with.

I also want to thank my business partner Martin Winkler for supporting my academic work and allowing me to take so much time off. Maybe some of what I have learnt is useful in our joint business context as well.

Last but not least, I thank my family and especially my partner Burçak Braukmann for their patience and support of my long late hours in front of the screen. Not to forget our cat Lucy who slept through almost all of this.

Chapter 1

Introduction

This thesis deals with analytical reductions of systems of ordinary differential equations specifying species concentrations for biological and chemical reaction networks. In this chapter, we will introduce several key concepts and recall a number of facts that will be used later on.

We start by introducing *(bio-)chemical reaction networks*, which form the foundation of our investigations. The mathematical modeling and analysis of chemical reaction networks involves *ordinary differential equations* (ODEs) as a potent tool. After recalling some fundamental facts about ODEs and their qualitative theory, we present a short survey of *mass action kinetics*, which governs the dynamics of many chemical reaction networks.

Then, we take a first look at the classical results of *singular perturbation theory* due to Tikhonov and Fenichel, which yields reductions of systems that are partitioned into two time scales.

Finally, we discuss the preparation of systems for the application of singular perturbation theory with three and more time scales. To this end, we introduce scalings and discuss some of their general properties. We close the chapter by taking a first look at *tropical equilibrations*, which allow us to determine appropriate scalings.

1.1 Chemical Reaction Networks

Chemical reaction networks (CRNs) describe the simultaneous interaction of chemical species. A *species* is a class of (bio-)chemical entities, e.g., atoms or molecules, whose members are chemically identical. We use the words “chemical” and “biochemical” interchangeably, but are particularly interested in biochemistry.

The interaction of chemical species of a specific reaction network with one another can be described by *reactions*. They describe how to convert input species, called *reactants*, to output species, called *products*. Names, which for simplicity are often letters, are assigned to all species, and with their help reactions can be written down easily. It might look like this:

$$\mathrm{CH_4} + 2\mathrm{O_2} \longrightarrow \mathrm{CO_2} + 2\mathrm{H_2O}, \tag{1.1}$$

or even only

$$2\mathrm{A} + \mathrm{B} \rightleftharpoons 3\mathrm{C}, \tag{1.2}$$

where species A, B, and C might remain anonymous. Equation (1.1) is an *irreversible reaction*, whereas (1.2) is a *reversible reaction.* The double-arrow is a shorthand for two separate one-way reactions. The object at each side of the arrows is called a *complex*, that is to say, a complex is a sum of species.

We assume that all chemical reactions take place in a well-stirred reactor where temperature, volume and other thermodynamic parameters are constant. This is a simplification from what happens in nature, but is approximately true for most biochemical reactions. Furthermore, it allows a convenient mathematical formulation of the dynamics by ordinary differential equations, which we introduce in the next section.

The number of molecules of a certain species in a complex is called their *stoichiometric coefficient.* On the left side of (1.2), the stoichiometric coefficients of A, B, and C in "$2\,\mathrm{A} + \mathrm{B}$" are 2, 1, and 0, respectively.

Chemical reaction networks consist of multiple reactions happening simultaneously with products of one reaction being possible reactants of another. Some networks approach a *stable state*, that is, the molar concentrations of all species approach constant values as time progresses. Yet, other networks show different evolutions. There are many possible ways the dynamics of a particular reaction network may evolve over time: there might be a steady state which is unstable, there may be multiple steady states, there might be oscillation between multiple states, or the dynamics could be chaotic altogether. Our goal is to come to an understanding about the evolution of networks by analytical methods. But first, we have to find a way to describe a network's dynamics.

1.2 Ordinary Differential Equations

The dynamics of species concentrations for chemical reaction networks can be described by systems of autonomous explicit first-order *ordinary differential equations* (ODEs), which we now briefly introduce. Informally speaking, an ODE determines the time evolution of a quantity where the velocity of change is determined by its momentary state. Note that we often use Newton's dot notation to represent a time derivative for variable t, that is

$$\dot{x} := \frac{\mathrm{d}x}{\mathrm{d}t}.$$

Definition 1.1. *Let $U \subseteq \mathbb{R}^n$ be open and non-empty, and $f : U \to \mathbb{R}^n$ continuously differentiable. Then one calls*

$$\dot{x}(t) = f(x(t)) \tag{1.3}$$

an autonomous ordinary differential equation of first order*. A solution φ of this differential equation is a differentiable function $\varphi : J \to U$, with $J \subseteq \mathbb{R}$ a nondegenerate interval, such that*

$$\dot{\varphi}(t) = f(\varphi(t))$$

for all $t \in J$.

If furthermore $\varphi(0) = y \in U$, then one calls φ a solution of the initial value problem

$$\dot{x}(t) = f(x(t)), \quad x(0) = y. \tag{1.4}$$

To remind the reader, a nondegenerate interval contains more than a single element. Also notice that the "$x(t)$" in (1.3) is just a placeholder for an actual function.

Example 1.1. *As a simple example for a differential equation we consider decay, which happens in chemical reactions as well as in physics and many other fields. Assume that the velocity of decay of quantity $N(t)$ is proportional to its abundance. This leads to the differential equation*

$$\dot{N}(t) = -\lambda N(t), \tag{1.5}$$

where $\lambda > 0$ is the proportionality constant. The minus sign signifies a decrease of the quantity. To solve this equation, we look for a function $N(t) : \mathbb{R} \to \mathbb{R}$ that solves (1.5)*. We find that $N(t) = N_0\, e^{-\lambda t}$ works fine, as the reader can easily convince himself of. The newly introduced variable N_0 has to be determined by the initial condition $N(0) = N_0$.* $\triangle$

This is an example of a *first-order autonomous* ordinary differential equation, where "first order" refers to the order of the derivative. The adjective "ordinary" implies that there is only one so-called *independent variable*, namely t, in contrast to partial differential equations that contain multiple independent variables, and the modifier "autonomous" means that the independent variable does not explicitly occur on the right hand side of the differential equation.

To properly solve autonomous ordinary first-order differential equations, we must be sure that solutions exist at all and that they are unique. The following theorem makes sure that this is the case. For a proof, see Amann [1].

Theorem 1.1 (Existence and uniqueness, and dependence on initial values)**.** *Let U, f as in Definition 1.1, and $y \in U$.*

1. *Then the initial value problem given by* (1.4) *has a unique solution $F(t, y)$ on an interval $I_{\max}(y) \in \mathbb{R}$, where the solution cannot be extended beyond this interval.*
2. *Furthermore, the set*

$$\widetilde{U} := \bigcup_{y \in U} \left(I_{\max}(y) \times \{y\}\right) \subseteq \mathbb{R} \times U$$

is open and the map

$$F : \widetilde{U} \to U, \quad (t, y) \mapsto F(t, y)$$

is continuously differentiable. If f is a $\mathcal{C}^r$-function with $r > 1$, then the same holds for F.

Since we frequently deal with parameter dependent differential equations, the following theorem provides existence and uniqueness results for those.

Theorem 1.2 (Dependence on parameters)**.** *Let $V \subseteq \mathbb{R}^n \times \mathbb{R}^m$ be open and non-empty, $h : V \to \mathbb{R}^n$ continuously differentiable, $(y, p) \in V$, and $H(t, y, p)$ a solution to*

$$\dot{x}(t) = h(x(t), p) \quad \textit{with} \quad x(0) = y.$$

Then H is defined on an open subset of $\mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m$ and is continuously differentiable. If h is a $\mathcal{C}^r$-function with $r > 1$, then the same holds for H.

Furthermore, the following theorem provides conditions for a solution $F(t, y)$ for given y to exist for all $t \in [0, \infty)$. That is, solutions persist for all positive times.

Theorem 1.3. *Let U, f as in Definition 1.1, $y \in U$, and $F(t, y)$ a solution to the initial value problem* (1.4) *on the interval $I_{\max}(y) \subseteq \mathbb{R}$. If there exists a closed and bounded set $K \subsetneq U$ such that*

$$F(t, y) \in K \text{ for all } t \in I_{\max}(y),\ t \geq 0$$

then it holds that

$$I_{\max}(y) \cap [0, \infty) = [0, \infty).$$

The following definition introduces some notions we will later need.

Definition 1.2. *Let U, f, F as in Theorem 1.3, $Y \subset U$, $F(t, y)$ a solution to the initial value problem* (1.4) *on the interval $I_{\max}(y) \subseteq \mathbb{R}$. Then*

(i) Y is called positively invariant *for* (1.4)*, if for each $y \in Y$ and all $t \in I_{\max}(y) \cap [0, \infty)$ it holds that $F(t, y) \in Y$.*

(ii) Y is called invariant *for* (1.4)*, if for each $y \in Y$ and all $t \in I_{\max}(y)$ it holds that $F(t, y) \in Y$.*

If Y is also a submanifold of $\mathbb{R}^n$, then Y is called a positively invariant manifold *or* invariant manifold*, respectively.*

We remind ourselves here that, informally speaking, a manifold is a topological space that locally resembles Euclidean space near each point. That is, around every point is a neighborhood that is topologically the same as the unit ball in $\mathbb{R}^n$.

1.3 Mass Action and Related Kinetics

Now, we define some entities that we already informally talked about. We follow Feinberg [30, Definition 3.1.1] with the following formal definition.

Definition 1.3. *A* chemical reaction network *consists of three sets:*

- *A finite set $\mathcal{S}$ of* species *with $n = |\mathcal{S}|$.*
- *A finite set $\mathcal{C} \subset [0, \infty)^n$ of vectors, called* complexes.
- *A finite set $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$, called* reactions.

We assume a fixed, yet arbitrary ordering of species. Molar concentrations will usually be called x_1, ..., x_n and stoichiometric coefficients for species in a complex c_1, ..., c_n. Furthermore, if we want to convey the name of the species, we write [A] to denote the molar concentration in time of species A.

For a complex $c = (c_1, \ldots, c_n)$ and molar concentrations $x = (x_1, \ldots, x_n)$, we abbreviate

$$x^c := x_1^{c_1} \cdots x_n^{c_n}.$$

If in the above equation both x_i and c_i are 0, we use the convention $0^0 = 1$.

As has already been stated, the dynamics of species concentrations of a chemical reaction network, given the conditions in Section 1.1, can be described by ordinary differential equations. They describe the *change* of each species' concentration over time. To actually write down the dynamics, we still need a *rate function* that determines the speed of the reaction. Call this function $\mathcal{K}_{c \to c'}(\cdot)$ for each reaction in question, where the index signifies reaction $c \to c'$, with c, $c' \in \mathcal{C}$. The system of all $\mathcal{K}_{c \to c'}(\cdot)$ is called a *kinetics*. The rate function takes as parameter the vector of molar concentrations, here usually called x. It is understood that those concentrations cannot be negative.

Consider the forward reaction of (1.2):

$$2\mathrm{A} + \mathrm{B} \longrightarrow 3\mathrm{C}. \tag{1.6}$$

Each time this reaction takes place, two molecules of A and one molecule of B are *lost* and three molecules of C are *gained*. Hence, the induced system of differential equations looks like this:

$$\begin{aligned}
[\dot{\mathrm{A}}] &= -2\mathcal{K}_{2\,\mathrm{A}+\mathrm{B}\to 3\,\mathrm{C}}(x)\\
[\dot{\mathrm{B}}] &= -\mathcal{K}_{2\,\mathrm{A}+\mathrm{B}\to 3\,\mathrm{C}}(x)\\
[\dot{\mathrm{C}}] &= 3\mathcal{K}_{2\,\mathrm{A}+\mathrm{B}\to 3\,\mathrm{C}}(x).
\end{aligned}$$

The fact that the stoichiometric coefficients reappear on the right hand side of the differential equations is due to our choice of *molar* concentrations.

Example 1.2. *Consider this slightly more interesting reaction network [70]:*

$$\mathrm{E} + \mathrm{S} \rightleftharpoons \mathrm{C} \longrightarrow \mathrm{E} + \mathrm{P}.$$

It describes three reactions taking place. The differential equations that are induced by this chemical reaction network are:

$$\begin{aligned}
[\dot{\mathrm{E}}] &= -\mathcal{K}_{\mathrm{E}+\mathrm{S}\to\mathrm{C}}(x) + \mathcal{K}_{\mathrm{C}\to\mathrm{E}+\mathrm{S}}(x) + \mathcal{K}_{\mathrm{C}\to\mathrm{E}+\mathrm{P}}(x)\\
[\dot{\mathrm{S}}] &= -\mathcal{K}_{\mathrm{E}+\mathrm{S}\to\mathrm{C}}(x) + \mathcal{K}_{\mathrm{C}\to\mathrm{E}+\mathrm{P}}(x)\\
[\dot{\mathrm{C}}] &= \mathcal{K}_{\mathrm{E}+\mathrm{S}\to\mathrm{C}}(x) - \mathcal{K}_{\mathrm{C}\to\mathrm{E}+\mathrm{S}}(x) - \mathcal{K}_{\mathrm{C}\to\mathrm{E}+\mathrm{P}}(x)\\
[\dot{\mathrm{P}}] &= \mathcal{K}_{\mathrm{C}\to\mathrm{E}+\mathrm{P}}(x).
\end{aligned}$$

△

Definition 1.4. *A chemical reaction network $(\mathcal{S}, \mathcal{C}, \mathcal{R})$ together with a kinetics $\mathcal{K}$ is called a* kinetic system *$(\mathcal{S}, \mathcal{C}, \mathcal{R}, \mathcal{K})$.*

In general, for a kinetic system given as $(\mathcal{S}, \mathcal{C}, \mathcal{R}, \mathcal{K})$, for each species $i \in \{1, \dots, n\}$ the following differential equation holds:

$$\dot{x}_i = \sum_{(c,c') \in \mathcal{R}} (c'_i - c_i)\, \mathcal{K}_{c \to c'}(x). \tag{1.7}$$

One of the most common kinetics is *mass action kinetics* [106]. It is based on the assumption that the reaction rate is proportional to the product of the molar concentrations of the reactants. The underlying idea is that each reactant has a certain probability to be at a certain point in space and that this probability is proportional to the molar concentration of the reactant, cf. Feinberg [30, Chapter 2.1.2].

This leads to the following definition, which also follows Feinberg [30, Definition 3.2.6].

Definition 1.5. *A kinetics $\mathcal{K}$ for a chemical reaction network $(\mathcal{S}, \mathcal{C}, \mathcal{R})$ is called* mass action *if for each $c \to c' \in \mathcal{R}$ there exists a $k_{c\to c'} \in (0, \infty)$ such that*

$$\mathcal{K}_{c\to c'}(x) = k_{c\to c'}\, x^c. \tag{1.8}$$

Such $k_{c\to c'}$ is called the rate constant *for reaction $c \to c'$.*

With that in mind, we write down the differential equations of a chemical reaction network with mass action kinetics. For each species $i \in \{1, \dots, n\}$ the molar concentration over time is described by

$$\dot{x}_i = \sum_{(c,c')\in\mathcal{R}} (c'_i - c_i)\, k_{c\to c'}\, x^c. \tag{1.9}$$

It is evident from the above equation (1.9) that the right hand side of the differential equation is a polynomial over $\mathbb{R}[x_1, \dots, x_n]$. Note that for the methods we are going to describe we only need polynomial right hand sides in our differential equations, not necessarily mass action kinetics. However, mass action is a prominent example of a kinetics that produces such equations.

Example 1.3. *We revisit Example 1.2, which is in fact the famous irreversible Michaelis–Menten reaction [70], a prototypical enzymatic reaction. This system is described by the chemical reactions*

$$\mathrm{E} + \mathrm{S} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \mathrm{C} \xrightarrow{k_2} \mathrm{E} + \mathrm{P}, \tag{1.10}$$

where E, S, C, *and* P *are the enzyme, substrate, enzyme-substrate complex, and product, respectively. We have introduced shorter names for the rate constants, namely $k_1 := k_{\mathrm{E}+\mathrm{S}\to\mathrm{C}}$, $k_{-1} := k_{\mathrm{C}\to\mathrm{E}+\mathrm{S}}$, and $k_2 := k_{\mathrm{C}\to\mathrm{E}+\mathrm{P}}$. By convention, the rate constants are written on top of or, for reverse reactions, below of the reaction arrows. Then the differential equations that are induced by this kinetic system are*

$$\begin{aligned}
[\dot{\mathrm{E}}] &= -k_1[\mathrm{E}][\mathrm{S}] + k_{-1}[\mathrm{C}] + k_2[\mathrm{C}] \\
[\dot{\mathrm{S}}] &= -k_1[\mathrm{E}][\mathrm{S}] + k_{-1}[\mathrm{C}] \\
[\dot{\mathrm{C}}] &= k_1[\mathrm{E}][\mathrm{S}] - k_{-1}[\mathrm{C}] - k_2[\mathrm{C}] \\
[\dot{\mathrm{P}}] &= k_2[\mathrm{C}].
\end{aligned} \tag{1.11}$$

△

1.4 Singular Perturbation Theory

Ordinary differential equations are a useful mathematical tool to describe the dynamics of kinetic systems. Recall that the reason for us to seek such a description is the desire to understand the evolution of such kinetic systems.

However, kinetic systems that model processes in nature often have many differential variables and different reactions may take place at different time scales, that is to say, at different speeds. This makes analysis and even simulation computationally expensive, if not infeasible. *Model reduction*, that is, the search for smaller models that under certain conditions behave approximately like the original model, is thus a worthwhile endeavor.

Given a system with different time scales and prescribed "slow" and "fast" variables, *singular perturbation theory* provides a method to determine asymptotically invariant manifolds and reduced systems on these manifolds.

In the following, we formalize the above for two time scales. This is due to Tikhonov [102] and Fenichel [31]. Example 1.4 was previously published in [51]. First, we recall the following definition.

Definition 1.6. *We write the* Jacobian matrix *in the following shorthand form, which is due to Euler. For*

$$f : \mathbb{R}^n \to \mathbb{R}^m, \quad (x_1, \dots, x_n) \mapsto f(x_1, \dots, x_n)$$

with $x = (x_1, \dots, x_n)$, $f = (f_1, \dots, f_m)$, and $s \in \{1, \dots, n\}$ we define

$$D_{x_1,\dots,x_s} f := \begin{pmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_s} \\ \multicolumn{3}{c}{\dotfill} \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_s} \end{pmatrix} \quad \text{and} \quad Df := \begin{pmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \multicolumn{3}{c}{\dotfill} \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{pmatrix}.$$

We start with a system with a small parameter ε for which intuitively $z_1 \in D \subseteq \mathbb{R}^{n_1}$ contains variables that change rapidly, whereas $z_2 \in G \subseteq \mathbb{R}^{n_2}$ contains variables that change more slowly in comparison, with $n_1 + n_2 = n$. Formally, this is

$$\begin{aligned} \dot{z}_1 &= f_1(z_1, z_2) + \varepsilon(\dots) \\ \dot{z}_2 &= \varepsilon f_2(z_1, z_2) + \varepsilon^2(\dots), \end{aligned} \tag{1.12}$$

where the dots represent sufficiently differentiable functions in z_1, z_2, and ε. We also perform a *rescaling* to "slow time" $\tau := \varepsilon t$, that is, substituting $t = \frac{\tau}{\varepsilon}$. To distinguish the two different time derivatives, we use Lagrange's prime notation when we mean a time derivative for τ, that is

$$x' := \frac{\mathrm{d}x}{\mathrm{d}\tau}. \tag{1.13}$$

After cancellation, this yields

$$\begin{aligned} \varepsilon z_1' &= f_1(z_1, z_2) + \varepsilon(\dots) \\ z_2' &= f_2(z_1, z_2) + \varepsilon(\dots). \end{aligned} \tag{1.14}$$

We call (1.12) the *fast system* and (1.14) the *slow system.*

Theorem 1.4 (Tikhonov–Fenichel)**.** *We assume that the following holds.*

(i) The so-called critical manifold $\widetilde{Z}$ *is not empty, that is*

$$\widetilde{Z} := \{\, (z_1, z_2) \in D \times G \mid f_1(z_1, z_2) = 0 \,\} \neq \varnothing.$$

(ii) There exists $\nu > 0$ such that for all $(z_1, z_2) \in \widetilde{Z}$, every eigenvalue of the partial derivative $D_{z_1} f_1(z_1, z_2)$ has a real part $\leq -\nu$.

Then there exists $T > 0$ and a neighborhood of $\widetilde{Z}$ in which all solutions of (1.12) *converge uniformly to solutions of*

$$\begin{aligned} 0 &= f_1(z_1, z_2) \\ z_2' &= f_2(z_1, z_2) \end{aligned} \tag{1.15}$$

on $[t_0, T]$, for arbitrary $t_0 > 0$.

Condition (ii) guarantees that locally the implicit equation $f_1(z_1, z_2) = 0$ can be resolved into the form $z_1 = h(z_2)$, for some function h, due to the implicit function theorem. Given such a resolution, we have the differential equation

$$z_2' = f_2(h(z_2), z_2). \tag{1.16}$$

Notice that it may not be possible to determine h explicitly. For a more detailed discussion, see Verhulst [104, Section 8.2].

Example 1.4. *We illustrate this process on the aforementioned Michaelis–Menten reaction, already described in Example 1.3:*

$$\mathrm{E} + \mathrm{S} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \mathrm{C} \xrightarrow{k_2} \mathrm{E} + \mathrm{P}.$$

We proceed similarly to Heineken, Tsuchiya, and Aris [39]. Observe that the mechanism has two conserved quantities $[\mathrm{E}] + [\mathrm{C}] =: \alpha$ and $[\mathrm{S}] + [\mathrm{C}] + [\mathrm{P}] =: \beta$, where α and β are constant functions representing the respective total concentrations. We choose the concentration units such that $\beta = 1$, and furthermore rename $\alpha =: \varepsilon$. We assume that ε is small. Thus, from a biological standpoint, the total concentration of enzyme is small. Next, we eliminate the variable $[\mathrm{E}] = \varepsilon - [\mathrm{C}]$ and notice that variable $[\mathrm{P}] = 1 - [\mathrm{S}] - [\mathrm{C}]$. Applying this to (1.11) *leads to the following reduced system of differential equations:*

$$\begin{aligned} [\dot{\mathrm{S}}] &= -\varepsilon k_1[\mathrm{S}] + (k_1[\mathrm{S}] + k_{-1})[\mathrm{C}] \\ [\dot{\mathrm{C}}] &= \varepsilon k_1[\mathrm{S}] - (k_1[\mathrm{S}] + k_{-1} + k_2)[\mathrm{C}]. \end{aligned}$$

The parameter ε represents the ratio of total concentrations α to β.

Next, we perform a rescaling and introduce new variables z_1 and z_2, and substitute $[\mathrm{C}] = \varepsilon z_1$ as well as $[\mathrm{S}] = z_2$. The biochemical rationale behind this rescaling is that one considers the case where there is only a small amount of C*, the enzyme-substrate complex. This yields the following system of differential equations:*

$$\begin{aligned} \dot{z}_1 &= k_1 z_2 - (k_1 z_2 + k_{-1} + k_2) z_1 \\ \dot{z}_2 &= \varepsilon(-k_1 z_2 + (k_1 z_2 + k_{-1}) z_1). \end{aligned} \tag{1.17}$$

This matches the set-up in (1.12) *with*

$$\begin{aligned} f_1(z_1, z_2) &= k_1 z_2 - (k_1 z_2 + k_{-1} + k_2) z_1, \\ f_2(z_1, z_2) &= -k_1 z_2 + (k_1 z_2 + k_{-1}) z_1, \end{aligned}$$

and no further terms. The next step is the rescaling to slow time by substituting $t = \frac{\tau}{\varepsilon}$*, which yields the analogue to* (1.14)*, namely*

$$\begin{aligned}\varepsilon z_1' &= k_1 z_2 - (k_1 z_2 + k_{-1} + k_2) z_1\\ z_2' &= (-k_1 z_2 + (k_1 z_2 + k_{-1}) z_1).\end{aligned}$$

This then leads for $\varepsilon \to 0$ *to the system*

$$\begin{aligned}0 &= k_1 z_2 - (k_1 z_2 + k_{-1} + k_2) z_1 && (1.18)\\ z_2' &= (-k_1 z_2 + (k_1 z_2 + k_{-1}) z_1), && (1.19)\end{aligned}$$

where (1.18) *is the quasi-steady-state condition. We substitute* (1.18) *into* (1.19) *to arrive at the following equation, which is the concretization of* (1.16) *from Theorem 1.4 above:*

$$z_2' = -\frac{k_1 k_2 z_2}{k_1 z_2 + k_{-1} + k_2}. \tag{1.20}$$

This is the same result as Briggs and Haldane found [15].

Now we prove that this reduction is correct by checking the conditions in Theorem 1.4.

(i) The critical manifold

$$\widetilde{Z} := \{\, (z_1, z_2) \mid k_1 z_2 - (k_1 z_2 + k_{-1} + k_2) z_1 = 0 \,\}$$

is not empty, since obviously $(0, 0) \in \widetilde{Z}$*.*

(ii) For all $(z_1, z_2) \in \widetilde{Z}$*, every eigenvalue of*

$$\begin{aligned}D_{z_1} f_1(z_1, z_2) &= \frac{\partial(k_1 z_2 - (k_1 z_2 + k_{-1} + k_2) z_2)}{\partial z_1}\\ &= -(k_1 z_2 + k_{-1} + k_2)\end{aligned}$$

has a real part $\leq -\nu$*, with* $\nu = k_{-1} + k_2$*, where* k_{-1} *and* k_2 *are both assumed positive.*

A back-transformation to time t *yields the reduced system*

$$\begin{aligned}0 &= k_1 z_2 - (k_1 z_2 + k_{-1} + k_2) z_1\\ \dot{z}_2 &= -\varepsilon \frac{k_1 k_2 z_2}{k_1 z_2 + k_{-1} + k_2}.\end{aligned} \tag{1.21}$$

△

In Section 2.1, we use a generalization of Theorem 1.4 by Cardin and Teixeira [18] that allows the partitioning into more than two time scales and we present algorithms for the reduction process.

1.5 A General Scaling Procedure

The results of this section were previously published in [51]. Our starting point is a parameter dependent system S of n polynomial ordinary differential equations

$$\dot{x}_k = \frac{\mathrm{d}x_k}{\mathrm{d}t} = \sum_J \gamma_{k,J}\, x^J, \quad 1 \le k \le n, \tag{1.22}$$

where the summation ranges over multi-indices $J = (j_1, \dots, j_n) \in \mathbb{N}^n$, $\gamma_{k,J} \in \mathbb{R}$, and only finitely many $\gamma_{k,J}$ are non-zero. Note that we use the convention that the natural numbers $\mathbb{N}$ include 0. We use the notation "$1 \le k \le n$" to express something for all n values of k at once.

Throughout this work, we require that positive x_k remain positive as time progresses. In other words, the positive first orthant $\mathcal{U} := (0, \infty)^n$ is positively invariant for system (1.22), which is the case for (1.22) if and only if $\gamma_{k,J} x^J \ge 0$ on all intersections of hyperplanes $\{\, (x_1, \dots, x_n) \in \mathbb{R}^n \mid x_k = 0 \,\}$ with $\overline{\mathcal{U}}$.

We fix some $\varepsilon_* \in (0, 1)$, which intuitively is thought to be "small", and we impose that

$$\gamma_{k,J} = {\varepsilon_*}^{c_{k,J}} \bar{\gamma}_{k,J}, \tag{1.23}$$

with rational numbers $c_{k,J}$. The tacit understanding is that only nonzero $\gamma_{k,J}$ are being considered. The intuitive idea, which will be made more precise in Section 2.2, is that the $\bar{\gamma}_{k,J}$ are close to one. Moreover, we introduce a positive parameter ε and consider the system

$$\dot{x}_k = \sum_J \varepsilon^{c_{k,J}}\, \bar{\gamma}_{k,J}\, x^J, \quad 1 \le k \le n, \tag{1.24}$$

with ε-dependent coefficients. Notice that (1.24) matches (1.22) at $\varepsilon = \varepsilon_*$. By renormalizing

$$x_k = \varepsilon^{d_k} y_k, \quad d_k \in \mathbb{Q}, \tag{1.25}$$

one obtains a system in scaled variables

$$\dot{y}_k = \sum_J \varepsilon^{c_{k,J} + \langle D, J \rangle - d_k}\, \bar{\gamma}_{k,J}\, y^J, \quad 1 \le k \le n, \tag{1.26}$$

with $D = (d_1, \dots, d_n)$ and the dot product in $\mathbb{R}^n$ denoted by $\langle \cdot, \cdot \rangle$. This transformation preserves the positive invariance of $\mathcal{U}$.[1] Continuing, we set

$$\nu_k = \min\{\, c_{k,J} + \langle D, J \rangle - d_k \mid \bar{\gamma}_{k,J} \ne 0 \,\}$$

to obtain

$$\dot{y}_k = \varepsilon^{\nu_k} \sum_J \varepsilon^{c_{k,J} + \langle D, J \rangle - d_k - \nu_k}\, \bar{\gamma}_{k,J}\, y^J, \quad 1 \le k \le n, \tag{1.27}$$

where now all exponents of ε inside the sums are nonnegative. Finally, one may perform a preliminary time scaling $\tau = \varepsilon^{\mu} t$, $\mu = \min\{\nu_1, \dots, \nu_n\}$ to arrive at

$$y_k' = \frac{\mathrm{d}y_k}{\mathrm{d}\tau} = \varepsilon^{\nu_k - \mu} \sum_J \varepsilon^{c_{k,J} + \langle D, J \rangle - d_k - \nu_k}\, \bar{\gamma}_{k,J}\, y^J, \quad 1 \le k \le n, \tag{1.28}$$

[1] Underlying the scaling is the implicit assumption that for i, $j \in \{1, \dots, n\}$, the relative order of x_i with respect to x_j is bounded by $x_i/x_j = \Theta(\varepsilon^{d_i - d_j})$ for $\varepsilon \to 0$, so that all y_k have the same order of magnitude.

with all exponents nonnegative. We are interested in system (1.28) for variable $\varepsilon > 0$, in the asymptotic limit $\varepsilon \to 0$.

We adopted a rather general scaling formalism that has been used recently in several publications [75, 76, 81, 82, 94] and is recurrent in the literature on singular perturbations, see for instance Nipp [74, Section 3].

So far, scaling has been a formal exercise. The question remains where the numbers $c_{k,J}$ in (1.23) and the vector $D = (d_1, \dots, d_n)$ in (1.26) may come from. That will be made precise in Section 2.2. We use tropical geometry for that purpose, which we introduce in the next section.

1.6 Tropical Equilibrations

Tropical geometry [16, 62], a relatively new subfield of mathematics, offers new possibilities for the analysis of parameter-dependent polynomial or rational ordinary differential equations. It has also applications in economics and optimization like network flows and scheduling. *Tropical equilibration methods* for the analysis and reduction of chemical reaction networks were introduced and developed in a series of papers by Noel et al. [76], Radulescu et al. [82], Samal et al. [94, 93], and others. These methods allow to determine scalings of polynomial or rational vector fields that model the kinetics of biochemical networks and open a feasible path to reduction of networks of high dimension. For a given system, they provide a list of possible slow-fast systems, which may or may not yield invariant manifolds and reduced systems.

We present a very short introduction to tropical geometry as far as it concerns us for the matter at hand and how to compute the tropical equilibration of a set of polynomials. Parts of this section were previously published in [51]. As for the name, Katz [47] instructs us that

> "tropical geometry is named in honor of Brazilian computer scientist Imre Simon. This naming is complicated by the fact that he lived in Sao Paolo and commuted across the Tropic of Capricorn. Whether or not his work is tropical depends on whether or not he preferred to do his research at home or in the office."

Definition 1.7. *The tropical semi-field* $(\mathbb{T}, \oplus, \otimes)$ *is defined as* $\mathbb{T} := \mathbb{R} \cup \{\infty\}$*, with addition defined as minimization and multiplication as classical addition:*

$$x \oplus y := \min\{x, y\} \quad \textit{and} \quad x \otimes y := x + y.$$

The neutral element of multiplication is 0, and the neutral element of addition is ∞. Notice that there is no general notion of subtraction: we cannot call any number "13 minus 4", since there is no x such that $4 \oplus x = 13$. That is what makes $\mathbb{T}$ a *semi*-field. Notice furthermore that addition is an *idempotent operation*, that is, $x \oplus x = x$. Thus, *tropical polynomials* are piecewise linear concave curves with integer coefficients.

Example 1.5. *The following is an example of a tropical polynomial and how it translates into classical algebra:*

$$(x_1 \otimes x_1 \otimes x_1) \oplus (x_1 \otimes x_1 \otimes x_2) \oplus x_3 = \min\{3x_1, 2x_1 + x_2, x_3\}. \qquad \triangle$$

From a practical standpoint, *tropicalization* of a classical polynomial is the following process: map all variables to new names, say, from x to $\bar{x}$. Then, replace "$x + y$" by "$\min(\bar{x}, \bar{y})$", "$x \cdot y$" by "$\bar{x} + \bar{y}$", and "x^c" by "$c \cdot \bar{x}$". Coefficients "c" are replaced by "$\log_{\varepsilon_*}(|c|)$" with $\varepsilon_* \in (0,1)$ [62]. More formally, this leads to the following

Definition 1.8. *Let $g : \mathbb{R}^n \to \mathbb{R}$, multi-indices $J = (j_1, \ldots, j_n) \in \mathbb{N}^n$, $\gamma_J \in (0, \infty)$, $\varepsilon_* \in (0,1)$. Then, the classical polynomial*

$$g(x) = \sum_J \gamma_J x^J \tag{1.29}$$

induces the tropical polynomial

$$\operatorname{trop}(g)(\bar{x}) = \min_J \{\log_{\varepsilon_*}(|\gamma_J|) + \langle \bar{x}, J \rangle\}. \tag{1.30}$$

In Section 1.5 we left unresolved the question of how to choose $c_{k,J}$ and D. In view of singular perturbation theory reductions, it is sensible to choose the scaling in such a way that cancellation of lowest order monomials is possible. This will be discussed in more detail in Section 2.2.2.

Consider a classical polynomial like the right hand side of (1.22) and partition its monomials by their sign into two sets, P and N. Tropicalize the monomials in both sets and call the resulting sets $\bar{P}$ and $\bar{N}$. Call the tropical monomials in those sets $\bar{P}_i$ and $\bar{N}_j$, respectively. Then, the *tropical equilibration* is the set of points where the minimum is attained at least twice, that is, at least once on each side:

$$\min_i \{\bar{P}_i\} = \min_j \{\bar{N}_j\}. \tag{1.31}$$

The resulting inequalities and equalities can be interpreted geometrically as a polyhedron in $\mathbb{R}^n$ with codimension of at least one, since the polyhedron contains by definition at least one equation.

The *tropical equilibration of a set of polynomials* is the set of points where the tropical equilibrations of all tropical polynomials coincide. Geometrically, it is the intersection of the polyhedra from (1.31) for different tropical polynomials. This intersection has the form of a union of polyhedra. We choose values for $D = (d_1, \ldots, d_n)$ in (1.26) from one of those polyhedra.

Example 1.6. *As an example, we consider `BIOMD0000000716`, which is related to the transmission dynamics of subtype H5N6 of the influenza A virus in the Philippines in August 2017 [56]. The model specifies four species: Susceptible birds (`S_b`), Infected birds (`I_b`), Susceptible humans (`S_h`), and Infected humans (`I_a`), the molar concentrations of which over time we map to differential variables y_1, ..., y_4, respectively. The input system is given by*

$$\begin{aligned}
\dot{x}_1 &= -\tfrac{9137}{2635182} x_1 x_2 - \tfrac{1}{730} x_1 + \tfrac{412}{73} \\
\dot{x}_2 &= \tfrac{9137}{2635182} x_1 x_2 - \tfrac{4652377}{961841430} x_2 \\
\dot{x}_3 &= -\tfrac{1}{6159375000} x_2 x_3 - \tfrac{1}{25258} x_3 + \tfrac{40758549}{3650000} \\
\dot{x}_4 &= \tfrac{1}{6159375000} x_2 x_3 - \tfrac{112500173}{2841525000000} x_4.
\end{aligned} \tag{1.32}$$

The numbers are the parameter values as specified in the biomodel.

Now we set the left hand sides to zero and partition the monomials by sign. This leads to the following system:

$$\begin{aligned}\frac{9137}{2635182}x_1x_2+\frac{1}{730}x_1&=\frac{412}{73}\\ \frac{4652377}{961841430}x_2&=\frac{9137}{2635182}x_1x_2\\ \frac{1}{6159375000}x_2x_3+\frac{1}{25258}x_3&=\frac{40758549}{3650000}\\ \frac{112500173}{2841525000000}x_4&=\frac{1}{6159375000}x_2x_3.\end{aligned} \tag{1.33}$$

Next, we tropicalize each side of each equation, setting $\varepsilon_*=\frac{1}{5}$. *For practical reasons, which will be made precise later, we round the result of the logarithm in* (1.30) *to integer values. This yields*

$$\begin{aligned}\min\{4+\bar{x}_1+\bar{x}_2,4+\bar{x}_1\}&=-1\\ 3+\bar{x}_2&=4+\bar{x}_1+\bar{x}_2\\ \min\{14+\bar{x}_2+\bar{x}_3,6+\bar{x}_3\}&=-1\\ 6+\bar{x}_4&=14+\bar{x}_2+\bar{x}_3.\end{aligned} \tag{1.34}$$

Further simplification leads to

$$\begin{aligned}\bar{x}_1&=-1\\ \bar{x}_2&=-4\\ \bar{x}_3&=-7\\ \bar{x}_4&=-3\end{aligned} \tag{1.35}$$

and hence we use $D=(-1,-4,-7,-3)$ *in* (1.26).

Following (1.25), *this leads to the following relabeling:* $x_1=\varepsilon^{-1}y_1$, $x_2=\varepsilon^{-4}y_2$, $x_3=\varepsilon^{-7}y_3$, *and* $x_4=\varepsilon^{-3}y_4$. *The corresponding version of* (1.26) *is*

$$\begin{aligned}\dot{y}_1&=-\frac{5710625}{2635182}y_1y_2-\varepsilon^4\frac{125}{146}y_1+\frac{412}{365}\\ \dot{y}_2&=\varepsilon^3\frac{5710625}{2635182}y_1y_2-\varepsilon^3\frac{116309425}{192368286}y_2\\ \dot{y}_3&=-\varepsilon^{10}\frac{15625}{15768}y_2y_3-\varepsilon^6\frac{15625}{25258}y_3+\varepsilon^6\frac{40758549}{18250000}\\ \dot{y}_4&=\varepsilon^6\frac{15625}{15768}y_2y_3-\varepsilon^6\frac{112500173}{181857600}y_4.\end{aligned} \tag{1.36}$$

△

Example 1.7. *As another example, recall the Michaelis–Menten system from* (1.11)*:*

$$\begin{aligned}\dot{[\mathrm{E}]}&=-k_1[\mathrm{E}][\mathrm{S}]+k_{-1}[\mathrm{C}]+k_2[\mathrm{C}]\\ \dot{[\mathrm{S}]}&=-k_1[\mathrm{E}][\mathrm{S}]+k_{-1}[\mathrm{C}]\\ \dot{[\mathrm{C}]}&=k_1[\mathrm{E}][\mathrm{S}]-k_{-1}[\mathrm{C}]-k_2[\mathrm{C}]\\ \dot{[\mathrm{P}]}&=k_2[\mathrm{C}].\end{aligned}$$

First, we replace the species C, S, *and* E *by variables* x_1, x_2, *and* x_3. *We may disregard the fourth equation. Notice that the order of variables has been changed, so* x_1 *and* x_2

map to the same species as y_1 *and* y_2*, respectively, did in Example 1.4. This yields*

$$\begin{aligned} \dot{x}_1 &= k_1x_2x_3 - k_{-1}x_1 - k_2x_1 \\ \dot{x}_2 &= -k_1x_2x_3 + k_{-1}x_1 \\ \dot{x}_3 &= -k_1x_2x_3 + k_{-1}x_1 + k_2x_1. \end{aligned} \tag{1.37}$$

Since we have no values for the parameters k_1*,* k_{-1}*, and* k_2*, we assume they are of order* 1*, more exactly: they are* $\in [\sqrt{\varepsilon_*}, \sqrt{1/\varepsilon_*}]$*. Next, we set the left hand side to zero and partition the system by sign of the monomials, which yields*

$$\begin{aligned} k_{-1}x_1 + k_2x_1 &= k_1x_2x_3 \\ k_1x_2x_3 &= k_{-1}x_1 \\ k_1x_2x_3 &= k_{-1}x_1 + k_2x_1. \end{aligned} \tag{1.38}$$

Since we assumed the parameters are of order 1*, their logarithm is* 0*. Hence, after tropicalization this is the result:*

$$\begin{aligned} \min\{\bar{x}_1, \bar{x}_1\} &= \bar{x}_2 + \bar{x}_3 \\ \bar{x}_2 + \bar{x}_3 &= \bar{x}_1 \\ \bar{x}_2 + \bar{x}_3 &= \min\{\bar{x}_1, \bar{x}_1\}, \end{aligned} \tag{1.39}$$

which is in all three cases the equation

$$\bar{x}_2 + \bar{x}_3 = \bar{x}_1. \tag{1.40}$$

From this we can choose $D = (1, 0, 1)$*. This can be plugged into* (1.25) *to lead to* $x_1 = \varepsilon^1 y_1$*,* $x_2 = \varepsilon^0 y_2$*,* $x_3 = \varepsilon^1 y_3$*. Notice that this is the same scaling that we used in* (1.17)*, in our example of Tikhonov's theorem. Since we assume the parameters* k_1*,* k_{-1}*, and* k_2 *to be near* 1*, the* $c_{k,J}$ *are* 0*. This then yields the following specialized version of* (1.26) *for our system:*

$$\begin{aligned} \dot{y}_1 &= \bar{k}_1y_2y_3 - \bar{k}_{-1}y_1 - \bar{k}_2y_1 \\ \dot{y}_2 &= -\varepsilon\bar{k}_1y_2y_3 + \varepsilon\bar{k}_{-1}y_1 \\ \dot{y}_3 &= -\bar{k}_1y_2y_3 + \bar{k}_{-1}y_1 + \bar{k}_2y_1. \end{aligned} \tag{1.41}$$

△

Example 1.8. *If we consider system* (1.37) *from the last example, yet use* $k_1 = k_{-1} = 1$*,* $k_2 = \frac{1}{5}$ *as parameter values, this leads to the following system of differential equations:*

$$\begin{aligned} \dot{x}_1 &= x_2x_3 - x_1 - \tfrac{1}{5}x_1 \\ \dot{x}_2 &= -x_2x_3 + x_1 \\ \dot{x}_3 &= -x_2x_3 + x_1 + \tfrac{1}{5}x_1. \end{aligned} \tag{1.42}$$

Again, we set the left hand side to zero and partition by sign of the monomials:

$$\begin{aligned} x_1 + \tfrac{1}{5}x_1 &= x_2x_3 \\ x_2x_3 &= x_1 \\ x_2x_3 &= x_1 + \tfrac{1}{5}x_1. \end{aligned} \tag{1.43}$$

Now tropicalize:

$$\begin{aligned} \min\{\bar{x}_1, 1+\bar{x}_1\} &= \bar{x}_2 + \bar{x}_3 \\ \bar{x}_2 + \bar{x}_3 &= \bar{x}_1 \\ \bar{x}_2 + \bar{x}_3 &= \min\{\bar{x}_1, 1+\bar{x}_1\}, \end{aligned} \tag{1.44}$$

which again leads to the equation

$$\bar{x}_2 + \bar{x}_3 = \bar{x}_1. \tag{1.45}$$

△

Research suggests a correspondence between the polyhedra of a chemical reaction network's tropical equilibration and so-called *metastable states* of a chemical reaction network. Those metastable states are where much of the observable chemical reactions happen, cf. Samal et al. [95]:

> For large networks with ordinary differential equations dynamics and multiple timescales it is reasonable to consider the following property: a typical trajectory consists in a succession of qualitatively different slow segments separated by faster transitions. The slow segments, generally called metastable states or regimes, can be of several types such as attractive invariant manifolds [...] The notion of metastability generalizes the notion of attractor. Like in the case of attractors, distant parts of the system can have coordinated activity for metastability. The dynamical states of large networks can be represented as points in a high dimensional space, called phase space. In this representation each coordinate represents the concentration of a molecular species. Coordinated activity means that many of the species concentrations are correlated, which can be geometrically represented by the proximity to a lower dimension hypersurface in the phase space. A system remains in the proximity of an attractor after entering its basin of attraction, but can leave a metastable regime after a relatively long time (much longer than the time needed for transitions between two different regimes).

In Chapter 3, we will concern ourselves with the efficient computation of the tropical equilibration and a novel algorithm to compute it is presented.

Chapter 2

Scaling and Reduction by Singular Perturbation Methods

ODE systems are well suited to model species concentrations of biochemical reaction networks. However, high-dimensional systems quickly become inconveniently hard to discuss. Partitioning systems into multiple time scales may allow their reduction into smaller systems and corresponding nested invariant manifolds. This can greatly facilitate their discussion or lead to new information about the original system.

The realization of the reduction is divided into two parts. First, we need to identify different time scales. One way to achieve this has been obtained by Noel et al. [75, 76] and Samal et al. [94, 93] using methods from tropical geometry, which we also apply. Having identified such time scales, the second step is to compute the reductions themselves. To this end, singular perturbation methods and especially the theory by Tikhonov–Fenichel [102, 31] are well known. Unfortunately, Tikhonov–Fenichel is limited to two time scales. Taking advantage of a recent extension to multiple time scales by Cardin and Teixeira [18], we specify this for our reaction network ODE systems.

This approach is made explicit through a collection of algorithms that precisely describe each part of the reduction procedure. Parts of this chapter were previously published in [51].

2.1 Singular Perturbations for Multiscale Systems

Building on Noel et al. [75, 76], Samal et al. were among the first to analyze multi-scale ODE systems based on tropical geometry [94, 93]. Yet they did not provide reduced systems, nor did they check for hyperbolic attractivity. While they succeeded in obtaining scalings via tropical geometry, they did not approach the problem of a systematic reduction and only discussed ad-hoc examples.

There we proceed due to the new developments in a recent paper by Cardin and Teixeira [18] which generalize Fenichel's theory to provide a solid foundation to obtain more than one nontrivial invariant manifold. This allows, in particular, the reduction of multi-scale ODE systems such as system (2.16). Technically, the approach considers a multi-parameter system using time scale factors ε_1, $\varepsilon_1\varepsilon_2$, … instead of increasing powers of one single ε.

In Section 1.4, we already discussed the partitioning of ODE systems into two time scales and the theorem by Tikhonov and Fenichel. In the current chapter, we introduce the theory for $m > 2$ time scales. To make matters transparent, yet to keep the exposition clear, we first consider a system with three time scales in the next section and scale this up to arbitrary m time scales in the section after that.

2.1.1 The Case With Three Time Scales

This section generalizes Kruff and Walcher [52, Section 2] and was previously published in [51, Section 3.1]. We start with a system of ordinary differential equations in differential variables z_1, z_2, z_3, and derivation by τ. In analogy to system (1.14), z_1, z_2, and z_3 are vectors of variables.

Let $z_1 \in \mathbb{R}^{n_1}$, $z_2 \in \mathbb{R}^{n_2}$, and $z_3 \in \mathbb{R}^{n_3}$; n_1, n_2, $n_3 \in \mathbb{N} \setminus \{0\}$, $n_1 + n_2 + n_3 = n$; $f_1 : \mathbb{R}^{n+2} \to \mathbb{R}$, $f_2 : \mathbb{R}^{n+2} \to \mathbb{R}$, $f_3 : \mathbb{R}^{n+2} \to \mathbb{R}$, $\varepsilon_1, \varepsilon_2 \in [0, 1)$. Furthermore, let $z = (z_1, z_2, z_3)$, $f = (f_1, f_2, f_3)$, and $\bar{\varepsilon} = (\varepsilon_1, \varepsilon_2)$. Let $U \subseteq \mathbb{R}^n$ be open and non-empty and f smooth on an open neighborhood of $U \times [0, \vartheta_1) \times [0, \vartheta_2)$ with $\vartheta_1 > 0$, $\vartheta_2 > 0$. We consider the following system of differential equations:

$$\begin{aligned} z_1' &= f_1(z, \bar{\varepsilon}) \\ z_2' &= \varepsilon_1 f_2(z, \bar{\varepsilon}) \\ z_3' &= \varepsilon_1 \varepsilon_2 f_3(z, \bar{\varepsilon}). \end{aligned} \tag{2.1}$$

We define the sets

$$\begin{aligned} \mathcal{M}_1 &:= \{\, z \in U \mid f_1(z, 0) = 0 \,\} \\ \mathcal{M}_2 &:= \{\, z \in U \mid f_1(z, 0) = f_2(z, 0) = 0 \,\} \end{aligned} \tag{2.2}$$

and assume that they are non-empty. Notice that $U \supseteq \mathcal{M}_1 \supseteq \mathcal{M}_2$. Furthermore, we let

$$\mathcal{M}_2^{\varepsilon_2} := \{\, z \in U \mid f_1(z, 0, \varepsilon_2) = 0 \,\}. \tag{2.3}$$

Cardin and Teixeira's theory [18] requires a series of hyperbolicity conditions, whereas we use the stronger notion of hyperbolic attractivity, since in our applications we focus on *attracting* invariant manifolds.

Definition 2.1. *We call $\mathcal{M}_1$* hyperbolically attractive *on U if for all $z \in \mathcal{M}_1$ all eigenvalues of the Jacobian $D_{z_1} f_1(z, 0)$ have negative real parts.*

Furthermore, we call $\mathcal{M}_2$ hyperbolically attractive *on $\mathcal{M}_1$ if the following condition holds. Recall that the implicit function theorem yields a unique local resolution of the equation $f_1(z, \bar{\varepsilon}) = 0$ in the form $z_1 = h(z_2, z_3, \bar{\varepsilon})$. With this, we define*

$$\begin{aligned} f_2^*(z_2, z_3, \bar{\varepsilon}) &:= f_2(z_1, z_2, z_3, \bar{\varepsilon}) \\ &= f_2(h(z_2, z_3, \bar{\varepsilon}), z_2, z_3, \bar{\varepsilon}). \end{aligned}$$

The condition is now that for all $z \in \mathcal{M}_2$ all eigenvalues of $D_{z_2} f_2^*(z_2, z_3, 0)$ have negative real parts.

The following proposition is similar to Kruff–Walcher [52, Proposition 1]. A proof can be found ibid.

Proposition 2.1. *$\mathcal{M}_2$ is hyperbolically attractive on $\mathcal{M}_1$ if $\mathcal{M}_1$ is hyperbolically attractive on U and either of the following equivalent conditions is met:*

(i) For all $z \in \mathcal{M}_2$, all eigenvalues of

$$W(z) := D_{z_2} f_2(z,0) - D_{z_1} f_2(z,0)\, D_{z_1} f_1(z,0)^{-1}\, D_{z_2} f_1(z,0)$$

have negative real parts.

(ii) For all $z \in \mathcal{M}_2$ and all sufficiently small $\varrho > 0$, all eigenvalues of

$$J(z,\varrho) := \begin{pmatrix} D_{z_1} f_1(z,0) & D_{z_2} f_1(z,0) \\ \varrho D_{z_1} f_2(z,0) & \varrho D_{z_2} f_2(z,0) \end{pmatrix}$$

have negative real parts.

We now turn to reduction. Firstly, by introducing $\tau_1 := \tau$ and letting $\varepsilon_1 \to 0$ in (2.1), we obtain the *boundary layer system*:

$$\begin{aligned} \frac{\mathrm{d}z_1}{\mathrm{d}\tau_1} &= f_1(z,0) \\ \frac{\mathrm{d}z_2}{\mathrm{d}\tau_1} &= 0 \\ \frac{\mathrm{d}z_3}{\mathrm{d}\tau_1} &= 0. \end{aligned} \tag{2.4}$$

Next, we introduce $\tau_2 := \varepsilon_1 \tau$. By substituting $\tau = \frac{\tau_2}{\varepsilon_1}$ in (2.1), canceling and letting $\varepsilon_1 \to 0$, we arrive at the *auxiliary system* on $\mathcal{M}_2^{\varepsilon_2}$:

$$\begin{aligned} 0 &= \quad f_1(z,0,\varepsilon_2) \\ \frac{\mathrm{d}z_2}{\mathrm{d}\tau_2} &= \quad f_2(z,0,\varepsilon_2) \\ \frac{\mathrm{d}z_3}{\mathrm{d}\tau_2} &= \varepsilon_2 f_3(z,0,\varepsilon_2), \end{aligned} \tag{2.5}$$

and by letting $\varepsilon_2 \to 0$ in (2.5), we arrive at the *intermediate reduced system* on $\mathcal{M}_1$:

$$\begin{aligned} 0 &= f_1(z,0) \\ \frac{\mathrm{d}z_2}{\mathrm{d}\tau_2} &= f_2(z,0) \\ \frac{\mathrm{d}z_3}{\mathrm{d}\tau_2} &= 0. \end{aligned} \tag{2.6}$$

Lastly, we introduce $\tau_3 := \varepsilon_1 \varepsilon_2 \tau$. Again, by substituting $\tau = \frac{\tau_3}{\varepsilon_1 \varepsilon_2}$ in (2.1), canceling and then letting ε_1, $\varepsilon_2 \to 0$, we arrive at the *completely reduced system* on $\mathcal{M}_2$:

$$\begin{aligned} 0 &= f_1(z,0) \\ 0 &= f_2(z,0) \\ \frac{\mathrm{d}z_3}{\mathrm{d}\tau_3} &= f_3(z,0). \end{aligned} \tag{2.7}$$

If the sets $\mathcal{M}_1$ and $\mathcal{M}_2$ for system (2.1) are hyperbolically attractive as per Definition 2.1, then the system can be partitioned into three time scales and the three reduced systems (2.4), (2.6), and (2.7), where the last two systems each describe an invariant manifold.

For the following theorem see Kruff and Walcher [52, Theorem 1]. It ensures that the reduced systems are approximate solutions to the original system.

Theorem 2.1. *Let system* (2.1) *be given and* $\mathcal{M}_2$ *be hyperbolically attractive on* $\mathcal{M}_1$*. Then the following holds.*

1. *Let* $\mathcal{N} \subseteq \mathcal{M}_2$ *be a compact submanifold. Then for all sufficiently small* $\bar{\varepsilon}$*, system* (2.1) *admits an invariant manifold* $\mathcal{N}_2^{\bar{\varepsilon}}$ *that is* $(\varepsilon_1 + \varepsilon_2)$*-close to* $\mathcal{N}$ *with respect to the Hausdorff distance. Given time scale* τ_2*, solutions of* (2.1) *on* $\mathcal{N}_2^{\bar{\varepsilon}}$ *converge to solutions of* (2.7) *on* $\mathcal{N}$*. More precisely, there exists* $T > 0$ *such that the convergence is uniform on any closed subinterval of* $(0, T)$ *as* $\varepsilon \to 0$*.*

2. *Let* ε_2 *be sufficiently small and* $\mathcal{N}_1 \subseteq \mathcal{M}_2^{\varepsilon_2}$ *be a compact submanifold. Then for all sufficiently small* ε_1*, system* (2.1) *admits an invariant manifold* $\mathcal{N}_1^{\bar{\varepsilon}}$ *that is* $(\varepsilon_1 + \varepsilon_2)$*-close to* $\mathcal{N}_1$ *with respect to the Hausdorff distance. Given time scale* τ_1*, solutions of* (2.1) *on* $\mathcal{N}_1^{\bar{\varepsilon}}$ *converge to solutions of* (2.6) *on* $\mathcal{N}_1$*. As above, there exists* $T > 0$ *such that the convergence is uniform on any closed subinterval of* $(0, T)$ *as* $\varepsilon \to 0$*.*

2.1.2 The General Case

In the last section, we described the special case for $m = 3$ time scales. Building on this example, in this section we expound upon the general case with m equations. We start with a system of m ordinary differential equations in differential variables $z_1, \ldots, z_m$ and derivation by τ.

Let $m \in \mathbb{N} \setminus \{0\}$. For $1 \le k \le m$, let $z_k \in \mathbb{R}^{n_k}$, $n_k \in \mathbb{N} \setminus \{0\}$, $f_k : \mathbb{R}^{n+m-1} \to \mathbb{R}$; $\sum_k n_k = n$. Furthermore, let $z = (z_1, \ldots, z_m)$, $f = (f_1, \ldots, f_m)$, $\varepsilon_1, \ldots, \varepsilon_{m-1} \in [0, 1)$, and $\bar{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_{m-1})$. Let $U \subseteq \mathbb{R}^n$ be open and non-empty and f smooth on an open neighborhood of $U \times [0, \vartheta_1) \times \ldots \times [0, \vartheta_{m-1})$ with $\vartheta_1 > 0, \ldots, \vartheta_{m-1} > 0$. We consider the following system of differential equations:

$$
\begin{aligned}
z_1' &= f_1(z, \bar{\varepsilon}) \\
z_2' &= \varepsilon_1 f_2(z, \bar{\varepsilon}) \\
&\vdots \\
z_m' &= \varepsilon_1 \cdots \varepsilon_{m-1} f_m(z, \bar{\varepsilon}).
\end{aligned}
\tag{2.8}
$$

We define the sets

$$
\begin{aligned}
\mathcal{M}_0 &:= U \\
\mathcal{M}_1 &:= \{\, z \in U \mid f_1(z, 0) = 0 \,\} \\
&\vdots \\
\mathcal{M}_{m-1} &:= \{\, z \in U \mid f_1(z, 0) = \ldots = f_{m-1}(z, 0) = 0 \,\}
\end{aligned}
\tag{2.9}
$$

and assume that all of them are non-empty. Notice that $\mathcal{M}_0 \supseteq \mathcal{M}_1 \supseteq \cdots \supseteq \mathcal{M}_{m-1}$.

For $1 \leq k \leq m$, we define time scales $\tau_k := \varepsilon_1 \cdots \varepsilon_{k-1}\tau$. As in [18, p. 1432], we also introduce the sets

$$\mathcal{M}_k^{(\varepsilon_k,\dots,\varepsilon_{m-1})} := \{\, z \in U \mid f_k(z, 0, \dots, 0, \varepsilon_k, \dots, \varepsilon_{m-1}) = 0 \,\}, \quad 1 \leq k \leq m \tag{2.10}$$

and auxiliary systems on $\mathcal{M}_k^{(\varepsilon_k,\dots,\varepsilon_{m-1})}$ defined as

$$\left.\begin{aligned} 0 &= f_1(z, \bar{\varepsilon}) \\ &\vdots \\ 0 &= f_{k-1}(z, \bar{\varepsilon}) \\ \frac{\mathrm{d}z_k}{\mathrm{d}\tau_k} &= f_k(z, \bar{\varepsilon}) \\ \frac{\mathrm{d}z_{k+1}}{\mathrm{d}\tau_k} &= \varepsilon_k \dots \varepsilon_{m-1} f_{k+1}(z, \bar{\varepsilon}) \\ &\vdots \\ \frac{\mathrm{d}z_m}{\mathrm{d}\tau_k} &= f_m(z, \bar{\varepsilon}) \end{aligned}\right\} \quad 1 \leq k \leq m. \tag{2.11}$$

These auxiliary systems are important for the proof of Theorem 2.2 below.

We now bring forth the notion of hyperbolic attractivity for the general case.

Definition 2.2. *We call $\mathcal{M}_1$* hyperbolically attractive *on $\mathcal{M}_0$ if for all $z \in \mathcal{M}_1$ all eigenvalues of the Jacobian $D_{z_1} f_1(z, 0)$ have negative real parts. Moreover, we define $f_1^*(z, 0) := f_1(z, 0)$.*

For $k \in \{2, \dots, m-1\}$, we call $\mathcal{M}_k$ hyperbolically attractive *on $\mathcal{M}_{k-1}$ if the following condition holds. The unique local resolution of the equation $f_{k-1}(z, \bar{\varepsilon}) = 0$ is $(z_1, \dots, z_{k-1}) = h_{k-1}(z_k, \dots, z_m, \bar{\varepsilon})$. With this, we define*

$$\begin{aligned} f_k^*(z_k, \dots, z_m, \bar{\varepsilon}) &:= f_k(z_1, \dots, z_m, \bar{\varepsilon}) \\ &= f_k(h_{k-1}(z_k, \dots, z_m, \bar{\varepsilon}), z_k, \dots, z_m, \bar{\varepsilon}). \end{aligned}$$

The condition is now that for all $z \in \mathcal{M}_k$ all eigenvalues of $D_{z_k} f_k^*(z_k, \dots, z_m, 0)$ have negative real parts. In the following we write $f_k^*(z, \bar{\varepsilon})$, by which we mean $f_k^*(z_k, \dots, z_m, \bar{\varepsilon})$ for the respective k.

Furthermore, for $k \in \{1, \dots, m\}$ define $F_k(z, \bar{\varepsilon}) := (f_1(z, \bar{\varepsilon}), \dots, f_k(z, \bar{\varepsilon}))$ and $Z_k := (z_1, \dots, z_k)$.

In the case of hyperbolic attractivity of $\mathcal{M}_k$ on $\mathcal{M}_{k-1}$ we write $\mathcal{M}_{k-1} \triangleright \mathcal{M}_k$, otherwise $\mathcal{M}_{k-1} \not\triangleright \mathcal{M}_k$. If we find for some $\ell \in \{1, \dots, m\}$ that $\mathcal{M}_0 \triangleright \mathcal{M}_1$, $\mathcal{M}_1 \triangleright \mathcal{M}_2$, $\dots$, $\mathcal{M}_{\ell-1} \triangleright \mathcal{M}_\ell$, then we simply write $\mathcal{M}_0 \triangleright \cdots \triangleright \mathcal{M}_\ell$, and call this a hyperbolically attractive ℓ-chain. *Such a chain is called* maximal *if either $\ell = m$ or $\mathcal{M}_\ell \not\triangleright \mathcal{M}_{\ell+1}$.*

The following proposition is the analog to Proposition 2.1 and was previously published in [51, Proposition 4 & Lemma 3]. Proofs can be found ibid.

Proposition 2.2. *Define $A_1 := D_{z_1} f_1(z, 0)$, and for $k \in \{2, \dots, m-1\}$*

$$\begin{pmatrix} A_{k-1} & B_k \\ C_k & V_k \end{pmatrix} := \begin{pmatrix} D_{Z_{k-1}} F_{k-1}(z, 0) & D_{z_k} F_{k-1}(z, 0) \\ D_{Z_{k-1}} f_k^*(z, 0) & D_{z_k} f_k^*(z, 0) \end{pmatrix}.$$

Note that $\begin{pmatrix} A_{k-1} & B_k \\ C_k & V_k \end{pmatrix} = A_k$. *Moreover, for* $k \in \{1, \dots, m\}$ *define*

$$\begin{aligned} J_k(z, \varrho_1, \dots, \varrho_{k-1}) &:= \begin{pmatrix} 1 & & \\ & \ddots & \\ & & \varrho_1 \cdots \varrho_{k-1} \end{pmatrix} \cdot D_{Z_k} F_k(z, 0) \\ &= \begin{pmatrix} D_{z_1} f_1^*(z,0) & \dots & D_{z_k} f_1^*(z,0) \\ \varrho_1 D_{z_1} f_2^*(z,0) & \dots & \varrho_1 D_{z_k} f_2^*(z,0) \\ \dots & \dots & \dots \\ \varrho_1 \cdots \varrho_{k-1} D_{z_1} f_k^*(z,0) & \dots & \varrho_1 \cdots \varrho_{k-1} D_{z_k} f_k^*(z,0) \end{pmatrix}. \end{aligned}$$

Then, for $k \in \{2, \dots, m-1\}$, $\mathcal{M}_k$ *is hyperbolically attractive on* $\mathcal{M}_{k-1}$ *if* $\mathcal{M}_{k-1}$ *is hyperbolically attractive on* $\mathcal{M}_{k-2}$ *and either of the following equivalent conditions is met.*

(i) For all $z \in \mathcal{M}_k$, *all eigenvalues of* $W_k(z)$ *with*

$$W_k := V_k - C_k A_k^{-1} B_k$$

have negative real parts.

(ii) For all $z \in \mathcal{M}_k$ *and all sufficiently small* $\varrho_1 > 0$, …, $\varrho_{k-1} > 0$, *all eigenvalues of* $J(z, \varrho_1, \dots, \varrho_{k-1})$ *have negative real parts.*

W_k is also known as the *Schur complement* [97, 37].

We now turn to reductions of system (2.8). By introducing τ_1 and letting ε_1, …, $\varepsilon_{m-1} \to 0$ in (2.8), we obtain the *boundary layer system*:

$$\begin{aligned} \frac{\mathrm{d}z_1}{\mathrm{d}\tau_1} &= f_1(z,0) \\ \frac{\mathrm{d}z_2}{\mathrm{d}\tau_1} &= 0 \\ &\vdots \\ \frac{\mathrm{d}z_m}{\mathrm{d}\tau_1} &= 0. \end{aligned} \tag{2.12}$$

For $k \in \{2, \dots, m\}$, we introduce τ_k by substituting $\tau = \frac{\tau_k}{\varepsilon_1 \cdots \varepsilon_{k-1}}$ in (2.8), canceling and letting ε_1, …, $\varepsilon_{m-1} \to 0$, and arrive at the *k-th intermediate reduced system* on $\mathcal{M}_{k-1}$:

$$\begin{aligned} 0 &= f_1(z,0) \\ &\vdots \\ 0 &= f_{k-1}(z,0) \\ \frac{\mathrm{d}z_k}{\mathrm{d}\tau_k} &= f_k(z,0) \\ \frac{\mathrm{d}z_{k+1}}{\mathrm{d}\tau_k} &= 0 \\ &\vdots \\ \frac{\mathrm{d}z_m}{\mathrm{d}\tau_k} &= 0. \end{aligned} \tag{2.13}$$

We call the m-th intermediate reduced system the *completely reduced system.*

The following theorem is adapted from [51, Theorem 1]. For detailed statements and proofs, see [18].

Theorem 2.2. *Let system* (2.8) *be given and $\mathcal{M}_k$ hyperbolically attractive on $\mathcal{M}_{k-1}$. Then for all sufficiently small $\bar{\varepsilon}$, system* (2.8) *admits an invariant manifold $\mathcal{N}_k^{\bar{\varepsilon}}$ that is* $(\varepsilon_1 + \cdots + \varepsilon_{m-1})$*-close to $\mathcal{M}_k$ with respect to the Hausdorff distance. Moreover, there exists $T > 0$ such that solutions of system* (2.8) *on $\mathcal{N}_k^{\bar{\varepsilon}}$ in time scales τ_{k+1} converge to solutions of* (2.12) *for $k = 0$ and* (2.13) *for $k > 1$, uniformly on any closed subinterval of* $(0, T)$*, as $\bar{\varepsilon} \to 0$.*

2.2 From Tropicalization to Cardin–Teixeira

This section describes how to use a scaled system as derived in Section 1.5 and transform it such that it can be properly used with the theory of Cardin and Teixeira. This section was previously published in [51, Sections 2 & 3.1].

2.2.1 Partitioning and Truncating a Scaled System

In Section 1.5, we described a general scaling procedure for a system of polynomial ordinary differential equations that culminated in (1.28), which we repeat here for convenience:

$$y_k' = \frac{\mathrm{d}y_k}{\mathrm{d}\tau} = \varepsilon^{\nu_k - \mu} \sum_J \varepsilon^{c_{k,J} + \langle D, J\rangle - d_k - \nu_k}\, \bar{\gamma}_{k,J}\, y^J, \quad 1 \le k \le n. \tag{2.14}$$

We restructure (2.14) by collecting all variables with equal $\nu_i - \mu$ in vectors $z_1, \ldots, z_m$, where $z_k \in \mathbb{R}^{n_k}$ for $k \in \{1, \ldots, m\}$, in ascending order of exponents and such that $n_1 + \cdots + n_m = n$. We obtain a system of the form

$$\left.\begin{aligned} z_k' &= \varepsilon^{a_k} \tilde{f}_k(z, \varepsilon) \\ &= \varepsilon^{a_k} \left(\tilde{f}_k(z, 0) + \varepsilon^{a'_{k,2}} p_{k,2} + \cdots + \varepsilon^{a'_{k,w_k}} p_{k,w_k}\right) \\ &= \varepsilon^{a_k} \left(\tilde{f}_k(z, 0) + o(1)\right) \end{aligned}\right\} \quad 1 \le k \le m, \tag{2.15}$$

where a_k, $a'_{k,j} \in \mathbb{Q}$, $0 = a_1 < a_2 < \ldots < a_m$, $0 < a'_{k,j}$, and $p_{k,j}$ are multivariate polynomials in z for $1 \le k \le m$ and $2 \le j \le w_k$. Note that the case $m = 1$ is not excluded. By substituting $\delta := \varepsilon^{1/q}$ with a sufficiently large positive integer q, one can ensure that only nonnegative integer powers of δ appear:

$$\left.\begin{aligned} z_k' &= \delta^{b_k} f_k(z, \delta) \\ &= \delta^{b_k} \left(f_k(z, 0) + \delta^{b'_{k,2}} p_{k,2} + \cdots + \delta^{b'_{k,w_k}} p_{k,w_k}\right) \\ &= \delta^{b_k} \left(f_k(z, 0) + o(1)\right) \end{aligned}\right\} \quad 1 \le k \le m, \tag{2.16}$$

where b_k, $b'_{k,j} \in \mathbb{N}$, $0 = b_1 < b_2 < \ldots < b_m$, $0 < b'_{k,j}$ for $1 \le k \le m$ and $2 \le j \le w_k$.

Our idea is that the indices k correspond to different time scales $\delta^{b_k}\tau$. For $m > 1$, system (2.16), as $\delta \to 0$, may be thought of as separating fast variables from increasingly slow ones. As it will turn out in Section 2.2.3, the exact number of time scales finally obtained by our overall approach can actually be smaller than m.

Given the conditions made explicit in Theorem 2.2 and with their application in the rest of this section, we may formally truncate the right hand sides of (2.16) and keep only terms of lowest order in δ:

$$z_k' = \delta^{b_k} f_k(z, 0), \quad 1 \le k \le m. \tag{2.17}$$

In the following, we refer to the transformation process from (1.22) to (1.28) and from (2.14) to (2.16) as *scaling*. Strictly speaking, this comprises scaling in combination with *partitioning*. We refer to the step from (2.16) to (2.17) as *truncating*.

2.2.2 Scaling via Tropical Geometry

The transformations leading to (1.26) are a formal exercise. No particular strategy was applied for choosing ε_*. Instead, we choose the value of ε_* freely to provide "power" parametric descriptions of all the quantities occurring in the differential equations (parameters, monomials, and time scales).

Given ε_*, we explain how to obtain the orders $c_{k,J}$ and D introduced in the Section 1.5 with (1.23) and (1.26), respectively. The orders $c_{k,J}$ are computed from $\varepsilon_* \in (0,1)$ and $\gamma_{k,J}$ as

$$c_{k,J} = \frac{\operatorname{round}(p \log_{\varepsilon_*} |\gamma_{k,J}|)}{p}. \tag{2.18}$$

The function round : $\mathbb{R} \to \mathbb{Z}$ rounds to nearest, ties to even, in the sense of IEEE 754 [43]. The positive integer p controls the precision of the rounding step. Using $\bar{\gamma}_{k,J} = \gamma_{k,J}/{\varepsilon_*}^{c_{k,J}}$ as defined in (1.23), our definition satisfies the constraint $\sqrt{{\varepsilon_*}^p} \le |\bar{\gamma}_{k,J}| \le \sqrt{1/{\varepsilon_*}^p}$. The orders $D = (d_1, \dots, d_n)$ satisfy certain constraints as well. Those constraints result heuristically from the idea of compensation of dominant monomials [75]. Slow dynamics is possible if for each dominant, i.e., much larger than the other, monomial on the right hand side of (1.28), there is at least one other monomial of the same order but with opposite sign. This condition, named *tropical equilibration condition* [75, 76, 81, 82, 94, 93], reads

$$\min_{\gamma_{k,J}>0}(c_{k,J} + \langle D, J\rangle) = \min_{\gamma_{k,J'}<0}(c_{k,J'} + \langle D, J'\rangle). \tag{2.19}$$

On these grounds, given system (1.22), the choice of ε_* boils down to defining orders of magnitude. Model parameters are coarse-grained and transformed to orders of magnitude in order to apply tropical scaling. The result depends on which parameters are close and which are very different as dictated by the coarse-graining procedure, i.e., by the choice of ε_*. Decreasing ε_* destroys details, and parameters tend to have the same order of magnitude. Increasing ε_* refines details, and parameters range over several orders of magnitude.

On the one hand, we have just noted that smaller choices of ε_* possibly hide details. On the other hand, in Chapter 2 we reviewed singular perturbation methods, which provide asymptotic results as a small parameter δ approaches zero. Following the construction in Section 1.5, small choices of ε_* lead to small δ, which gives a heuristic argument for choosing ε_* rather small. Thus, in practice one has to reconcile two competing requirements, which unfortunately still requires some human intuition.

2.2.3 Connecting to the Cardin–Teixeira Black Box

The general case as described in Section 2.1.2 assumes that hyperbolic attractivity can be verified throughout all sets $\mathcal{M}_1, \ldots, \mathcal{M}_{m-1}$. In applications, it turns out that hyperbolic attractivity can often be verified for only $\ell < m$ sets. Yet, the rest of the system needs to be represented as well.

Furthermore, [18] and Section 2.1.2 work with $\bar{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_{m-1})$, i.e., a tuple of parameters, whereas our input system (2.16) is written with a single parameters ε with multiple different exponents. Connecting these different expositions is the content of this section.

We consider our system (2.16) over the positive first orthant $\mathcal{U} = (0, \infty)^n \subset \mathbb{R}^n$. Let $\ell \in \{2, \ldots, m\}$ and

$$\beta_1 := b_2 - b_1 = b_2, \quad \ldots, \quad \beta_{\ell-1} := b_\ell - b_{\ell-1}. \tag{2.20}$$

Furthermore, let $\delta_1 := \delta^{\beta_1}, \ldots, \delta_{\ell-1} := \delta^{\beta_{\ell-1}}$, and $\bar{\delta} = (\delta_1, \ldots, \delta_{\ell-1})$.

These definitions allow us to express also all $\delta^{b'_{k,j}}$ occurring in (2.16) as products of powers of $\delta_1, \ldots, \delta_{\ell-1}$, with nonnegative but possibly non-integer rational exponents, via expressing each $b'_{k,j}$ as a nonnegative rational linear combination of $\beta_1, \ldots, \beta_{\ell-1}$. We introduce

$$g_k(z, \bar{\delta}) := f_k(z, \delta), \quad 1 \leq k \leq m. \tag{2.21}$$

Moreover, we express

$$\delta^{b_{\ell+1}} = \delta_1 \cdots \delta_{\ell-1} \cdot \eta_{\ell+1}(\bar{\delta}), \quad \ldots, \quad \delta^{b_m} = \delta_1 \cdots \delta_{\ell-1} \cdot \eta_m(\bar{\delta}),$$

via

$$\eta_k(\bar{\delta}) := \delta^{b_k - b_\ell}, \quad \ell + 1 \leq k \leq m, \tag{2.22}$$

which is obtained by writing each $b_k - b_\ell$ as a nonnegative rational linear combination of $\beta_1, \ldots, \beta_{\ell-1}$. In these terms our system (2.16) translates to

$$\begin{aligned} z'_1 &= g_1(z, \bar{\delta}) \\ &\vdots \\ z'_\ell &= \delta_1 \cdots \delta_{\ell-1} g_\ell(z, \bar{\delta}) \\ z'_{\ell+1} &= \delta_1 \cdots \delta_{\ell-1} \eta_{\ell+1}(\bar{\delta}) g_{\ell+1}(z, \bar{\delta}) \\ &\vdots \\ z'_m &= \delta_1 \cdots \delta_{\ell-1} \eta_m(\bar{\delta}) g_m(z, \bar{\delta}). \end{aligned} \tag{2.23}$$

In terms of the right hand sides of (2.23), the application of relevant results in [18] requires that $g_1, \ldots, g_\ell$ and $\eta_{\ell+1} g_{\ell+1}, \ldots, \eta_m g_m$ are smooth on an open neighborhood of $\mathcal{U} \times [0, \vartheta_1) \times \cdots \times [0, \vartheta_{\ell-1})$ with $\vartheta_1 > 0, \ldots, \vartheta_{\ell-1} > 0$. We are going to tacitly assume such smoothness here and address this issue from an algorithmic point of view in Section 2.3.

We are now ready to transform our system into ℓ time scales as follows:

$$\tau_1 = \tau, \quad \tau_2 = \delta_1 \tau, \quad \ldots, \quad \tau_\ell = \delta_1 \cdots \delta_{\ell-1} \tau.$$

In time scale τ_k system (2.23) then becomes

$$\begin{aligned}
\delta_1 \cdots \delta_{k-1} \frac{\mathrm{d}z_1}{\mathrm{d}\tau_k} &= g_1(z, \bar{\delta}) \\
&\vdots \\
\delta_{k-1} \frac{\mathrm{d}z_{k-1}}{\mathrm{d}\tau_k} &= g_{k-1}(z, \bar{\delta}) \\
\frac{\mathrm{d}z_k}{\mathrm{d}\tau_k} &= g_k(z, \bar{\delta}) \\
\frac{\mathrm{d}z_{k+1}}{\mathrm{d}\tau_k} &= \delta_k g_{k+1}(z, \bar{\delta}) \\
&\vdots \\
\frac{\mathrm{d}z_\ell}{\mathrm{d}\tau_k} &= \delta_k \cdots \delta_{\ell-1}\, g_\ell(z, \bar{\delta}) \\
\frac{\mathrm{d}z_{\ell+1}}{\mathrm{d}\tau_k} &= \delta_k \cdots \delta_{\ell-1} \eta_{\ell+1}(\bar{\delta}) g_{\ell+1}(z, \bar{\delta}) \\
&\vdots \\
\frac{\mathrm{d}z_m}{\mathrm{d}\tau_k} &= \delta_k \cdots \delta_{\ell-1} \eta_m(\bar{\delta}) g_m(z, \bar{\delta}).
\end{aligned} \tag{2.24}$$

For $k = 1$ and $k = \ell$ we obtain empty products, which yield the neutral element 1, as usual.

Equation (2.24) constitutes a version of (2.17) that can used with the theory of Cardin and Teixeira as laid out in Section 2.1.2.

2.3 The Algorithmic Approach

This section contains the algorithms, formulated as program code, that perform automatic partitioning of an input ODE system into possibly several time scales and computation of reduced systems. This section was previously published in [51]. In the following, each algorithm is discussed shortly and its connection to the theory is highlighted. Figure 2.1 depicts the flow of information between the different algorithms.

The general procedure for finding time scales and singular perturbation reduction is as follows: systems of type (1.22) or (2.14), respectively, are scaled and truncated with Algorithm 1 as per Section 1.5. In that process, Algorithms 2 and 3 are used to compute $c_{k,J}$ and D, respectively. Algorithm 3 further utilitizes Algorithm 4 to compute a disjunctive normal form (DNF) of polyhedra, which are the result of tropicalization. Obtaining an efficient algorithm for the computation of this DNF is another focal point of this thesis. An efficient algorithm will be discussed in Chapter 3 and leads to Algorithms 11 and 12.

Algorithm 5 returns reduced systems on invariant manifolds from an input of a scaled and truncated system. To that end, Algorithm 6 is utilized to verify hyperbolic

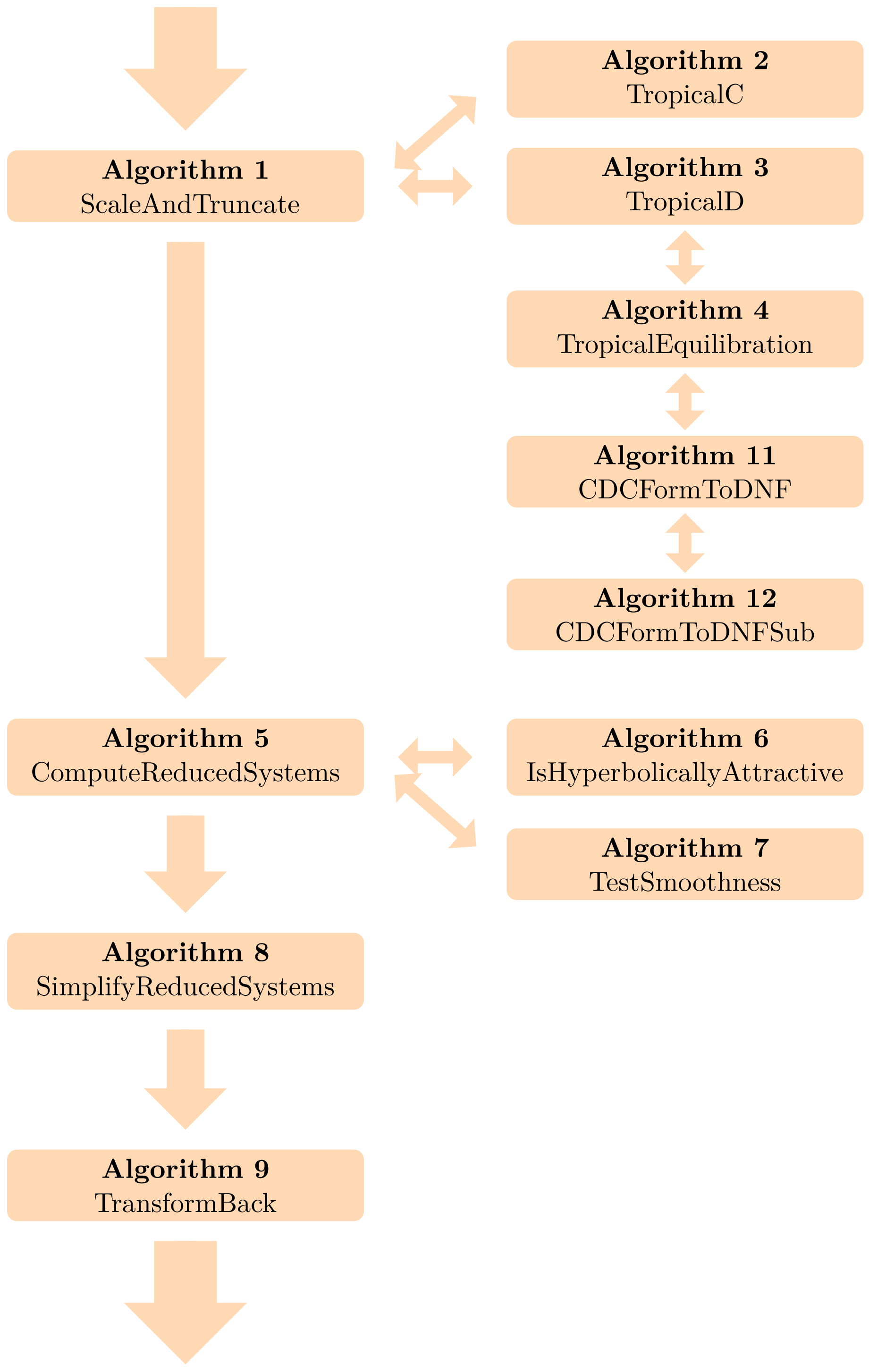


Figure 2.1: Principal data flow between our algorithms

attractivity as described in Section 2.1.2, and Proposition 2.2 in particular, and Algorithm 7 to verify smoothness conditions which were briefly mentioned after (2.23).

To return systems of the simplest possible form, Algorithm 8 uses techniques based on Gröbner bases to simplify later reduced systems based on definitions of invariant manifolds of earlier reduced systems. Finally, Algorithm 9 can be used to transform the resulting reduced systems back to the original time scale, and Algorithm 10 is a wrapper that combines all calls to the aforementioned algorithms into one function to provide an easy user interface.

2.3.1 **Algorithm 1:** ScaleAndTruncate

Algorithm 1 takes as input a list S of differential equations in variables $x_1, \dots, x_n$ representing system (1.22) and a choice of $\varepsilon_* \in (0,1)$ for (1.23). For our practical purposes, the polynomial coefficients in S as well as ε_* are taken from $\mathbb{Q}$. Our algorithm is furthermore parameterized with a function c mapping suitable indices to rational numbers and a constant function d yielding either a tuple $D = (d_1, \dots, d_n)$ of rational numbers or $\perp$. The black-box functions c and d reflect the mathematical assumptions around (1.23) and (1.26) that suitable $c_{k,J}$ and d_k exist, respectively. Suitable instantiations for the parameters c and d can be realized, e.g., using tropical geometry and example implementations can be found as Algorithms 2 and 3. It will turn out that instantiations of d can fail on a given combination of S and ε_*, which is signaled by the return value $\perp$ of d, and checked right away in l.1 of Algorithm 1.

The output is a list $[T_1, \dots, T_m]$ of truncated systems in variables $y_1, \dots, y_n$. Note that the order of δ is strictly increasing with each T_k and that the difference of the orders of T_k and T_{k+1} is an arbitrary positive integer. Furthermore, a list $[P_1, \dots, P_m]$ of higher-order terms is returned, as well as a substitution σ.

The output satisfies the following invariant: Denote $\widetilde{S} = (\bigcup_{k=1}^{m} T_k \oplus P_k)\sigma$, where $(y' = g) \oplus p$ stands for $y' = g + p$ and is applied elementwise. Then $\widetilde{S}$ is equal to S up to multiplication of the differential equation $\dot{x}_i = \sum_J \gamma_{i,J} x^J$ in S with a positive scalar factor $1/{\varepsilon_*}^{\mu + d_i}$.

2.3.2 **Algorithm 2:** TropicalC

Algorithm 2, which is only called by Algorithm 1, explicitly uses, besides the parameters k and J specified for c in Algorithm 1, also the right hand sides of the input system (1.22) and the choice of ε_*. As yet another parameter it takes the desired precision p for rounding in (2.18). Notice that the use of this extra information is compatible with the abstract scaling procedure in Section 1.5. Currying [22] allows to use Algorithm 2 in place of c in a formally clean manner.

2.3.3 **Algorithm 3:** TropicalD

Algorithm 3 is used to compute $D = (d_1, \dots, d_n)$, which was introduced in (1.25). The idea of D is to get a scaling for the variables $x_1, \dots, x_n$, such that the renormalized variables $y_1, \dots, y_n$ are more or less of the same order of magnitude. This must be, of course, only a guess which we hope will turn out useful in the end. Since the right

Algorithm 1 ScaleAndTruncate

Input:
1. A list $S = \left[\frac{\mathrm{d}x_1}{\mathrm{d}t} = f_1, \dots, \frac{\mathrm{d}x_n}{\mathrm{d}t} = f_n\right]$ of autonomous first-order ordinary differential equations where $f_1, \dots, f_n \in \mathbb{Q}[x_1, \dots, x_n]$;
2. $c : \{1, \dots, n\} \times \{1, \dots, n\}^n \to \mathbb{Q}$;
3. $d : () \to \mathbb{Q}^n \cup \{\perp\}$;
4. $\varepsilon_* \in (0, 1) \cap \mathbb{Q}$

Output:
1. A list $[T_1, \dots, T_m]$ where, abbreviating $\frac{\mathrm{d}}{\mathrm{d}\tau}$ by a prime, $T_k = (z'_k = \delta^{b_k} f_k)$ with $z'_k \subseteq [y'_1, \dots, y'_n]$, $\bigcup_k z'_k = [y'_1, \dots, y'_n]$, $z'_1, \dots, z'_m$ pairwise disjoint, $b_1 < \dots < b_m \in \mathbb{N}$, and $f_k \subseteq \mathbb{Q}[y_1, \dots, y_n]$, or the empty list;
2. A list $[P_1, \dots, P_m]$ of lists with $P_k \subseteq \mathbb{Q}[y_1, \dots, y_n][\delta]$ and $|P_k| = |T_k|$ for $k \in \{1, \dots, m\}$;
3. A substitution σ for $y_1, \dots, y_n, \tau, \delta$, and ε

The first output $[T_1, \dots, T_m]$ contains differential equations $z'_k = \delta^{b_k} f_k(z, 0)$ for $k \in \{1, \dots, m\}$ in terms of system (2.16). The second output $[P_1, \dots, P_m]$ contains the higher order terms in (2.16) as polynomials $p_k = \delta^{b_k + b'_{k,2}} p_{k,2} + \dots + \delta^{b_k + b'_{k,w_k}} p_{k,w_k}$. The last output is a substitution that undoes all substitutions applied for obtaining (2.16) from (1.22).

For $q \in \mathbb{Q}[y_1, \dots, y_n](\delta)$ we use $\deg_\delta(q)$ for the univariate degree of q in δ. Similarly, $\operatorname{tmon}_\delta(q)$ is the trailing monomial in δ.

```
1: if $d() = \perp$ then
2:     return [], [], []
3: end if
4: $\mu := \infty$
5: $q := 1$
6: $(d_1, \dots, d_n) := d()$                                   $\in \mathbb{Q}^n$
7: for $k := 1$ to $n$ do
8:     $h_k := 0$
9:     for all monomials $\gamma x^J$ in $f_k$ do
10:        $\bar{\gamma} := \gamma / {\varepsilon_*}^{c(k,J)}$                 $\in \overline{\mathbb{Q}}$
11:        $\eta := c(k, J) + \langle (d_1, \dots, d_n), J \rangle - d_k$     $\in \mathbb{Q}$
12:        $\mu := \min(\mu, \eta)$                              $\in \mathbb{Q}$
13:        $q := \operatorname{lcm}(q, \operatorname{denom} \eta)$                  $\in \mathbb{N} \setminus \{0\}$
14:        $h_k := h_k + \varepsilon^\eta \bar{\gamma} y^J$
15:    end for
16: end for
17: for $k := 1$ to $n$ do
18:    $h_k := h_k / \varepsilon^\mu$
19:    $h_k := h_k[\varepsilon \leftarrow \delta^q]$                       $\in \overline{\mathbb{Q}}[y_1, \dots, y_n][\delta]$
20:    $g_k := \operatorname{tmon}_\delta h_k$
21:    $p_k := h_k - g_k$
22: end for
23: $L := \left[\frac{\mathrm{d}y_1}{\mathrm{d}\tau} = g_1, \dots, \frac{\mathrm{d}y_n}{\mathrm{d}\tau} = g_n\right]$
24: $[b_1, \dots, b_m] := \operatorname{sort}(\deg_\delta g_1, \dots, \deg_\delta g_n)$, ascending and removing duplicates
25: for $k := 1$ to $m$ do
26:    $T_k := [\, \frac{\mathrm{d}y}{\mathrm{d}\tau} = g \in L \mid \deg_\delta g = b_k \,]$
27:    $P_k := [\, p_j \in \{p_1, \dots, p_n\} \mid \deg_\delta g_j = b_k \,]$
28: end for
29: $\sigma := [y_1 \leftarrow x_1/\varepsilon^{d_1}, \dots, y_n \leftarrow x_n/\varepsilon^{d_n}] \circ [\tau \leftarrow \varepsilon^\mu t] \circ [\delta \leftarrow \varepsilon^{1/q}] \circ [\varepsilon \leftarrow \varepsilon_*]$
30: return $[T_1, \dots, T_m]$, $[P_1, \dots, P_m]$, $\sigma$
```

Algorithm 2 TropicalC

```
Input:  1. $k \in \{1, \dots, n\}$;
        2. $J \in \{1, \dots, n\}^n$;
        3. A list $S = [\dot{x}_1 = f_1, \dots, \dot{x}_n = f_n]$ of autonomous first-order ordinary differential equations
           where $f_1, \dots, f_n \in \mathbb{Q}[x_1, \dots, x_n]$;
        4. $\varepsilon_* \in (0, 1) \cap \mathbb{Q}$.
        5. $p \in \mathbb{N} \setminus \{0\}$
Output: $c \in \mathbb{Q}$
1: $\gamma :=$ coeff$(f_k, x^J)$                                    $\in \mathbb{Q}$
2: $c :=$ round$(p \log_{\varepsilon_*} |\gamma|)/p$                  $\in \mathbb{Q}$
3: return $c$
```

Algorithm 3 TropicalD

```
Input:  1. A list $S = [\dot{x}_1 = f_1, \dots, \dot{x}_n = f_n]$ of autonomous first-order ordinary differential
           equations where $f_1, \dots, f_n \in \mathbb{Q}[x_1, \dots, x_n]$;
        2. $\varepsilon_* \in (0, 1) \cap \mathbb{Q}$.
        3. $p \in \mathbb{N} \setminus \{0\}$
Output: $(d_1, \dots, d_n) \in \mathbb{Q}^n \cup \{\bot\}$
1: $\Pi(a_1, \dots, a_n) :=$ TropicalEquilibration$(S, \varepsilon_*, p)$
2: if not $\mathbb{R} \models \exists a_1 \dots \exists a_n \Pi$ then
3:     return $\bot$
4: end if
5: $(d_1, \dots, d_n) :=$ one possible choice for $a_1, \dots, a_n$
6: return $(d_1, \dots, d_n)$
```

hand sides of functions that make up the input list S are polynomials, there is no limit to their values and as such a scaling D will most likely not be valid for all possible values of the domain. However, in many applications it turns out that the approach of equilibrating dominant monomials yields values that provide meaningful invariant manifolds and reductions.

Algorithm 3 takes parameters ε_* and p, while d is specified in Algorithm 1 to have no parameters at all. In l.1 we use Algorithm 4 as a sub-algorithm for tropical equilibration. One obtains a disjunctive normal form Π, which explicitly describes a set $\mathcal{P} = \{ p \in \mathbb{Q}^n \mid \Pi(p) \}$ as a finite union of convex polyhedra, as known from tropical geometry. Every $(d_1, \dots, d_n) \in \mathcal{P}$ satisfies (2.19). The satisfiability condition in l.2 tests whether $\mathcal{P} = \varnothing$. We employ *Satisfiability Modulo Theories (SMT)* solving [73] using the logic `QF_LRA` [6] for quantifier-free linear real arithmetic. The set $\mathcal{P}$ can be empty, e.g, when all monomials on the right hand side of some differential equation have the same sign and hence equilibration is impossible. Such an exceptional situation is signaled with a return value $\bot$ in l.3. In the regular case $\mathcal{P} \neq \varnothing$, the choice $(d_1, \dots, d_n)$ in l.5 is provided by the SMT solver.

2.3.4 Algorithm 4: TropicalEquilibration

Algorithm 4 takes as input a system S of differential equations with a polynomial vector field, together with parameters ε_* and p, and tries to equilibrate the monomials of each

Algorithm 4 TropicalEquilibration

Input: 1. A list $S = [\dot{x}_1 = f_1, \dots, \dot{x}_n = f_n]$ of autonomous first-order ordinary differential equations where $f_1, \dots, f_n \in \mathbb{Q}[x_1, \dots, x_n]$;

2. $\varepsilon_* \in (0,1) \cap \mathbb{Q}$;

3. $p \in \mathbb{N} \setminus \{0\}$.

Output: A formula $\Pi(a_1, \dots, a_n)$ describing a finite union of convex polyhedra in $\mathbb{R}^n$.

```
1: $A_0 := (1, a_1, \dots, a_n)$                                   $\in \mathbb{Q}[a_1, \dots, a_n]^{n+1}$
2: for $j := 1$ to $n$ do
3:     $c := 0$
4:     for all monomials $\gamma x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ in $f_j$ do
5:         $\alpha_0 := \mathrm{round}(p \log_{\varepsilon_*} |\gamma|)/p$                 $\in \mathbb{Q}$
6:         $c := c + 1$
7:         $\Sigma_c := \operatorname{sgn} \gamma$                                   $\in \{-1, 1\}$
8:         $A_c := (\alpha_0, \alpha_1, \dots, \alpha_n)$                       $\in \mathbb{Q} \times \mathbb{Z}^n \subseteq \mathbb{Q}^{n+1}$
9:     end for
10:    $B_j := \varnothing$
11:    for $k := 1$ to $c$ do
12:        for $\ell := k + 1$ to $c$ do
13:            if $\Sigma_k \Sigma_\ell < 0$ then
14:                $P := \{\langle A_k - A_\ell, A_0 \rangle = 0\}$          $\langle A_k - A_\ell, A_0 \rangle \in \mathbb{Q}[a_1, \dots, a_n]$
15:                for $m := 1$ to $c$ do
16:                    $P := P \cup \{\langle A_m - A_k, A_0 \rangle \geq 0\}$    $\langle A_m - A_k, A_0 \rangle \in \mathbb{Q}[a_1, \dots, a_n]$
17:                end for
18:                $B_j := B_j \cup \{P\}$                        set of sets of constraints
19:            end if
20:        end for
21:    end for
22: end for
23: $\Pi := \mathrm{CDCFormToDNF}(\bigwedge_{j=1}^{n} \bigvee_{P \in B_j} \bigwedge P)$
24: return $\Pi$
```

polynomial using tropical geometry. For that, it builds a set B_j of polyhedra P for each polynomial and computes the intersection of all such B_j, which is then transformed into the disjunctive normal form Π. This transformation can be very time-consuming and Chapter 3 contains an in-depth discussion of the novel Algorithm 11 for that task.

Notice that, if a polynomial f_j does not contain monomials with both positive and negative signs, the corresponding B_j is empty. In consequence, this leads to a DNF Π that is logically equivalent to false. Furthermore, we tacitly assume that $f_j = 0$ leads to no execution of the loop body in l.4–9. Thus, $c = 0$ and $B_j = \varnothing$, leading also to $\Pi \equiv \mathsf{false}$.

With applications in the natural sciences one often wants to make in l.5 an adequate choice for $(d_1, \dots, d_n)$ lying in a specific convex polyhedron $P \subseteq \mathcal{P}$, which technically corresponds to one conjunction in Π. Such choices are subtle and typically require human interaction. For instance, when the chain of reduced dynamical systems ends with a steady state, it is interesting to consider the polyhedron P that is closest to that steady state. Such strategies are not covered by our algorithms presented here.

Algorithm 5 ComputeReducedSystems

Input: Output of Algorithm 1:

1. $[T_1, \dots, T_m]$, a list of lists $z'_k = \delta^{b_k} f_k$;
2. $[P_1, \dots, P_m]$, a list of lists of polynomials in $\mathbb{Q}[y_1, \dots, y_n][\delta]$;

We denote $\xi_k := |T_k|$, $\Xi_k := \sum_{i=1}^{k} \xi_i$, and $Y = (y_1, \dots, y_n)$.

Output: A list $[(M_0, T_1, R_1), \dots, (M_{\ell-1}, T_\ell, R_\ell)]$ of triplets where $\ell \in \{2, \dots, m\}$, or the empty list. For $k \in \{1, \dots, \ell\}$, M_{k-1} is a list of real constraints defining $\mathcal{M}_{k-1} \subseteq \mathbb{R}^n$; T_k is a list of differential equations; R_k is a list of trivial differential equations $y' = 0$ for all differential variables from T_{k+1}, $\dots$, T_m.

The triplets (M_{k-1}, T_k, R_k) represent reduced systems according to (2.25).

```
1: $U := [y_1 > 0, \dots, y_n > 0]$
2: $M_0, Z, F := [\,]$
3: $A := (\,)$
4: for $k := 1$ to $m$ do
5:     $z := [\, y \mid y' = \delta^{b_k} g \in T_k \,]$                          $\subseteq Y,\ |z| = \xi_k$
6:     $f := [\, g \mid y' = \delta^{b_k} g \in T_k \,]$                          $= f_k(z, 0) \in \mathbb{Q}[Y]^{\xi_k}$
7:     $M_k := M_{k-1} \circ [f = 0]$                                      $= M_0 \circ [F = 0] \circ [f = 0]$
8:     $\varphi, A :=$ IsHyperbolicallyAttractive$(U \circ M_k, Z, z, F, f, k, A)$
9:     if not $\varphi$ then
10:        break
11:    end if
12:    $R_k := [\, y' = 0 \mid y' = h \in T_{k+1} \cup \dots \cup T_m \,]$            $\Xi_{k-1} + \xi_k + |R_k| = n$
13:    $Z := Z \circ z$                                                  $\subseteq Y,\ |Z| = \Xi_k$
14:    $F := F \circ f$                                                  $\in \mathbb{Q}[Y]^{\Xi_k}$
15: end for
16: # We either broke in line 10 preserving k, or we have $k = m + 1$.
17: $\ell := k - 1$
18: if $\ell < 2$ then
19:    return ()
20: end if
21: if TestSmoothness$([T_1, \dots, T_m], [P_1, \dots, P_m], \ell)$ = failed then
22:    print "Warning: differentiability requires further verification"
23: end if
24: return $[(M_0, T_1, R_1), \dots, (M_{\ell-1}, T_\ell, R_\ell)]$
```

2.3.5 Algorithm 5: ComputeReducedSystems

Algorithm 5 takes as input parts of the output of Algorithm 1 and tests for hyperbolic attractivity as described in Section 2.1.2, especially Proposition 2.2. This test is done incrementally by going through the scaled input systems in order of their time scales. If the test for hyperbolic attractivity on one system fails, the procedure is terminated and the chain of verified reduced systems is returned.

This and later algorithms use the notion of *triplets* (M_{k-1}, T_k, R_k) which contain entries as follows:

$$F_{k-1}(z, 0) = 0, \quad \frac{\mathrm{d}z_k}{\mathrm{d}\tau} = \delta^{b_k} f_k(z, 0), \quad \frac{\mathrm{d}z_{k+1}}{\mathrm{d}\tau} = \dots = \frac{\mathrm{d}z_m}{\mathrm{d}\tau} = 0. \tag{2.25}$$

Each of these triplets describes a reduced system in the respective time scale.

Algorithm 5 starts with the output $[T_1, \dots, T_m]$ of Algorithm 1, which represents the scaled system (2.17). Notice that each T_k already meets the specification in (2.25). In

Algorithm 6 IsHyperbolicallyAttractive

Input: 1. M, 2. Z, 3. z, 4. F, 5. f, 6. k, 7. A, as in the calling Algorithm 5
Knowing that $\mathcal{M}_0 \rhd \cdots \rhd \mathcal{M}_{k-1}$, we check here whether also $\mathcal{M}_{k-1} \rhd \mathcal{M}_k$. We denote $\xi := |f| = |z|$, $\Xi := |F| = |Z|$, and $X = \{y_1, \dots, y_n\}$. In these terms, $A \in \mathbb{Q}[X]^{\Xi\times\Xi}$.

Output: 1. Boolean, 2. $A' \in \mathbb{Q}[X]^{(\Xi+\xi)\times(\Xi+\xi)}$

```
1: if not $\mathbb{R} \models \exists \bigwedge M$ then
2:     return false, $(\,)$
3: end if
4: $V :=$ Jacobian$(f, z)$                                    $\in \mathbb{Q}[X]^{\xi\times\xi}$
5: if $k = 1$ then
6:     $W := V$
7:     $A' := V$
8: else
9:     $B :=$ Jacobian$(F, z)$                                $\in \mathbb{Q}[X]^{\Xi\times\xi}$
10:    $C :=$ Jacobian$(f, Z)$                                $\in \mathbb{Q}[X]^{\xi\times\Xi}$
11:    $W := V - CA^{-1}B$                                    $\in \mathbb{Q}[X]^{\xi\times\xi}$
12:    $A' := \begin{pmatrix} A & B \\ C & V \end{pmatrix}$                 $\in \mathbb{Q}[X]^{(\Xi+\xi)\times(\Xi+\xi)}$
13: end if
14: $\chi := \lambda^\xi + \cdots + a_\xi :=$ CharacteristicPolynomial$(W)$      $\in \mathbb{Q}[X][\lambda]$
15: $H :=$ HurwitzMatrix$(\chi)$                              $\in \mathbb{Q}[X]^{\xi\times\xi}$
16: for $j := 1$ to $\xi - 1$ do
17:     $\Delta_j := \det\left(H_{r,s}\right)_{1\le r,s\le j}$                 $\in \mathbb{Q}[X]$
18: end for
19: $\Gamma := \{\Delta_1 > 0, \dots, \Delta_{\xi-1} > 0, a_\xi > 0\}$
20: return $\mathbb{R} \models \forall(\bigwedge M \longrightarrow \bigwedge \Gamma)$, $A'$
```

l.1 we define U to contain defining inequalities of the first orthant $\mathcal{U}$. Starting with $k = 1$, the for-loop in l.4–15 successively constructs M_k and R_k such that in combination with T_k from the input, (M_{k-1}, T_k, R_k) forms a reduced system as in (2.25). The loop stops when either $k = m + 1$ or the test in l.8 finds no hyperbolic attractivity. We are going to discuss this test in detail in the next section. Note that we maintain a matrix A for storing information between the subsequent calls of our test. In either case we arrive at a maximal hyperbolically attractive $(k-1)$-chain of reduced systems given as a list $[(M_0, T_1, R_1), \dots, (M_{k-2}, T_{k-1}, R_{k-1})]$. The variable ℓ contains the length of this chain, so we set ℓ to $k - 1$ in l.17. The test in l.18 reflects the choice of $\ell \in \{2, \dots, m\}$ at the beginning of this section. Finally, l.21 uses the second input $[P_1, \dots, P_m]$ of the algorithm to address the smoothness requirements for system (2.23). We are going to discuss the corresponding procedure in detail in Section 2.3.7. It will turn out that this procedure provides only a sufficient test. Therefore we issue only a warning in case of failure, allowing the user to verify smoothness a posteriori, using alternative algorithms or human intelligence. One might mention that it is actually sufficient to consider weaker, finite differentiability conditions instead of smoothness, which can be seen by inspection of the proofs in [18].

2.3.6 Algorithm 6: IsHyperbolicallyAttractive

Algorithm 6 tests for $\mathcal{M}_{k-1} \rhd \mathcal{M}_k$ assuming that $\mathcal{M}_0 \rhd \cdots \rhd \mathcal{M}_{k-1}$ holds. It follows Proposition 2.2 (i).

In l.1 we test the non-emptiness condition imposed after (2.9). Using from the input the defining inequalities and equations $M = U \circ M_k$ of $\mathcal{M}_k$ along with $Z = Z_{k-1}$, $z = z_k$, $F = F_{k-1}$, $f = f_k$, and $A = A_{k-1}$, we construct in l.4–13 $A' = A_k$ as noted in Proposition 2.2. To test for negative real parts of the eigenvalues of W as per (i) of Proposition 2.2, we use a test by Hurwitz [42]. In l.14–20 we construct the conditions Γ for this test and perform the test itself. We additionally return $A' = A_k$ for reuse with the next iteration. The validity tests in l.1 and l.20, respectively, again amount to SMT solving, this time using the logic `QF_NRA` [6] for quantifier-free nonlinear real arithmetic. Recall the positive integer parameter p used for the precision with both Algorithm 2 and Algorithm 3. For $p > 1$, symbolic computation possibly yields fractional powers of numbers in the defining equations for manifolds as well as in the vector fields of the differential equations. Such expressions are not covered by `QF_NRA`. When this happens, we catch the corresponding error from the SMT solver and restart with floats.

2.3.7 Algorithm 7: TestSmoothness

Algorithm 7 tests the smoothness conditions referred to in Section 2.2.3. $g_1, \dots, g_\ell$ and $\eta_{\ell+1}g_{\ell+1}, \dots, \eta_m g_m$ occurring on the right hand sides of system (2.23) all have to be smooth on an open neighborhood of $\mathcal{U} \times [0, \vartheta_1) \times \cdots \times [0, \vartheta_{\ell-1})$ with $\vartheta_1 > 0$, $\dots$, $\vartheta_{\ell-1} > 0$. A sufficient criterion for smoothness is that all those expressions are polynomials in z and $\bar{\delta}$. Note that smoothness has to hold for the original system (2.16) and not only for the scaled and truncated system (2.17).

Algorithm 7 specifies the sufficiency test applied in l.21 of Algorithm 5. The first two parameters $[T_1, \dots, T_m]$ and $[P_1, \dots, P_m]$ originate from Algorithm 1, while the last parameter ℓ originates from Algorithm 5.

In l.1–8 of Algorithm 7 we compute $\beta_1, \dots, \beta_{\ell-1}$ as defined in (2.20) and simultaneously obtain $b_1, \dots, b_\ell$. In l.9–14 we compute the set of powers of some δ_k that is contained in (2.16). For checking those conditions in l.16 we once more employ SMT solving, this time using the adequate logic `QF_LIA` [6] for quantifier-free linear integer arithmetic. Since we are aiming at nonnegative integer solutions, we introduce explicit non-negativity conditions $r_1 \geq 0, \dots, r_{\ell-1} \geq 0$. In case of unsatisfiability, Algorithm 7 returns "failed" in l.17. Recall that in this case the calling Algorithm 5 issues a warning but continues. In case of satisfiability, in contrast, smoothness is guaranteed, we reach l.20, and return "true." We remark that the computation time spent on E is negligible compared to the SMT solving later on. The construction of the entire set E beforehand avoids duplicate SMT instances.

2.3.8 Algorithm 8: SimplifyReducedSystems

Algorithm 8 employs Gröbner basis techniques [17, 8] to possibly simplify the defining equations for reduced systems.

Recall that z_k are the variables occurring on the left hand sides of differential equations in T_k, and $Z_{k-1} = (z_1, \dots, z_{k-1})$. In l.1–5 we construct a block term order ω on all variables $\{y_1, \dots, y_n\}$ so that variables from Z_{k-1} are larger than variables from z_k. This ensures that all multivariate polynomial reductions with respect to ω throughout our algorithm will eliminate variables from Z_{k-1} in favor of variables from z_k rather

Algorithm 7 TestSmoothness

Input: $[T_1, \dots, T_m]$, $[P_1, \dots, P_m]$, ℓ as in the calling Algorithm 5:

1. $[T_1, \dots, T_m]$, a list of lists $z'_k = \delta^{b_k} f_k$;
2. $[P_1, \dots, P_m]$, a list of lists of polynomials in $\mathbb{Q}[y_1, \dots, y_n][\delta]$;
3. $\ell \in \mathbb{N}$, $\ell \geq 2$;

We check here a sufficient criterion for smoothness as required for (2.23).

Output: "true" or "failed" in terms of a 3-valued logic;

```
 1: $b_1 := 0$
 2: for $k := 2$ to $\ell$ do
 3:    $b_k :=$ the unique exponent of $\delta$ in $T_k$
 4:    $\beta_{k-1} := b_k - b_{k-1}$
 5:    if $\beta_{k-1} = 1$ then
 6:       return true
 7:    end if
 8: end for
 9: $E := \varnothing$
10: for $k = 1$ to $m$ do
11:    for all $p$ in $P_k$ do
12:       $E := E \cup \{ \deg_\delta m - b_{\min(k,\ell)} \mid m$ monomial of $p \}$        $\subseteq \mathbb{N} \setminus \{0\}$
13:    end for
14: end for
15: for all $e \in E$ do
16:    if not $\mathbb{Z} \models \exists r_1 \dots \exists r_{\ell-1} (r_1 \geq 0 \wedge \dots \wedge r_{\ell-1} \geq 0 \wedge \langle (\beta_1, \dots, \beta_{\ell-1}), (r_1, \dots, r_{\ell-1}) \rangle = e)$ then
17:       return failed
18:    end if
19: end for
20: return true
```

Algorithm 8 SimplifyReducedSystems

Input: A list $[(M_0, T_1, R_1), \dots, (M_{\ell-1}, T_\ell, R_\ell)]$, the output of Algorithm 5, with entries corresponding to (2.25)

Output: A list $[(M'_0, T'_1, R_1), \dots, (M'_{\ell-1}, T'_\ell, R_\ell)]$; M'_{k-1} describes the same manifold as M_{k-1} in a canonical form; the system T'_k is equivalent to T_k modulo M'_{k-1}, its right hand sides are in a canonical normal form modulo M'_{k-1}, possibly with fewer different differential variables than T_k

```
 1: for $k := 1$ to $\ell$ do
 2:    $z_k := \{ y \mid y' = g \in T_k \}$
 3: end for
 4: $x := \{ y \mid y' = 0 \in R_\ell \}$
 5: $\omega :=$ a block term order with $z_1 \gg \dots \gg z_\ell \gg x$
 6: for $k := 1$ to $\ell$ do
 7:    $F := [ f \mid f = 0 \in M_{k-1} ]$
 8:    $G :=$ GroebnerBasis(Radical($F$), $\omega$)
 9:    $M'_{k-1} := [ g = 0 \mid g \in G ]$
10:    $T'_k := [\,]$
11:    for $y' = g$ in $T_k$ do
12:       $T'_k := T'_k \circ [y' = h]$ where $g \longrightarrow^*_G h$ and $h$ is irreducible mod $G$
13:    end for
14: end for
15: return $[(M'_0, T'_1, R_1), \dots, (M'_{\ell-1}, T'_\ell, R_\ell)]$
```

Algorithm 9 TransformBack

Input: 1. $[(M_0, T_1, R_1), \dots, (M_{\ell-1}, T_\ell, R_\ell)]$, the output of either Algorithm 5 or Algorithm 8;

2. σ, the output of Algorithm 1

Output: A list $[(M_0^*, T_1^*, R_1^*), \dots, (M_{\ell-1}^*, T_\ell^*, R_\ell^*)]$.

```
1: for $k := 1$ to $\ell$ do
2:     $M_{k-1}^* := M_{k-1}\sigma$
3:     $v := \big((\delta^{b_k}\tau)\sigma\big)/t$, extracting $\delta^{b_k}$ from $T_k$          $= {\varepsilon_*}^{(b_k/q)+\mu}$
4:     $T_k^* := [\,]$
5:     for all $y_j' = \delta^{b_k} f_j \in T_k$ do
6:         $h := (x_j f_j / y_j)\sigma$                                $= {\varepsilon_*}^{d_j}(f_j\sigma)$
7:         $T_k^* := T_k^* \circ [\dot{x}_j = vh]$                      $= {\varepsilon_*}^{b_k/q+\mu+d_j}(f_j\sigma)$
8:     end for
9:     $R_k^* := [\,\dot{x}_j = 0 \mid y_j' = 0 \in R_k\,]$
10: end for
11: return $[(M_0^*, T_1^*, R_1^*), \dots, (M_{\ell-1}^*, T_\ell^*, R_\ell^*)]$
```

than vice versa. Prominent examples for such block orders are pure lexicographic orders, but ordering by total degree inside the $z_1, \dots, z_\ell, x$ will heuristically give more efficient computations.

Recall that the radical ideal $\sqrt{\langle F\rangle}$ is the infinite set of *all* polynomials with the same common complex roots as F. In l.8, we compute a finite reduced Gröbner basis G with respect to ω of that radical. If radical computation is not available on the software side, then the algorithm remains correct with a Gröbner basis of the ideal $\langle F\rangle$ instead of the radical ideal, but might miss some simplifications.

In l.9, the polynomials in G equivalently replace the left hand side polynomials of the equations in M_{k-1}. In l.12, reduction with respect to ω, which comes with heuristic elimination of variables, applies once more to the reduction results h obtained from right hand sides g of differential equations in T_k. Since G is a Gröbner basis, the reduction in l.11–13 furthermore produces unique normal forms with the following property: if two polynomials g_1, g_2 coincide on the manifold $\mathcal{M}_{k-1}$ defined by M_{k-1}, then they reduce to the same normal form h. In particular, if g_1 vanishes on $\mathcal{M}_{k-1}$, then it reduces to 0. We call the output of Algorithm 8 *simplified reduced systems.*

2.3.9 Algorithm 9: TransformBack

Algorithm 9 transforms the reduced and possibly simplified systems back to their original time scales by applying the substitution σ, which is one output of Algorithm 1.

Our back-transformation is realized in Algorithm 9. In l.3 we compute the time scale factor ${\varepsilon_*}^{(b_k/q)+\mu}$ for T_k^* as described above, and in l.6 we compute its co-factor ${\varepsilon_*}^{d_j} f\sigma$ as $(x_j f/y_j)\sigma$.

2.3.10 Algorithm 10: TropicalMultiReduce

Algorithm 10 provides a wrapper combining all our algorithms to decompose input systems like (1.22) into several time scales. The underlying tropicalization is not made explicit, and the resulting reduced systems are presented on the original time scale.

Algorithm 10 TropicalMultiReduce

Input:
1. A list $S = [\dot{x}_1 = f_1, \dots, \dot{x}_n = f_n]$ of autonomous first-order ordinary differential equations where $f_1, \dots, f_n \in \mathbb{Q}[x_1, \dots, x_n]$;
2. $\varepsilon_* \in (0, 1) \cap \mathbb{Q}$;
3. $p \in \mathbb{N} \setminus \{0\}$

Output: A list $[(M_0^*, T_1^*, R_1^*), \dots, (M_{\ell-1}^*, T_\ell^*, R_\ell^*)]$ of triplets where $\ell \in \{2, \dots, m\}$, or the empty list. For $k \in \{1, \dots, \ell\}$, M_{k-1}^* is a list of real constraints defining $\mathcal{M}_{k-1}^* \subseteq \mathbb{R}^n$; T_k^* is a list of differential equations; R_k^* is a list of trivial differential equations $\dot{x} = 0$ for all differential variables from T_{k+1}^*, $\dots$, T_m^*.

The relevance of the output in terms of the input is discussed in Section 2.3.9.

```
1: $\mathrm{TropicalC}_{S,\varepsilon_*,p} := \mathrm{curry}(\mathrm{TropicalC}, S, \varepsilon_*, p)$        $\mathrm{TropicalC}_{S,\varepsilon_*,p}$ is a binary function
2: $\mathrm{TropicalD}_{S,\varepsilon_*,p} := \mathrm{curry}(\mathrm{TropicalD}, S, \varepsilon_*, p)$        $\mathrm{TropicalD}_{S,\varepsilon_*,p}$ is a constant function
3: $T, P, \sigma := \mathrm{ScaleAndTruncate}(S, \mathrm{TropicalC}_{S,\varepsilon_*,p}, \mathrm{TropicalD}_{S,\varepsilon_*,p}, \varepsilon_*)$
4: $\Sigma := \mathrm{ComputeReducedSystems}(T, P)$        $= [(M_0, T_1, R_1), \dots, (M_{\ell-1}, T_\ell, R_\ell)]$
5: $\Sigma' := \mathrm{SimplifyReducedSystems}(\Sigma)$        $= [(M_0', T_1', R_1), \dots, (M_{\ell-1}', T_\ell', R_\ell)]$
6: $\Sigma^* := \mathrm{TransformBack}(\Sigma', \sigma)$        $= [(M_0^*, T_1^*, R_1^*), \dots, (M_{\ell-1}^*, T_\ell^*, R_\ell^*)]$
7: **return** $\Sigma^*$
```

Chapter 3

Model-Driven Computation of Disjunctive Normal Forms

In Chapter 2 we presented an approach to scalings of polynomial differential equations based on tropical geometry. Algorithm 4, the corresponding algorithmic description, makes explicit the computation of the tropical equilibration as a conjunction of disjunctive normal forms of quantifier-free first-order formulas over the reals. Line 23 of Algorithm 4 then calls CDCFormToDNF, that is, Algorithm 11 to obtain a logically equivalent disjunctive normal form (DNF) Π.

Formally, the input to Algorithm 11 is the formula

$$\varphi = \bigwedge_{j=1}^{n} \bigvee_{P \in B_j} \bigwedge P, \tag{3.1}$$

where $B_1, \ldots, B_n$ are sets and each P is a set of real constraints describing a polyhedron. In other words, each B_j is a DNF and φ is a conjunction of DNFs.

The output of the computation is the DNF

$$\varphi' = \bigvee_{k} \gamma_k, \tag{3.2}$$

where each γ_k is a conjunction of constraints that describe a polyhedron. Thus, φ' describes a finite union of polyhedra. The relationship between input and output is that φ and φ' are logically equivalent.

The computation of the DNF in (3.2) is bound to be a bottleneck in many cases and requires good heuristic strategies. An ad-hoc solution [59] based on a polyhedral library proved to be too slow in many cases. To convey some idea of the size of possible inputs: the text input for Redlog describing the conjunction of DNFs of the tropical equilibration of `BIOMD0000000501` with $\varepsilon_* = \frac{1}{5}$ and $p = 1$ is 102 685 bytes long and the corresponding output describing 1665 polyhedra would take an estimated 9 MiB in Redlog format.

In the following, we present a calculus to compute a DNF from a conjunction of DNFs. For a less formal discussion, see [60]. Benchmarks of this algorithm against established software are presented in Chapter 5.

The content of this chapter is as follows: Section 3.1 contains some historical notes about the disjunctive normal form and its computation. Section 3.2 presents a motivating example of our approach of a model-driven computation. In Section 3.3, we present our approach in the form of a calculus. Termination, soundness, and minimality of the calculus are proven in Section 3.4. For the reader's convenience, Section 3.5 contains an algorithmic description of the calculus, and Section 3.6 contains some final remarks.

3.1 Historical Context

The disjunctive normal form received less early interest than its cousin, the conjunctive normal form. Peirce already wrote about the canonical conjunctive normal form in 1880, albeit without naming it [78]. The name *conjunctive normal form* was introduced by Bernays and published in 1926 [9, 10]. It seems that the name *disjunctive normal form* (DNF) came into use by analogy.

The design of logic circuits made it necessary to simplify DNFs so that they require as few logical operations as possible. To that end, Veitch introduced his *chart* [103], a rediscovery of Marquand's *logical diagram* [66]. The chart was improved by Karnaugh in 1953 [45] and led to the *Karnaugh–Veitch map.* This map allows a human in a visual process to find a minimal DNF of a given canonical DNF (CDNF). It works well for DNFs with up to four variables and can be extended to up to six variables, yet this already makes the visual process more difficult.

In contrast, Quine's method [80], improved by McCluskey in 1956 [67], has no such immediate limit on the number of variables and is well suited for implementation on computers. For a propositional formula with α variables, it requires first to build a complete truth table with 2^α entries. From this table, *prime implicants* are computed, which can also be exponential in number. Robinson's *resolution method* can be used to find prime implicants without the need for a CDNF [87]. In either case, a *set cover* problem has to be solved in a second step, which is known to be NP-complete [46].

Several improvements to the Quine–McCluskey method have been published, allowing larger problems to be solved [14, 90, 68]. Their main advantage is that they save computation time and space by improving the size of the prime implicants table or by only implicitly generating it [21].

While the method of Quine–McCluskey and its descendants are designed for propositional logic, Dolzmann and Sturm's approach supports real semantics [25] and does simplification of quantifier-free first-order formulas over ordered fields. Nevertheless, it also requires a CDNF to work on.

Satisfiability Modulo Theories (SMT) solving [73, 72] allows to test the satisfiability of first-order formulas. It has a multitude of applications, such as computer hardware and software verification [11], model checking [2, 7], scheduling, theorem proving [5], and test-case generation. SMT solving is an extension of SAT checking, which checks the satisfiability of a propositional formula in Boolean logic and which itself is NP-complete [20]. Even though SMT solving has the same complexity, it has become feasible in recent years for an increasing number of problems [64]. Well-known and well-supported SMT solvers are CVC4 [4], MathSAT [19], Yices [27], and Z3 [71], among others. Yices was the fastest solver for `QF_LRA` at SMT-COMP 2017 [84].

In this chapter, we present for the first time to our knowledge, a DNF computation

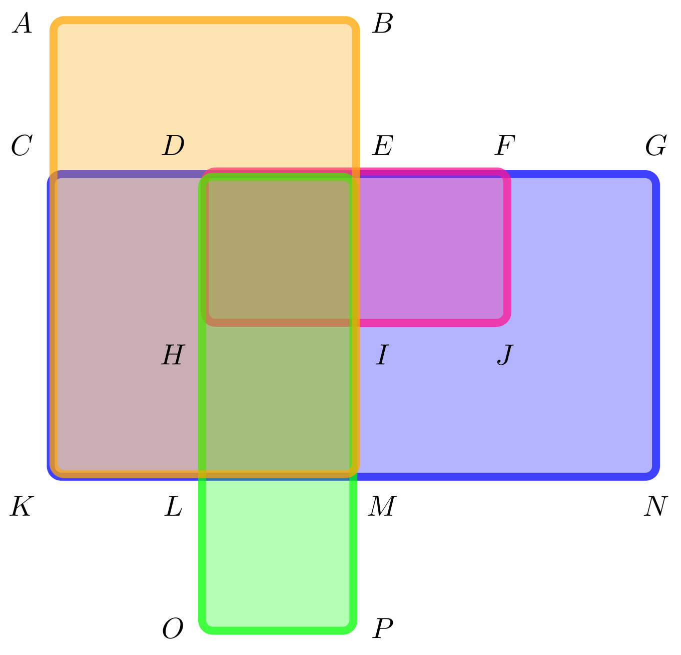


Figure 3.1: Four intersecting rectangles

of quantifier-free first-order formulas based on SMT solving. In contrast to the methods based on the Quine–McCluskey approach, our novel method does not need a CDNF to begin with, nor does it compute one in its process. Furthermore, it circumvents the exponential blow-up of conjunctions that comes with an ad-hoc approach. Of course, it cannot avoid some NP-complete component. Since our method is based on SMT solving, it has outsourced the NP-complete problem to the highly researched area of SMT solving and automatically benefits from improvements there.

3.2 Motivating Example

To motivate our idea we use a geometric example. Consider Figure 3.1, which depicts four rectangles. Each rectangle is defined by its vertices and we refer to it by them. The orange rectangle is $AKMB$, the blue one is $CKNG$, the green rectangle is $DOPE$, and the pink one is $DHJF$.[1] The four rectangles describe areas in 2D-space, which we identify with point sets. The boundaries of the rectangles are included in the described space.

We ask the following question: What is

$$(AKMB \cup CKNG) \cap (DOPE \cup DHJF)?$$

Or, expressed in terms of the colors of the rectangles: What is

$$(\text{orange} \cup \text{blue}) \cap (\text{green} \cup \text{pink})?$$

[1]If one looks at the image closely, one can observe that some rectangles are slightly shifted. This is only done to enhance the visual perception of the edges.

Table 3.1: Coordinates of Points in the Motivating Example

A	B	C	D	E	F	G	H
(0,4)	(2,4)	(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(1,2)
I	J	K	L	M	N	O	P
(2,2)	(3,2)	(0,1)	(1,1)	(2,1)	(4,1)	(1,0)	(2,0)

One way to answer this is to exploit the distributivity of union and intersection. We get

$$\begin{aligned}(AKMB \cup CKNG) \cap\\ (DOPE \cup DHJF) &= (AKMB \cap DOPE) \cup (AKMB \cap DHJF) \cup\\ &\quad (CKNG \cap DOPE) \cup (CKNG \cap DHJF)\\ &= DLME \cup DHIE \cup DLME \cup DHJF\\ &= DLME \cup DHIE \cup DHJF.\end{aligned} \tag{3.3}$$

This approach becomes impractical quickly if there are more unions with more rectangles, since an exponential number of intersections are generated.

Our novel approach uses SMT solving. To employ SMT solving, we need to convert the geometric representations of our rectangles into quantifier-free first-order formulas. The named points lie on a rectangular grid with coordinates as listed in Table 3.1. For this exposition, we identify rectangles with their describing formulas and use both descriptions interchangeably.

We convert all geometric rectangles to formulas and express union and intersection by logical disjunction and conjunction, respectively. This then leads to the following formula that describes the same geometric problem as (3.3):

$$\begin{aligned}\varphi_0 = ((&x \geq 0 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 4) \vee\\ (&x \geq 0 \wedge x \leq 4 \wedge y \geq 1 \wedge y \leq 3)) \wedge\\ ((&x \geq 1 \wedge x \leq 2 \wedge y \geq 0 \wedge y \leq 3) \vee\\ (&x \geq 1 \wedge x \leq 3 \wedge y \geq 2 \wedge y \leq 3)).\end{aligned} \tag{3.4}$$

Notice that the structure of (3.4) is a conjunction of disjunctions of conjunctions.

If this formula is satisfiable, SMT solving finds a *model* whose free constants (x and y in this example) describe a point that is included in the resulting intersection (3.3). For an unsatisfiable formula, SMT solving returns `unsat`. Note that in the case of satisfiability, the found model describes a point that lies in at least one rectangle of each union.

Let us run through our example and formula (3.4) with the help of an imaginary SMT solver. Assume SMT solving first yields point $E = (2, 3)$. Because E is a satisfying point of a conjunction, it must satisfy all terms of the conjunction, which are themselves disjunctions. For each disjunction, we look for one disjunction term that is satisfied by E. Notice that there is always at least one such term. Since E is included in all four rectangles, we select the terms that describe $AKMB$ and $DHJF$. Next, we build the conjunction of these two disjunction terms. This yields a conjunction of atoms, describing the rectangle *DHIE*:

$$\gamma_1 = x \geq 1 \land x \leq 2 \land y \geq 2 \land y \leq 3. \tag{3.5}$$

We have now found the description of one rectangle. In order to repeat this process, we want to make sure to get a new point that does not satisfy this already known part of the result. This is accomplished by adding one more conjunction that excludes the newly found conjunction γ_1 to the formula φ_0. Thus, we build $\varphi_1 = \varphi_0 \land \neg\gamma_1$.

This allows us to repeat the process on the new formula φ_1. Assume SMT solving on φ_1 yields point $F = (3, 3)$. F lies in rectangles *CKNG* and *DHJF*. The conjunction of the respective disjunction terms describes the rectangle *DHJF* expressed by the formula

$$\gamma_2 = x \geq 1 \land x \leq 3 \land y \geq 2 \land y \leq 3. \tag{3.6}$$

We add γ_2 to the result. As we can see now, through this process we possibly find rectangles that include some already found rectangle: $DHIE \subsetneq DHJF$. Since we want to avoid superfluous disjunctions, we remove *DHIE*, respectively γ_1, from the result. Note that this inclusion test has to be performed with each newly found rectangle against all already found ones.

Again, we exclude the newly found rectangle via $\varphi_2 = \varphi_1 \land \neg\gamma_2$ and use SMT solving next on φ_2. Assume that SMT solving returns point $L = (1, 1)$ this time. L is contained in *AKMB*, *CKNG*, and *DOPE*. Intersecting *AKMB* and *DOPE* yields *DLME* with the formula

$$\gamma_3 = x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 3, \tag{3.7}$$

and we add γ_3 to the result. Inclusion checking shows that *DLME* does not contain *DHJF*. We exclude γ_3 from future searches by building $\varphi_3 = \varphi_2 \land \neg\gamma_3$. Performing SMT solving on φ_3 returns `unsat`, so the computation is complete. The result is

$$\begin{aligned} \varphi' &= \gamma_2 \lor \gamma_3 \\ &= (x \geq 1 \land x \leq 3 \land y \geq 2 \land y \leq 3) \lor \\ &\quad (x \geq 1 \land x \leq 2 \land y \geq 1 \land y \leq 3). \end{aligned} \tag{3.8}$$

Converting this back to rectangles, this describes

$$DHJF \cup DLME.$$

This result is equivalent to 3.3, since *DHIE* was superfluous there.

In this example, the SMT solver always yielded points on the corner of a rectangle. This owes to the fact that in this case SMT solvers are used with the logic `QF_LRA` and use the Simplex algorithm [28], which proceeds from corner to corner. Nevertheless, any point that satisfies the formula would have done for us.

3.3 The Calculus

Now we present our method in the form of a calculus. Our calculus uses Satisfiability Modulo Theories (SMT). The calculus' functioning relies on the assumption that SMT is complete in the sense that it returns either `sat` or `unsat`. Recall that we are solving linear real problems, where `QF_LRA` is the adequate SMT logic. Here, SMT solvers typically use the Simplex method [28] and meet our requirements.

Let P be a finite set of variable-free constraints, which may however contain additional *free constants* x_1, …, x_n. Recall that technically, `QF_LRA` establishes a partial model where numbers, arithmetic symbols, and relations have an interpretation, but free constants have not. If $p \in P$, then p and $\neg p$ are called *literals.* Let $r \in \mathbb{N}$, and note that $\mathbb{N}$ includes 0. If ℓ_1, …, ℓ_r are literals, then $\ell_1 \wedge \cdots \wedge \ell_r$ is called a *conjunction* of literals. If $r = 0$, then the empty conjunction is true. If γ_1, …, γ_r are conjunctions of literals, then $\gamma_1 \vee \cdots \vee \gamma_r$ is called a *disjunctive normal form* (DNF). If $r = 0$, then the empty DNF is false. Let D be a DNF and P the set of all constraints of D. A DNF of D where each conjunction contains each $p \in P$ exactly once, either with or without negation, is called a *canonical disjunctive normal form* (CDNF). We call a conjunction of DNFs a *CDC-form.* It owes its name to its structure as a conjunction of disjunctions of conjunctions.

Let φ be a formula. A *model* of φ is a set of interpretations for all free constants x_1, …, x_n such that φ holds. We then write $M \models \varphi$ and say φ is *satisfiable*, otherwise we say φ is *unsatisfiable.* Given two formulas φ and φ' such that φ' holds for all models of φ, we write $\varphi \Vdash \varphi'$ and say φ *entails* φ', otherwise we write $\varphi \nVdash \varphi'$. If both $\varphi \Vdash \varphi'$ and $\varphi' \Vdash \varphi$, we call φ and φ' *logically equivalent* and write $\varphi \equiv \varphi'$.

A *state* is a 4-tuple $(\varphi, \delta, \psi, \zeta)$, where φ is a formula, δ is a DNF, ψ is a formula or $\bot$, and ζ is a DNF or $\bot$. A *rule* transforms states. Let S_1 be a state and C a condition. We say that rule $L \Longrightarrow R$ **if** C *applies* if S_1 syntactically matches L and semantically satisfies C. The rule then uniquely specifies a new state S_2 matching R. We then say that a *transition* of S_1 to S_2 exists and write $S_1 \vdash S_2$. Let $r \in \mathbb{N}$. A sequence of transitions of the form $S_0 \vdash S_1$, $S_1 \vdash S_2$, …, $S_{r-1} \vdash S_r$ is called a *derivation* and is written $S_0 \vdash^* S_r$. Note that $S_0 \vdash^* S_0$. In other words, $\vdash^*$ is the reflexive-transitive closure of $\vdash$.

Let $\mathcal{I}$ be an index set, $i \in \mathcal{I}$, and let $\mathcal{J}_i$ be an index set. We rewrite (3.1) as

$$\varphi_0 = \bigwedge_{i \in \mathcal{I}} \bigvee_{j \in \mathcal{J}_i} \gamma_{i,j}, \tag{3.9}$$

where $\gamma_{i,j}$ are conjunctions of constraints. We may limit ourselves to positive literals here, because negation can be coded into the constraints if needed.

If the input is φ_0, then the initial state of our calculus is

$$S_0 := (\varphi_0, \mathsf{false}, \bot, \bot). \tag{3.10}$$

An end state $(\varphi, \delta_{\text{end}}, \psi, \zeta)$ is reached if no rules are applicable. In this case, δ_{end} is the output.

Consider the CDC-form

$$\varphi = \bigwedge_{i \in \mathcal{I}} \bigvee_{j \in \mathcal{J}_i} \gamma_{i,j}. \tag{3.11}$$

Let M be a model of φ and let $i \in \mathcal{I}$. We define the function

$$\mathcal{S}(\varphi, M, i) := \{\, j \in \mathcal{J}_i \mid M \models \gamma_{i,j} \,\}. \tag{3.12}$$

Filter :

$$(\varphi,\, \delta,\, \bot,\, \bot) \implies (\varphi,\, \delta,\, \psi_0,\, \textsf{false})$$
where $\varphi = \varphi_0 \wedge \varphi_*,\ \ \psi_0 = \bigwedge_{i\in\mathcal{I}} \bigvee_{j\in\mathcal{S}(\varphi_0,M,i)} \gamma_{i,j}$
if φ is $\textsf{sat}$ with $M \models \varphi$

Add Clause :

$$(\varphi,\, \delta,\, \psi,\, \zeta) \implies (\varphi,\, \delta,\, \psi \wedge \neg\gamma',\, \zeta \vee \gamma')$$
where $\psi = \psi_0 \wedge \psi^*,\ \ \psi_0 = \bigwedge_{i\in\mathcal{I}} \bigvee_{j\in\mathcal{S}(\varphi_0,M,i)} \gamma_{i,j},\ \ \gamma' = \bigwedge_{i\in\mathcal{I}} \gamma_{i,\mathrm{S}(\psi_0,N,i)}$
if $\psi \neq \bot$, $\zeta \neq \bot$, and ψ is $\textsf{sat}$ with $N \models \psi$

Entailment :

$$(\varphi,\, \delta,\, \psi,\, \zeta) \implies (\varphi,\, \delta,\, \psi,\, \zeta')$$
where $\zeta = \gamma_1 \vee \cdots \vee \gamma_s \vee \gamma',\ \ \zeta' = \gamma_1 \vee \cdots \vee \gamma_{k-1} \vee \gamma_{k+1} \vee \cdots \vee \gamma_s \vee \gamma'$
if $\psi \neq \bot$, $\zeta \neq \bot$, and $\gamma_k \longrightarrow \gamma'$ is $\textsf{sat}$

Combine :

$$(\varphi,\, \delta,\, \psi,\, \zeta) \implies (\varphi \wedge \neg\zeta,\, \delta \vee \zeta,\, \bot,\, \bot)$$
if $\psi \neq \bot$, $\zeta \neq \bot$, and ψ is $\textsf{unsat}$

Figure 3.2: The calculus

It is easy to see that $\mathcal{S}(\varphi, M, i)$ is not empty. Furthermore, we define

$$\mathrm{S}(\varphi, M, i) := \text{select nondeterministically one } j \in \mathcal{S}(\varphi, M, i). \tag{3.13}$$

Figure 3.2 contains our calculus.

3.4 Termination and Soundness

Lemma 3.1 (Invariants of Add Clause and Entailment). *Consider a derivation*

$$S_0 \vdash^* R \vdash R' \vdash^* T.$$

As usual, $S_0 = (\varphi_0, \cdot, \cdot, \cdot)$ denotes the initial state, where φ_0 is the input formula. Furthermore, R, R', and T are states, where $R \vdash R'$ is an application of Filter *and $R' \vdash^* T$ consists only of applications of* Add Clause *or* Entailment. *Let M be a model of φ_0. $R' = (\cdot, \cdot, \psi_0, \cdot)$, where $\psi_0 = \bigwedge_{i\in\mathcal{I}} \bigvee_{j\in\mathcal{S}(\varphi_0,M,i)} \gamma_{i,j}$. $T = (\cdot, \cdot, \psi, \zeta)$. Let N be a model of ψ_0. Then*

(i) $\psi_0 \Vdash \varphi_0$,

(ii) $\mathcal{S}(\psi_0, N, i) \subseteq \mathcal{S}(\varphi_0, M, i)$,

(iii) $\zeta \Vdash \psi_0$,

(iv) $\psi \vee \zeta \equiv \psi_0$.

Proof. (i) Let K be a model of ψ_0, which equals $\bigwedge_{i\in\mathcal{I}} \bigvee_{j\in\mathcal{S}(\varphi_0,M,i)} \gamma_{i,j}$. According to (3.12), $\mathcal{S}(\varphi_0, M, i) \subseteq \mathcal{J}_i$ for all $i \in \mathcal{I}$. It follows that $\bigwedge_{i\in\mathcal{I}} \bigvee_{j\in\mathcal{J}_i} \gamma_{i,j}$. Hence, K is also a model of $\bigwedge_{i\in\mathcal{I}} \bigvee_{j\in\mathcal{J}_i} \gamma_{i,j}$, which equals φ_0 according to (3.9).

(ii) Recall that $\psi_0 = \bigwedge_{i\in\mathcal{I}} \bigvee_{j\in\mathcal{S}(\varphi_0,M,i)} \gamma_{i,j}$. Let $i \in \mathcal{I}$, and let $\ell \in \mathcal{S}(\psi_0, N, i)$. That is, according to (3.12), $\ell \in \{\, j \in \mathcal{S}(\varphi_0, M, i) \mid N \models \gamma_{i,j} \,\}$. Hence, $\ell \in \mathcal{S}(\varphi_0, M, i)$.

(iii) We perform induction on the number $n \in \mathbb{N}$ of derivation steps. Consider a derivation $S_0 \vdash^* R \vdash R' \vdash^n T_n$, where $T_n = (\cdot, \cdot, \psi_n, \zeta_n)$. We show $\zeta_n \Vdash \psi_0$.

Recall that $R \vdash R'$ is the last application of Filter. For $n = 0$, T_0 equals R'. According to Filter, $\zeta_0 = \mathsf{false}$. Hence, ψ_0 follows trivially.

Assume that $\zeta_n \Vdash \psi_0$ for $n \in \mathbb{N}$. Consider $R \vdash R' \vdash^{n+1} T_{n+1}$, say $R \vdash R' \vdash^n T_n \vdash T_{n+1}$, where $T_{n+1} = (\cdot, \cdot, \psi_{n+1}, \zeta_{n+1})$. We show the induction step $\zeta_{n+1} \Vdash \psi_0$ by case distinction on the rule, Add Clause or Entailment, applied in $T_n \vdash T_{n+1}$.

Consider the case that $T_n \vdash T_{n+1}$ is an application of Add Clause. According to that rule, $\zeta_{n+1} = \zeta_n \vee \gamma'$. Let K be a model of ζ_{n+1}. Since ζ_{n+1} is variable-free, it follows that K is a model of ζ_n, or of γ', or of both. If K is a model of ζ_n, then it follows by the induction hypothesis that K is a model of ψ_0. If K is not a model of ζ_n, then K is a model of γ'. Examination of Add Clause shows that $\gamma' = \bigwedge_{i\in\mathcal{I}} \gamma_{i,\mathrm{S}(\psi_0,N,i)}$. According to (3.13) and (ii), it follows that $\mathrm{S}(\psi_0, N, i) \in \mathcal{S}(\psi_0, N, i) \subseteq \mathcal{S}(\varphi_0, M, i)$ for all $i \in \mathcal{I}$. Thus, K is also a model of $\bigwedge_{i\in\mathcal{I}} \bigvee_{j\in\mathcal{S}(\varphi_0,M,i)} \gamma_{i,j}$, which equals ψ_0. Hence, $\zeta_{n+1} \Vdash \psi_0$.

Finally, consider the case that $T_n \vdash T_{n+1}$ is an application of Entailment. By the induction hypothesis, $\zeta_n \Vdash \psi_0$. According to Entailment, ζ_n has the form $\gamma_1 \vee \dots \vee \gamma_s \vee \gamma'$. Furthermore, there exists $k \in \{1, \dots, s\}$ such that $\gamma_k \longrightarrow \gamma'$ is sat. Thus, $\zeta_n \equiv \gamma_1 \vee \dots \vee \gamma_{k-1} \vee \gamma_{k+1} \vee \dots \vee \gamma_s \vee \gamma'$, which equals ζ_{n+1}. Hence, $\zeta_{n+1} \Vdash \psi_0$.

(iv) We perform induction on the number $n \in \mathbb{N}$ of derivation steps. Consider a derivation $S_0 \vdash^* R \vdash R' \vdash^n T_n$, where $T_n = (\cdot, \cdot, \psi_n^*, \zeta_n)$. We show $\psi_n^* \vee \zeta_n \equiv \psi_0$.

Recall that $R \vdash R'$ is the last application of Filter. For $n = 0$, T_0 equals R'. According to Filter, $\psi_0^* = \psi_0$ and $\zeta_0 = \mathsf{false}$. Hence, $\psi_0^* \vee \zeta_0 = \psi_0 \vee \mathsf{false} \equiv \psi_0$.

Assume that $\psi_n^* \vee \zeta_n \equiv \psi_0$ for $n \in \mathbb{N}$. Consider $R \vdash R' \vdash^{n+1} T_{n+1}$, say $R \vdash R' \vdash^n T_n \vdash T_{n+1}$, where $T_{n+1} = (\cdot, \cdot, \psi_{n+1}, \zeta_{n+1})$. We show the induction step $\psi_{n+1}^* \vee \zeta_{n+1} \equiv \psi_0$ by case distinction on the rule applied in $T_n \vdash T_{n+1}$.

Consider the case that $T_n \vdash T_{n+1}$ is an application of Add Clause. According to that rule, $\psi_{n+1}^* = \psi_n^* \wedge \neg\gamma'$ and $\zeta_{n+1} = \zeta_n \vee \gamma'$. Thus,

$$\begin{aligned}
\psi_{n+1}^* \vee \zeta_{n+1} &= (\psi_n^* \wedge \neg\gamma') \vee (\zeta_n \vee \gamma') \\
&\equiv (\psi_n^* \vee \zeta_n \vee \gamma') \wedge (\neg\gamma' \vee \zeta_n \vee \gamma') \\
&\equiv \psi_n^* \vee \zeta_n \vee \gamma'.
\end{aligned}$$

By the induction hypothesis, $\psi_n^* \vee \zeta_n \equiv \psi_0$. Therefore and by idempotency, $\psi_n^* \vee \zeta_n \vee \gamma' \equiv \psi_0 \vee \zeta_n \vee \gamma' \equiv \psi_0 \vee \zeta_{n+1}$. According to (iii), $\zeta_{n+1} \Vdash \psi_0$. Hence, $\psi_0 \vee \zeta_{n+1} \equiv \psi_0$.

Finally, consider the case that $T_n \vdash T_{n+1}$ is an application of Entailment. According to that rule, $\psi_{n+1}^* = \psi_n^*$. Also, ζ_n has the form $\gamma_1 \vee \cdots \vee \gamma_s \vee \gamma'$. Furthermore, there exists a $k \in \{1, \ldots, s\}$ such that $\gamma_k \longrightarrow \gamma'$ is sat. Thus, $\zeta_n \equiv \gamma_1 \vee \cdots \vee \gamma_{k-1} \vee \gamma_{k+1} \vee \cdots \vee \gamma_s \vee \gamma'$, which equals ζ_{n+1}. It follows that $\psi_{n+1}^* \vee \zeta_{n+1} \equiv \psi_n^* \vee \zeta_n$. Hence, by the induction hypothesis, $\psi_{n+1}^* \vee \zeta_{n+1} \equiv \psi_0$. □

Proposition 3.1 (Invariant of the calculus)**.** *Consider a derivation $S_0 \vdash^* T$. As usual, $S_0 = (\varphi_0, \mathsf{false}, \cdot, \cdot)$ is the initial state, where φ_0 is the input formula. $T = (\varphi, \delta, \cdot, \cdot)$ is a state. Then*

$$\varphi \vee \delta \equiv \varphi_0.$$

Proof. We perform induction on the number $n \in \mathbb{N}$ of derivation steps. Consider a derivation $S_0 \vdash^n T_n$, where $T_n = (\varphi_n^*, \delta_n, \psi_n^*, \zeta_n)$. We show $\varphi_n^* \vee \delta_n \equiv \varphi_0$.

For $n = 0$, T_0 equals the initial state $S_0 = (\varphi_0, \mathsf{false}, \cdot, \cdot)$. Thus, $\varphi_0^* = \varphi_0$ and $\delta_0 = \mathsf{false}$. Hence, $\varphi_0^* \vee \delta_0 = \varphi_0 \vee \mathsf{false} \equiv \varphi_0$.

Assume that $\varphi_n^* \vee \delta_n \equiv \varphi_0$ for $n \in \mathbb{N}$. Consider the derivation $S_0 \vdash^{n+1} T_{n+1}$, say $S_0 \vdash^n T_n \vdash T_{n+1}$. We show the induction step $\varphi_{n+1}^* \vee \delta_{n+1} \equiv \varphi_0$ by case distinction on the rule applied in $T_n \vdash T_{n+1}$.

Consider the case that $T_n \vdash T_{n+1}$ is an application of Filter, Add Clause, or Entailment. Inspection of these rules shows that $\varphi_{n+1}^* = \varphi_n^*$ and $\delta_{n+1} = \delta_n$. Hence, the induction step follows trivially.

Finally, consider the case that $T_n \vdash T_{n+1}$ is an application of Combine. It is easy to see that Filter has been applied before at least once. Rewrite the derivation as $S_0 \vdash^* R \vdash R' \vdash^* T_n \vdash T_{n+1}$, where R and $R' = (\cdot, \cdot, \psi_0, \cdot)$ are states such that $R \vdash R'$ is an application of Filter and $R' \vdash^* T_n$ contains no application of Filter. According to Combine, we have

$$\begin{aligned}
\varphi_{n+1}^* \vee \delta_{n+1} &= (\varphi_n^* \wedge \neg\zeta_n) \vee (\delta_n \vee \zeta_n) \\
&\equiv (\varphi_n^* \vee \delta_n \vee \zeta_n) \wedge (\neg\zeta_n \vee \delta_n \vee \zeta_n) \\
&\equiv \varphi_n^* \vee \delta_n \vee \zeta_n.
\end{aligned}$$

By the induction hypothesis, $\varphi_n^* \vee \delta_n \vee \zeta_n \equiv \varphi_0 \vee \zeta_n$. It is easy to see that $R' \vdash^* T_n$ consists only of applications of Add Clause or Entailment. Thus, Lemma 3.1 applies. By Lemma 3.1 (iv), $\psi_n^* \vee \zeta_n \equiv \psi_0$. By the condition of Combine, ψ_n^* is unsat. Therefore, $\psi_n^* \equiv \mathsf{false}$. It follows that $\zeta_n \equiv \psi_0$. Thus, $\varphi_0 \vee \zeta_n \equiv \varphi_0 \vee \psi_0$. Hence, by Lemma 3.1 (i) we obtain $\varphi_0 \vee \psi_0 \equiv \varphi_0$. □

If desired, one can ensure the minimality of the resulting DNF in the sense that no conjunction can be removed from the DNF without changing its truth value.

Let $V = (\cdot, \delta, \cdot, \cdot)$ be a state such that $\delta = \bigvee_{k=1}^{p} \gamma_k$. Consider a finite derivation $S_0 \vdash^* V$. Let ℓ, $m \in \mathbb{N}$ such that $\ell \neq m$. Obviously, minimality can be ensured by evaluating all $\gamma_\ell \Vdash \gamma_m$ and removing γ_ℓ if entailment holds. In the worst case, this requires $\mathcal{O}(p^2)$ entailment tests, where p is the maximum number of conjunctions at any stage of the computation.

If Entailment is applied, our calculus checks only conjunctions that result from the same Filter rule against one another, resulting in $\mathcal{O}(q^2)$ entailment tests, where q is the maximum number of conjunctions after Add Clause. Even though complexity is still $\mathcal{O}(p^2)$, this can drastically reduce the constant inside the $\mathcal{O}$.

Proposition 3.2 (Minimality)**.** *Consider a derivation $S_0 \vdash^* V$, where Entailment is applied whenever applicable. As usual, S_0 denotes the initial state. Furthermore, $V = (\cdot, \delta, \cdot, \cdot)$ is a state. δ is a DNF and has the form $\bigvee_k \gamma_k$. Let ℓ, $m \in \mathbb{N}$ such that $\ell \neq m$. Then*

$$\gamma_\ell \nVdash \gamma_m.$$

Proof. Let $k \in \mathbb{N}$. It is easy to see that all γ_k are added to δ in Combine. Furthermore, it is easy to see that each application of Combine is preceded by a matching application of Filter. Moreover, immediately after Filter the only applicable rule is Combine. Let $h \in \mathbb{N}$. Rewrite the derivation as

$$S_0 \vdash R_1 \vdash R_1' \vdash^* U_1 \vdash R_2 \vdash R_2' \vdash^* U_2 \vdash \cdots \vdash R_h \vdash^* V.$$

Let r, $s \in \mathbb{N} \setminus \{0\}$. $R_r = (\varphi_r^*, \cdot, \cdot, \cdot)$, R_r', and $U_r = (\cdot, \cdot, \cdot, \zeta_r)$ are states, where $R_r \vdash R_r'$ is the r-th application of Filter, $U_r \vdash R_{r+1}$ is the r-th application of Combine, and $R_h \vdash^* V$ contains no application of Filter. We may assume that r has been chosen such that γ_ℓ is contained in ζ_r and s such that γ_m is contained in ζ_s. That is, for r_1, r_2, s_1, $s_2 \in \mathbb{N}$, ζ_r has the form $\gamma_{r_1} \vee \cdots \vee \gamma_\ell \vee \cdots \vee \gamma_{r_2}$, and ζ_s has the form $\gamma_{s_1} \vee \cdots \vee \gamma_m \vee \cdots \vee \gamma_{s_2}$. Let M_r be a model of φ_r^*.

Firstly, consider the case that $r = s$. Consider the subcase that $\ell < m$. As stated in the Proposition, Entailment has been applied whenever applicable. Hence, $\gamma_\ell \nVdash \gamma_m$.

Consider the alternate subcase that $\ell > m$. Consider the derivation $R_r' \vdash^* T_m \vdash T_m' \vdash^* T_\ell \vdash T_\ell' \vdash^* U_r$, where $T_m = (\cdot, \cdot, \chi_m, \cdot)$, $T_m' = (\cdot, \cdot, \chi_m', \cdot)$, $T_\ell = (\cdot, \cdot, \chi_\ell, \cdot)$, and T_ℓ' are states such that $T_m \vdash T_m'$ and $T_\ell \vdash T_\ell'$ are applications of Add Clause that yield γ_m and γ_ℓ, respectively. Let N_ℓ be a model of χ_ℓ. Inspection of Add Clause shows that $N_\ell \models \gamma_\ell$. According to Add Clause, $\chi_m' = \chi_m \wedge \neg\gamma_m$. Since $\ell > m$, $\neg\gamma_m$ is contained in the conjunction χ_ℓ. Thus, $N_\ell \not\models \gamma_m$. Hence, $\gamma_\ell \nVdash \gamma_m$.

Secondly, consider the case that $r < s$. It is not hard to see that $M_r \models \gamma_\ell$ and thus $M_r \models \zeta_r$. According to Combine, we have $\varphi_{r+1}^* = \varphi_r^* \wedge \neg\zeta_r$. Since $s > r$, it follows that $M_r \not\models \varphi_s^*$. Thus, $M_r \not\models \zeta_s$ and consequently, $M_r \not\models \gamma_m$. Hence, $\gamma_\ell \nVdash \gamma_m$.

Thirdly, consider the case that $r > s$. We prove this by contradiction. Assume $\gamma_\ell \Vdash \gamma_m$. It is easy to see that $M_r \models \gamma_\ell$. It follows from the assumption that also $M_r \models \gamma_m$, and thus $M_r \models \zeta_s$ and $M_r \models \varphi_s^*$. Since $\varphi_{s+1}^* = \varphi_s^* \wedge \neg\zeta_s$ and $r > s$, it follows that $M_r \not\models \zeta_s$, which is a contradiction. Hence, $\gamma_\ell \nVdash \gamma_m$. □

Theorem 3.1 (Termination)**.** *The calculus terminates. In other words, there are no infinite derivations $S_0 \vdash S_1 \vdash \dots$*

Proof. Let $T = (\varphi, \delta, \psi, \zeta)$ be a state, where φ is a formula. Let P be the set of all constraints of φ and let $\alpha = |P|$, the number of constraints. Recall that there is exactly one CDNF for φ modulo reordering of literals [48, p. 52–53] and that it consists of at most 2^α conjunctions.

Let "$\prec$" be lexicographically strictly less on $\mathbb{N}^3$. A weight $(a, b, c) \in \mathbb{N}^3$ is assigned to T as follows:

- a is the number of conjunctions of the CDNF of φ that are satisfiable;
- b is $2^\alpha + 1$ if $\psi = \bot$, otherwise the number of conjunctions of the CDNF of ψ that are satisfiable;
- c is 0 if $\zeta = \bot$, otherwise the word length of ζ.

Let $S = (\varphi, \delta, \psi, \zeta)$ and $S' = (\varphi', \delta', \psi', \zeta')$ be states. Consider a transition $S \vdash S'$. Let (a, b, c) and (a', b', c') be the weights for S and S', respectively.

Filter yields $a' = a$, $b' < b$, and $c' > c$. We have $a' = a$, since $\varphi' = \varphi$. It holds that $b' < b$, since $\psi' \neq \bot$ and thus $b' \leq 2^\alpha$, whereas $\psi = \bot$ and thus $b = 2^\alpha + 1$. $c' > c$, since $\zeta' = \mathsf{false}$ and thus $c' = 1$, whereas $\zeta = \bot$ and thus $c = 0$.

Add Clause yields $a' = a$, $b' < b$, and $c' > c$. Then $a' = a$, since $\varphi' = \varphi$. Furthermore, we have $b' < b$. ψ is satisfied according to the condition of the rule. According to (3.13), γ' is also satisfied. Thus, the CDNF of $\psi \wedge \neg\gamma'$ has at least one satisfiable conjunction less than the CDNF of ψ. Lastly, $c' > c$, since the word length of ζ' is greater than that of ζ.

Entailment yields $a' = a$, $b' = b$, and $c' < c$, since $\varphi' = \varphi$ and $\psi' = \psi$, and the word length of ζ' is less than that of ζ.

Combine yields $a' < a$, $b' > b$, $c' < c$. It is easy to see that ζ consists of at least one satisfiable conjunction. Thus and by Lemma 3.1 (i) & (iii), $a' < a$, since the CDNF of $\varphi \wedge \neg\zeta$ has at least one satisfiable conjunction less than the CDNF of φ. Furthermore, $b' > b$, since $\psi' = \bot$ and $\psi \neq \bot$, and thus $b' = 2^\alpha + 1$ and $b \leq 2^\alpha$. Lastly, $c' < c$, since $\zeta' = \bot$ and $\zeta \neq \bot$, and thus $c' = 0$ and $c > 0$.

Hence, $S \vdash S'$ implies $(a', b', c') \prec (a, b, c)$. Since "$\prec$" is well-founded, every derivation terminates. □

Theorem 3.2 (Soundness)**.** *The calculus is sound. In other words, let φ_0 be an input CDC-form and let δ_{end} be an output DNF. Consider a terminating derivation $S_0 \vdash^* T$, where $S_0 = (\varphi_0, \cdot, \cdot, \cdot)$ is the initial state and $T = (\cdot, \delta_{\text{end}}, \cdot, \cdot)$ is a terminating state. Then $\varphi_0 \equiv \delta_{\text{end}}$.*

Proof. Let $U = (\varphi, \delta, \cdot, \cdot)$ be a state. According to Proposition 3.1 it holds that $\varphi \vee \delta \equiv \varphi_0$. It is easy to see that this holds especially for a terminating state, where $\varphi \equiv \mathsf{false}$ and $\delta = \delta_{\text{end}}$. Hence, $\varphi_0 \equiv \delta_{\text{end}}$. □

3.5 The Calculus as an Algorithm

By courtesy to the reader, we present the calculus of the last section in the form of two algorithms. We chose the presentation as *two* algorithms, where one is the sole caller of the other, since this is not only advantageous for implementation, but also allows us to point out the similarities as well as the differences of the two procedures.

Algorithm 11 CDCFormToDNF

Input: A CDC-form $\varphi_0 = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} \gamma_{i,j}$, where $\gamma_{i,j}$ are conjunctions of quantifier-free first-order formulas over the reals.

Output: A DNF δ_{end} with $\delta_{\text{end}} \equiv \varphi_0$.

The DNF δ_{end} is minimal in the sense that no conjunction can be left out without violating $\delta_{\text{end}} \equiv \psi_0$.

```
1: $\delta :=$ false
2: $\varphi := \varphi_0$
3: while $M \models \varphi$ do
4:     $\psi_0 := \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} \{ \gamma_{i,j} \mid M \models \gamma_{i,j} \}$
5:     $\zeta :=$ CDCFormToDNFSub($\psi_0$)
6:     $\delta := \delta \vee \zeta$
7:     $\varphi := \varphi \wedge \neg\zeta$
8: end while
9: return $\delta$
```

3.5.1 Algorithm 11: CDCFormToDNF

The main algorithm to be called is Algorithm 11. It is called in Algorithm 4 to return the DNF Π.

Line 1 of Algorithm 11 initializes δ, the eventual DNF to be returned, to false, and l.2 sets φ to the input CDC-form φ_0. The while-loop in l.3–8 proceeds as long as there exists a model M that satisfies φ. This constitutes the condition of the Filter rule, where l.4 contains the body. Add Clause and Entailment are outsourced to a separate algorithm in l.5. The returned DNF ζ is disjunctively added to the DNF δ and its negation conjunctively added to φ to exclude further models that satisfy ζ in l.6–7. These two lines constitute the Combine rule. Finally, l.9 returns the DNF δ.

3.5.2 Algorithm 12: CDCFormToDNFSub

Algorithm 11 calls function CDCFormToDNFSub to perform the Add Clause and Entailment rules of the calculus. Whereas the calculus allows Entailment but does not enforce it, Algorithm 12 always employs Entailment.

Algorithm 12 accepts a CDC-form and returns a DNF, as does its calling Algorithm 11. In l.1–2 the DNF ζ and the formula ψ are initialized to false and ψ_0, respectively. The while-loop in l.3–21 is executed as long as ψ is satisfiable. Line 4 and the nested for-loops in l.5–12 construct γ' as in the Add Clause rule. Lines 13–18 contain the body of the Entailment rule and l.19–20 the body of the Combine rule. After ψ is unsat, the while-loop is left and in l.22 the DNF ζ is returned.

Notice that Algorithm 12 has the same semantics as Algorithm 11 in the sense that it takes the same type of arguments and returns the same type of return value with the same property. The noteworthy difference between the two is that Algorithm 12 checks the Entailment rule. In the process, it checks a newly found DNF conjunction (l.4–12) for entailment against all conjunctions found before (l.13–18). In contrast to this, Algorithm 11 does not need to check for entailment, since this cannot happen by design. See Proposition 3.2 for a proof.

Algorithm 12 CDCFormToDNFSub

Input: A CDC-form $\psi_0 = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} \gamma_{i,j}$, where $\gamma_{i,j}$ are conjunctions of quantifier-free first-order formulas over the reals.

Output: A DNF ζ with $\zeta \equiv \psi_0$.

```
1: $\zeta :=$ false
2: $\psi := \psi_0$
3: while $N \models \psi$ do
4:   $\gamma' :=$ true
5:   for $i := 1$ to $n$ do
6:     for $j := 1$ to $m_i$ do
7:       if $N \models \gamma_{i,j}$ then
8:         $\gamma' := \gamma' \wedge \gamma_{i,j}$
9:         break
10:      end if
11:    end for
12:  end for
13:  $\zeta' :=$ false
14:  for all conjunctions $\gamma$ in $\zeta$ do
15:    if $\mathbb{R} \not\models \gamma \longrightarrow \gamma'$ then
16:      $\zeta' := \zeta' \vee \gamma$
17:    end if
18:  end for
19:  $\zeta := \zeta' \vee \gamma'$
20:  $\psi := \psi \wedge \neg\gamma'$
21: end while
22: return $\zeta$
```

3.6 Remarks

1. The calculus is described using the SMT logic `QF_LRA`. It can be straightforwardly adapted to other complete theories T. “Complete” is used in the sense that for every sentence φ either $T \models \varphi$ or $T \models \neg\varphi$. Recall that the SMT solver implementation for the corresponding theory has to be complete as well, in the sense that for every formula it returns either `sat` or `unsat`, but not `unknown`.

   Beyond SMT solving, the calculus is compatible with any decision procedure that produces a witness in the positive case. Examples are the theory of real closed fields, corresponding to the SMT logic `QF_NRA`; Presburger arithmetic [79], corresponding to the SMT logic `QF_LIA`; or the linear theory of p-adic numbers for a fixed prime p. For real closed fields, one can use cylindrical algebraic decomposition [3] as a decision procedure with the test point of a satisfying cell as the witness. For the other theories mentioned above one can use extended quantifier elimination by virtual substitution [50] for the weak linear theory of the integers [53] and for the linear theory of discretely valued fields [101], respectively.

2. Recall that rule Add Clause adds conjunctions $\gamma' = \bigwedge_{i\in\mathcal{I}} \gamma_{i,\mathrm{S}(\psi_0,N,i)}$ to the DNF, as per the calculus. So far, we make no effort to minimize these conjunctions in the sense as to remove redundant terms γ_k from γ'. Take as an example the rectangles from Section 3.2. SMT solving first found point E and we chose

rectangles $AKMB$ and $DHJF$. This leads to the following conjunction:

$$x \geq 0 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 4 \wedge x \geq 1 \wedge x \leq 3 \wedge y \geq 2 \wedge y \leq 3. \tag{3.14}$$

It is easy to see that half of the constraints are superfluous. If we somehow drop the superfluous ones, we get the minimal result (3.5). Such a minimization could be formulated quite straightforwardly with a rule similar to Entailment, which we only sketch here: Given $\gamma' = \gamma_1 \wedge \cdots \wedge \gamma_s \wedge \gamma''$ one can replace γ' with $\gamma_1 \wedge \cdots \wedge \gamma_{k-1} \wedge \gamma_{k+1} \wedge \cdots \wedge \gamma_s \wedge \gamma''$ provided that $\gamma'' \longrightarrow \gamma_k$ is sat. We call this the Small Entailment rule. It is noteworthy that δ never contains constraints that are logically equivalent to false, such as $x^2 < 0$, even without Small Entailment. In contrast, constraints that are logically equivalent to true, such as $x^2 \geq 0$, may be contained in δ, but would be removed by a Small Entailment rule.

3. Notice that Algorithm 11 is an *anytime algorithm* [109]. That is, the longer it runs, the more its output approximates the real result. Even if the full computation of the result for some input φ_0 might not be feasible, at least parts of the result can be computed. If the while-loop in l.3–8 of Algorithm 11 is prematurely broken out of, any result δ satisfies $\delta \Vdash \varphi_0$, while only the full result also satisfies $\varphi_0 \Vdash \delta$ and thus $\delta \equiv \varphi_0$.

4. If Algorithm 11 is implemented with the help of an actual SMT solver, this solver can be used in *incremental mode*. This is possible whenever one wants to conjunctively add further conditions to a formula being solved after the solver has found a model. This is exactly what happens in l.7 of Algorithm 11 and l.20 of Algorithm 12. If these loops cycle many times, the reduction in run-time can be quite substantial.

5. Minimality can be ensured by applying rule Entailment whenever applicable. However, this can cause a quadratic run-time in the number of DNF terms. As of now, the calculus is broken up into two parts where one consists of Filter and Combine and the other of Add Clause and Entailment. Filter creates from φ the CDC-form ψ_0, which contains only those conjunctions that are modeled by some M, where $M \models \varphi$. A possible enhancement would be to repeat this filtering multiple times until the number of possible combinations, and therefore the maximum number of disjunction terms, is below a constant threshold. If it could be proven that this leads to only a logarithmic depth, time needed to ensure minimality could be reduced from $\mathcal{O}(p^2)$ to $\mathcal{O}(p \log p)$.

Chapter 4

ODEbase: A Repository of ODE Systems for Systems Biology

One advantage of symbolic computation is its potential to provide qualitative answers to biological questions. Qualitative methods analyze dynamical input systems as formal objects, in contrast to investigating only part of the state space, as is the case with numerical simulation. However, symbolic computation tools and libraries have a different set of requirements for their input data than their numerical counterparts. A common format used in mathematical modeling of biological processes is SBML. We point out that the use of SBML data in symbolic computation requires significant pre-processing, incorporating external biological and mathematical expertise. ODEbase provides high quality symbolic computation input data derived from established existing biomodels, covering in particular the BioModels database. We make use of ODEbase in Chapter 5, where we use it to provide data sets for our benchmarks. This chapter was previously published in [61].

4.1 Introduction

Recently, symbolic computation methods are playing an increasing role in systems biology and mathematical biology [12, and the references there]. Problems investigated using such methods include Hopf bifurcations, multi-stationarity, multi-scale model reduction, dynamical invariants, and structural properties of steady state varieties; for details see, e.g., [29, 36, 23, 13, 51, 35]. The biological systems investigated so far were focused on reaction networks in the sense of chemical reaction network theory [30]. Such networks are usually stored and exchanged in Systems Biology Markup Language (SBML), a free, open, and standardized XML-based format [41].

On the one hand, symbolic computation does not utilize the full information contained in SBML models. For instance, SBML was designed with a focus on network simulation and supports corresponding concepts like events and initial assignments, which are not natural from a formal symbolic computation point of view. On the other hand, symbolic computation operates on formal objects, which are not readily available in SBML. One prominent class of examples are ODEs describing differential network kinetics along with algebraic constraints, such as conservation laws. The genuine difference between dynamic simulation and static formal analysis requires sensitivity to details and rigor in the course of the construction of symbolic computation input from available SBML descriptions. It is noteworthy that existing SBML parsers generate input for numerical simulation, which is not suited for symbolic computation. MathSBML [99], SBFC [88], SBMLtoODEpy [91], and SBML2Modelica [63] fall into this category.

The rigorous construction of symbolic computation input poses some substantial problems. Solving, or even recognizing, such problems, requires joint competence and combined efforts not only from biology, but also from mathematics and computer science. We note some examples for such problems:

- SBML allows floating-point values for various entities. However, floating-point values exhibit representation errors and computations are prone to rounding errors. This is inadequate for symbolic computation, where exact computations are performed.

- SBML has liberal naming conventions for species and parameters that interfere with the typically strict rules of symbolic computation software, which are oriented towards mathematical notation. If different users of symbolic computation software rename those identifiers at their own discretion, it becomes difficult to compare their results.

- SBML gives modelers versatile opportunities of expression, such as local parameters, function definitions, rules, and initial assignments. For practical reasons, scientific software does not generally support the full SBML feature set. This leads to incompletely imported models or it prohibits their import entirely.

- Symbolic computation is concerned with mathematical properties like deficiency and linear conservation laws, which are available in SBML only implicitly through computation. Explicit availability is desirable, especially, since some of those computations can become surprisingly time-consuming.

With this in mind, the question is now how to make the abundantly available SBML data accessible for the symbolic computation community in a suitably prepared form.

A natural idea would be to integrate symbolic computation input into the SBML format. However, there are obstacles on both sides: On the symbolic computation side, established software is usually general-purpose, and systems biology is not yet firmly established in the community. Therefore, widespread support of SBML as an input format for symbolic computation software cannot be expected in the near future. On the systems biology side, the SBML standard would need to be extended. Standardization generally requires considerable efforts, and it seems unlikely that this will be pursued before the links between symbolic computation and systems biology have been further strengthened.

The interdisciplinary project SYMBIONT brings together researchers from mathematics, computer science, and systems biology [12]. Within SYMBIONT, we have started the online database ODEbase, which collects symbolic computation input for existing SBML models. All models have been carefully constructed taking into consideration the issues discussed above. At the time of writing, all our models originate from the BioModels database [77], the world's largest repository of curated mathematical models of biological processes and one of the most important data sources for modeling [65]. Out of the 1044 models from the curated branch of BioModels, we have currently compiled 662 into ODEbase.

As ODEbase has turned out to be extremely valuable throughout the SYMBIONT project, we now make it available to the community under `https://odebase.org` as a free and open database, beyond the lifespan of the project. ODEbase has already been used as data source in a number of papers [35, 83, 44]. If models require updates, revised versions will be made available, keeping all previous versions for reference. Data can be extracted in LaTeX, Maple, Reduce, and SageMath format. We are open to supporting further formats in the future.

ODEbase provides a canonical source of symbolic computation input related to existing models of biological processes. This has a number of advantages:

1. Interdisciplinary competence: The derivation of adequate ODEs for the kinetics of existing biomodels requires the incorporation of external biological and mathematical expertise. We have accomplished this task for a large set of available models and make the results available to the community.

2. Economic use of human resources: Symbolic computation input has been precomputed and is directly available.

3. Availability: ODEbase models used and cited in the literature can be conveniently reviewed on the basis of the original data and re-used in follow-up publications.

4. Canonical reference: ODEbase fixes choices for the inevitable renaming of species and parameters to common mathematical notation. This facilitates comparability of results.

5. Benchmarking: Beyond its primary purpose, ODEbase is perfectly suited to generate benchmark sets for novel algorithms and software in the field.

4.2 Details

4.2.1 The Content of ODEbase Data Sets

For each model in ODEbase, the following data set is computed from the original SBML input:

Stoichiometric and kinetic matrices. Stoichiometric and kinetic matrices are made explicit. In that course, floating-point values in the SBML input are converted to exact rational numbers.

ODEs for species concentrations. These are explicit first-order, nonlinear ODEs that are often, but not necessarily, autonomous. Species are named $x_1, \dots, x_n$, following common mathematical notation. The ODEs are created from the stoichiometric matrix and the relevant kinetic laws. Again, floating-point values are converted to exact rational numbers. We have taken care to preserve the structure of mathematical terms by using abstract syntax trees as an intermediate format. One visible effect of this are uncanceled rational functions and the presence of stoichiometric coefficients of 1 or -1. If species rate rules are present in the model, the corresponding ODEs are included as well.

Parameter values. Our naming of parameters also follows common mathematical notation, viz., k_1, …, k_m. Parameter values are converted from floating-point representation to exact rational numbers. If there are initial assignments or assignment rules in the SBML model, they are applied in the proper order to calculate the parameter values. To avoid any representation errors, all values are queried as text from the XML source.

Map between ODEbase names and original model names. A bijective mapping between the mathematical names for species and parameters and their respective SBML names is provided.

Constraints. All SBML species assignment rules are converted to formal constraints. Furthermore, linear conservation constraints are computed from the stoichiometric matrix using an algorithm by Schuster and Höfer [98], extended to handle multiple model compartments. All constraints introduced in this way can be combined with the ODEs mentioned above which yields the relevant ODE system for the model.

Deficiency. The deficiency of the reaction network is computed from its complexes. This is a measure of how independent the reaction vectors are, given the network's linkage class structure [30, Sect. 6.3].

Classification. Polynomials and, more generally, rational functions play a crucial role in symbolic computation. We classify whether ODE vector fields and constraints are covered by such expressions. In the polynomial case, we furthermore check if the SBML-specified kinetics differs from the regular mass action kinetics [30, Sect. 2.1.2] only by a constant factor. This is a conservative heuristic for identifying models with mass action kinetics.

4.2.2 Supported SBML Features

All models in ODEbase are a faithful conversion from the respective SBML model. SBML features recognized during the conservation process include the following:

- species with boundary condition,
- local parameters,
- parameter and species assignment rules,
- parameter and species initial assignments,
- species rate rules,

- function definitions.

SBML supports events, i.e., discrete model changes at certain points in time, and furthermore it supports parameter rate rules. Models that contain either of those are currently not included in ODEbase. Neither are models with irrational parameter values.

4.2.3 The ODEbase Web Service

The website at `https://odebase.org` is a free and open service, where the user can filter and search for models to display or download them. The website uses the responsive front-end framework Bootstrap [100] and is easy to view and navigate on both desktop and mobile computers.

The website constains a top navbar with the following options:

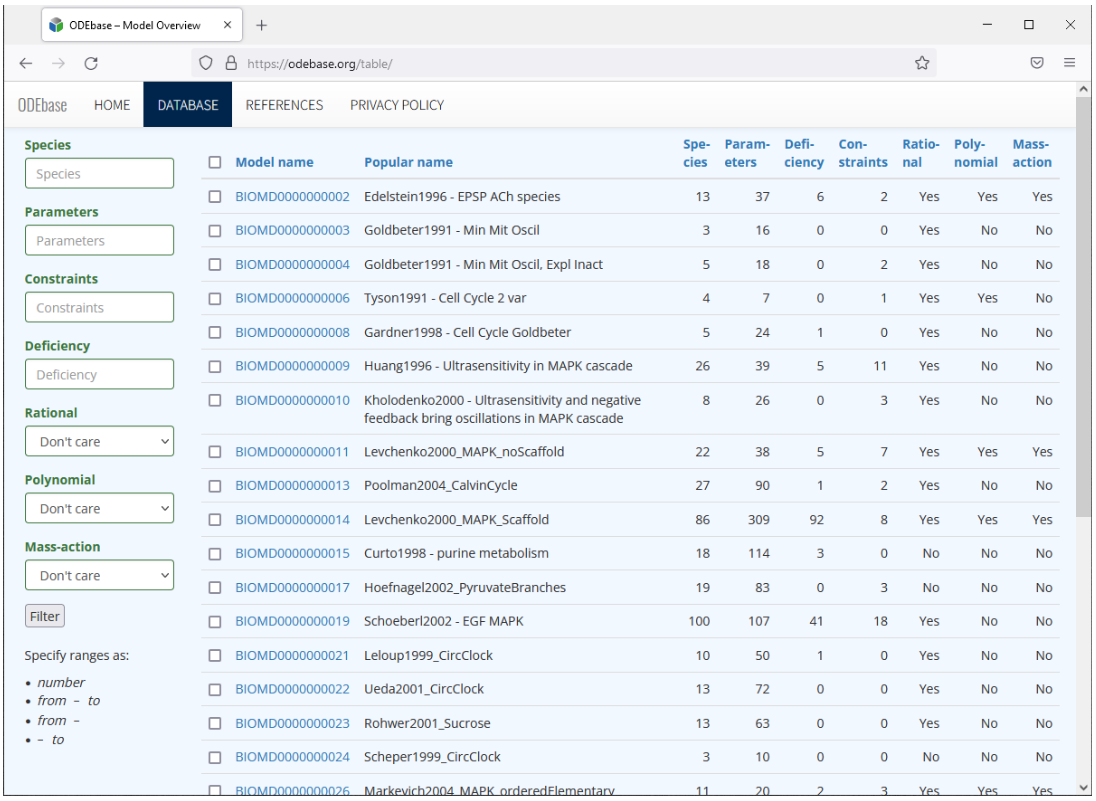
- "Home", the main page with information about the operators of the website;
- "Database", where models can be accessed;
- "References", a list of scientific publications that utilize ODEbase;
- "Privacy Policy", the data protection declaration, which is required by law.

The table of models is reached by clicking "database" in the navbar. Once there, on the left of the table is a sidebar, where the user can refine which models should be displayed. This can be done by entering explicit numerical values or ranges for several biomathematical properties of a model, namely, the number of species, the number of parameters, the number of constraints, and the deficiency. Furthermore, there are inputs that accept 3-valued logic with the values "Yes", "No", and "Don't care" to select for rational functions, polynomials, and systems with mass action kinetics. Figure 4.1 shows a screenshot of the table without any selection.

By clicking on the model identifier, the user gets a detailed view of the respective model. Here, the model's properties are displayed, which are the number of total, reversible, and one-way reactions, in addition to the properties available through the filter sidebar. Furthermore, the following mathematical objects can be downloaded: stoichiometric matrix, kinematic matrix, ODEs, constraints, and parameter values. Each of these data can be downloaded in either LaTeX, Maple, Reduce, or SageMath format.

A mapping between ODEbase names for species and parameters and SBML names can be downloaded as well. Additionally, a link to the original model in the BioModels database and possible links to known research about each model are displayed here. Figure 4.2 shows a sample details page.

Lastly, multiple models can be selected by clicking a checkbox in front of the model name. Thereafter, the user can click on "Download selected" and select which of the aforementioned mathematical objects in which formats are desired to download. When started, a .zip file is created for download so that the user gets everything bundled into one file. See Figure 4.3 for a screenshot.



Species
Species

Parameters
Parameters

Constraints
Constraints

Deficiency
Deficiency

Rational
Don't care

Polynomial
Don't care

Mass-action
Don't care

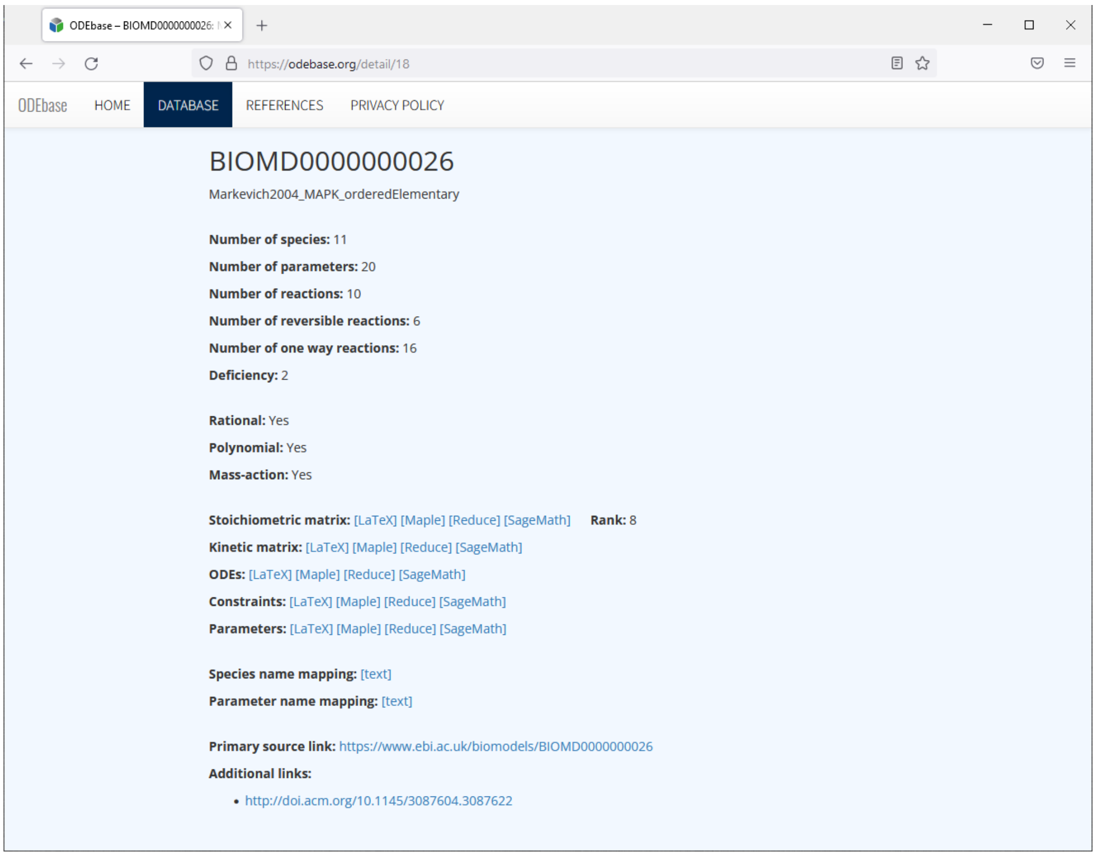
Filter

Specify ranges as:

- *number*
- *from* – *to*
- *from* –
- – *to*

Model name	Popular name	Spe-cies	Param-eters	Defi-ciency	Con-straints	Ratio-nal	Poly-nomial	Mass-action
BIOMD0000000002	Edelstein1996 - EPSP ACh species	13	37	6	2	Yes	Yes	Yes
BIOMD0000000003	Goldbeter1991 - Min Mit Oscil	3	16	0	0	Yes	No	No
BIOMD0000000004	Goldbeter1991 - Min Mit Oscil, Expl Inact	5	18	0	2	Yes	No	No
BIOMD0000000006	Tyson1991 - Cell Cycle 2 var	4	7	0	1	Yes	Yes	No
BIOMD0000000008	Gardner1998 - Cell Cycle Goldbeter	5	24	1	0	Yes	No	No
BIOMD0000000009	Huang1996 - Ultrasensitivity in MAPK cascade	26	39	5	11	Yes	No	No
BIOMD0000000010	Kholodenko2000 - Ultrasensitivity and negative feedback bring oscillations in MAPK cascade	8	26	0	3	Yes	No	No
BIOMD0000000011	Levchenko2000_MAPK_noScaffold	22	38	5	7	Yes	Yes	Yes
BIOMD0000000013	Poolman2004_CalvinCycle	27	90	1	2	Yes	No	No
BIOMD0000000014	Levchenko2000_MAPK_Scaffold	86	309	92	8	Yes	Yes	Yes
BIOMD0000000015	Curto1998 - purine metabolism	18	114	3	0	No	No	No
BIOMD0000000017	Hoefnagel2002_PyruvateBranches	19	83	0	3	No	No	No
BIOMD0000000019	Schoeberl2002 - EGF MAPK	100	107	41	18	Yes	No	No
BIOMD0000000021	Leloup1999_CircClock	10	50	1	0	Yes	No	No
BIOMD0000000022	Ueda2001_CircClock	13	72	0	0	Yes	No	No
BIOMD0000000023	Rohwer2001_Sucrose	13	63	0	0	Yes	No	No
BIOMD0000000024	Scheper1999_CircClock	3	10	0	0	No	No	No
BIOMD0000000026	Markevich2004_MAPK_orderedElementary	11	20	2	3	Yes	Yes	Yes

Figure 4.1: ODEbase table of models



BIOMD0000000026

Markevich2004_MAPK_orderedElementary

Number of species: 11

Number of parameters: 20

Number of reactions: 10

Number of reversible reactions: 6

Number of one way reactions: 16

Deficiency: 2

Rational: Yes

Polynomial: Yes

Mass-action: Yes

Stoichiometric matrix: [LaTeX] [Maple] [Reduce] [SageMath] **Rank:** 8

Kinetic matrix: [LaTeX] [Maple] [Reduce] [SageMath]

ODEs: [LaTeX] [Maple] [Reduce] [SageMath]

Constraints: [LaTeX] [Maple] [Reduce] [SageMath]

Parameters: [LaTeX] [Maple] [Reduce] [SageMath]

Species name mapping: [text]

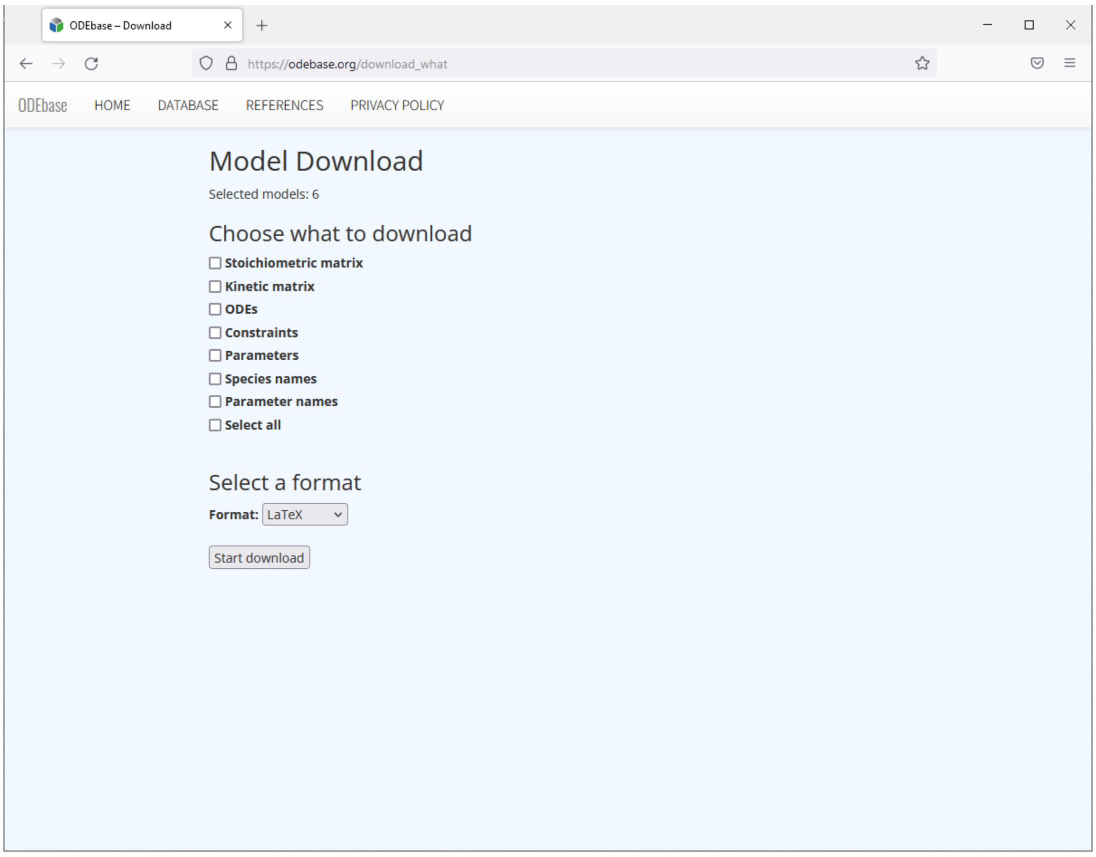
Parameter name mapping: [text]

Primary source link: https://www.ebi.ac.uk/biomodels/BIOMD0000000026

Additional links:

- http://doi.acm.org/10.1145/3087604.3087622

Figure 4.2: ODEbase detail page for `BIOMD0000000026`



Model Download

Selected models: 6

Choose what to download

- ☐ Stoichiometric matrix
- ☐ Kinetic matrix
- ☐ ODEs
- ☐ Constraints
- ☐ Parameters
- ☐ Species names
- ☐ Parameter names
- ☐ Select all

Select a format

Format: LaTeX

Start download

Figure 4.3: ODEbase download page for multiple models

Chapter 5

Benchmarks

In Chapter 2, we presented an algorithmic approach to partition ODE systems into multiple time scales. Part of this approach is Algorithm 4, which computes the tropical equilibration of a system of polynomials. The algorithm builds a CDC-form of quantifier-free first-order formulas over the reals. Recall that a CDC-form is a conjunction of disjunctions of conjunctions.

In principle, one could compute a DNF from such a CDC-form by repeated application of the distributive law. This creates a formula with the *theoretical maximum number of combinations* of conjunctions. In the following, we simply call this the *number of combinations*. Since run-time and memory consumption grow exponentially with the number of equations, it is infeasible for many systems to proceed in this way. To compute a DNF from a CDC-form in a more efficient way, we presented Algorithm 11 in Chapter 3. This algorithm is then called in l.23 of Algorithm 4.

ODEbase was introduced in Chapter 4. One of the purposes of ODEbase is to provide real-world examples of ODE systems for benchmarking. We use it now as a source for realistic input data for our benchmarks.

In Section 5.1, we investigate the run-times of SMTcut, a software prototype of Algorithm 11, and compare them with existing software. Section 5.2 contains a comparison of the run-times of SMTcut with a software prototype by Samal [92]. Lastly, Section 5.3 contains a short account of the run-times of the reduction process of Algorithms 1 and Algorithms 5–7.

5.1 Comparison of SMTcut and Redlog

We compare the speed of Algorithm 11 to established software, that is, Redlog [24], which is available in the computer algebra system Reduce [38]. No other system could be found that supports first-order formulas over the reals and exploits the structure of ordered fields. If such structure is supported, then, for example, $x < 0 \wedge x > 0 \equiv \mathsf{false}$ and $x \leq 0 \wedge x \geq 0 \equiv x = 0$. Without such support, a resulting DNF contains many conjunctions that are logically equivalent to false and thus the result is much too long. This can quickly reach a point where computation becomes infeasible.

There are some obstacles to the successful computation of a tropical equilibration. For one, polynomials must not be zero. This might however be the case in models with species that have a boundary condition, where ODEs have the form $\frac{\mathrm{d}x_i}{\mathrm{d}t} = 0$. Then there is the case where the stoichiometry of a species is zero and the corresponding ODE right hand side is zero as well. Also, polynomials may become zero due to zero parameters. Lastly, tropicalization requires at least one monomial for both, positive and negative signs. If only one of these types of monomials is present, it is called an *unbalanced monomial* and tropicalization can not be performed. If any of these obstacles occur, tropicalization is not possible.

For our benchmark, we first performed tropicalization with $\varepsilon_* = \frac{1}{5}$ and $p = 1$. This part is not part of the benchmark and was done with SMTcut. The output of this computation, which describes a CDC-form, was saved to be used later in the benchmarks of Redlog and SMTcut.

The machine used for benchmarks is running Ubuntu Linux 20.04.3 in 64-bit mode with kernel version 5.13.0. It has a 2.9 GHz Intel i5-9400 CPU with 6 cores and 32 GiB of memory. Benchmarks were run in parallel on all cores. Run-time was limited to 2 hours per instance by using the "`timeout 7200s`" command. Memory was limited to 4 GiB of memory per program instance and this was conveyed to the software by parameters. All listed computation times are CPU times.

The SMTcut Setup. We used SMTcut 6.1.0 for these benchmarks. SMTcut is written in Python and uses PySMT [33] as a solver-agnostic interface to the actual SMT solver. SMTcut was run on Python 3.8.10 with PySMT 0.9.0, gmpy2 2.0.8, and SymPy 1.7.1 [69]. As solvers were used: CVC4 1.7-prerelease [4], MathSAT 5.6.5 [19], Yices 2.6.2 [27], and Z3 4.8.7 [71].

The Redlog Setup. Redlog benchmarks were run with CSL Reduce revision 6110, dated "19-Oct-2021" and Redlog revision 6088 dated "2021-10-08, 16:03:01Z". The memory limit was communicated to Reduce through the "`-maxmem 4096`" option. Times in Reduce were measured with the built-in `showtime` command, which returns the time needed for the execution of the algorithm and additionally the time needed for garbage collection. The sum of both times is the total run-time that was recorded. Notice that all times returned by Reduce are multiples of 0.01 seconds.

With Redlog, five different benchmarks were run:

1. Command `rldnf` with switches `off rlbnfsac; off rlbnfsm`.
2. Command `rldnf` with switches `off rlbnfsac; on rlbnfsm`.
3. Command `rldnf` with switches `on rlbnfsac; off rlbnfsm`. This is the default switch setting.
4. Command `rldnf` with switches `on rlbnfsac; on rlbnfsm`.
5. Command `rlgsn` with option `form=dnf`.

Composition of the benchmark set. ODEbase currently contains 662 models. Of those, 199 are polynomial. We exclude 47 models that contain species with a boundary condition from our test set. Furthermore, 16 models with species with zero stoichiometry are excluded. Then, there are rare cases when the ODE right hand side

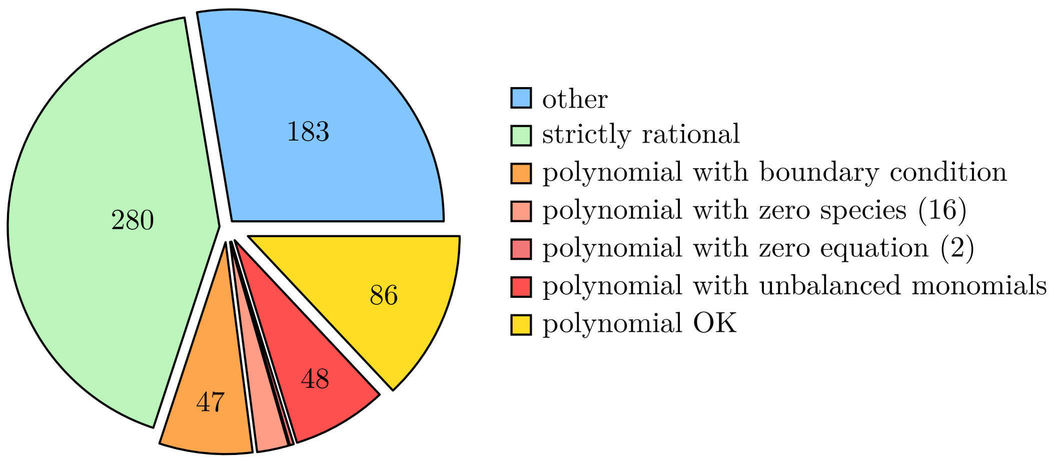


Figure 5.1: Distribution of models. Gold and reddish slices represent polynomial models that were used, respectively, were not used in our tests.

is zero because of parameters that are zero. This is true for 2 models. Excluding these three types leaves us with 134 models.

Models with unbalanced monomials are not considered as well and leaving out 48 such models leaves us with a total of 86 models as a benchmark set. See Figure 5.1 for a chart of the distribution of the different types of models.

Recall that Chapter 2 describes an approach on autonomous first-order ODEs where algebraic constraints are not covered. However, computation of the tropical equilibration for systems with algebraic constraints makes biological sense nonetheless, since it amounts to tropicalization of the steady-state variety, cf. [94]. In this section, we consider all such 86 models from ODEbase for benchmarking, regardless of possibly existing algebraic constraints and thus without regard for their applicability in the sense of Chapter 2. Nevertheless, our speed tests are performed with biologically relevant and realistic input data.

SMTcut runs. All four SMT solvers we are using support the logic `QF_LRA`. We first compare their speed with SMTcut to one another. Of the 86 models in our test set, for 68 models we could compute the DNF in less than 0.1 seconds using any solver. The maximum number of combinations in any of those runs was 1 981 355 655 168. We do not compare the run-times for those models, since measurement inaccuracies of the timings are too high to draw proper conclusions. Of the remaining 18 models that took more than 0.1 seconds to finish with any solver, only `BIOMD0000000014` did not finish within the limits of time and memory with any solver. Furthermore, MathSAT crashed after some time while SMTcut was computing the DNF of `BIOMD0000000103`.

Figure 5.2 shows the run-times of the different solvers in comparison to the best run-time with any solver. See Table A.1 for the exact timings of all solver runs that took at least 0.1 seconds. The geometrical means for all models that finished in at least 0.1 seconds with all four solvers are:

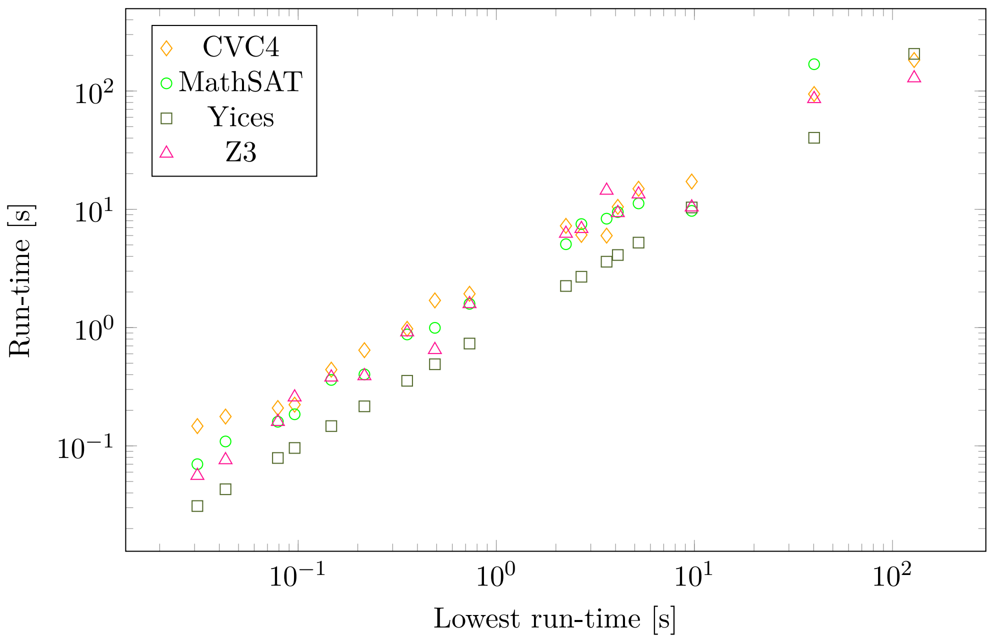


Figure 5.2: SMTcut run-times with different solvers vs. fastest run-time with any solver

Solver	Geometric mean [s]	Factor to Yices time
Yices	0.739	—
Z3	1.591	2.15
MathSAT	1.630	2.20
CVC4	2.015	2.72

Redlog runs. We first compare the run-times of Redlog with the five different commands and switch settings to one another. Of the 86 models in our test set, Redlog could compute a DNF for 55 models using any command in less than 0.1 seconds. The maximum number of combinations in any of those runs was 2 592 000. We do not include these models in our overview, since time measurement inaccuracies make drawing conclusions unreliable. For computations where at least one command took at least 0.1 seconds, both Redlog commands `rldnf` with switch `on rlbnfsm` were always slower than any of the other Redlog commands. Sometimes, this slowdown was extreme, as in the case of `BIOMD0000000940`, where it is about 300 times slower. See Figure 5.3 for a scatter plot of run-times for the different commands, or see Table A.2 for details.

For computing a DNF from a CDC-form, `rldnf` with switch `on rlbnfsm` is a comparatively slow choice. This indicates that `off rlbnfsm` as the default switch setting for `rldnf` is a useful choice, at least for our purpose here.

Disregarding the commands `on rlbnfsm`, the geometrical means for all models that finished with all other commands are:

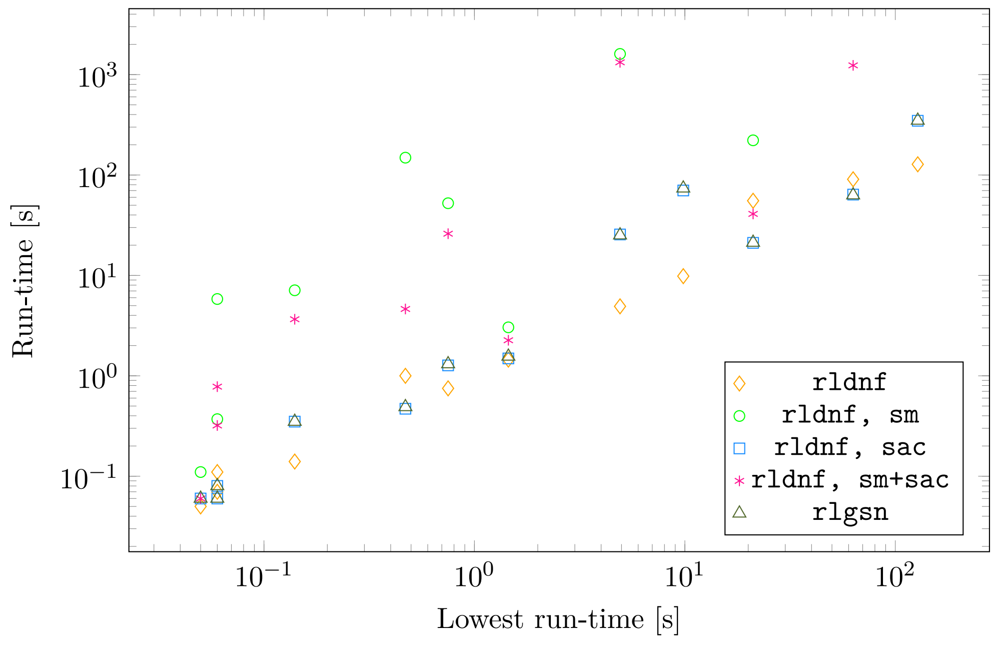


Figure 5.3: Redlog run-times for different commands vs. fastest run-time

Command	Geometric mean [s]	Factor to fastest time
`rldnf; off rlbnfsac`	1.869	—
`rldnf; on rlbnfsac`	2.549	1.36
`rlgsn`	2.582	1.38

Comparison of SMTcut vs. Redlog. For the comparison of run-times between Redlog and SMTcut, we use the fastest timing that was measured with any Redlog command against the fastest time for SMTcut with any solver. Again, we only consider run-time measurements where at least one of both, SMTcut or Redlog, took at least 0.1 seconds. That leaves out 64 models. Moreover, 13 models could not be computed by Redlog within the time and memory limits and thus no direct comparison can be made. This leaves 9 models where both programs finished computation.

The geometric means of run-times for models where both programs finished in at least 0.1 seconds are:

Program	Geometric mean [s]	Speed-up
SMTcut	0.013	—
Redlog	4.376	338

Figure 5.4 depicts a graph of the speed-up of SMTcut vs. Redlog. The exact timings are in Table A.3.

Conclusion. SMTcut was significantly faster than Redlog in computing a DNF from a CDC-form, on average by a factor of 338. Furthermore, several computations could be

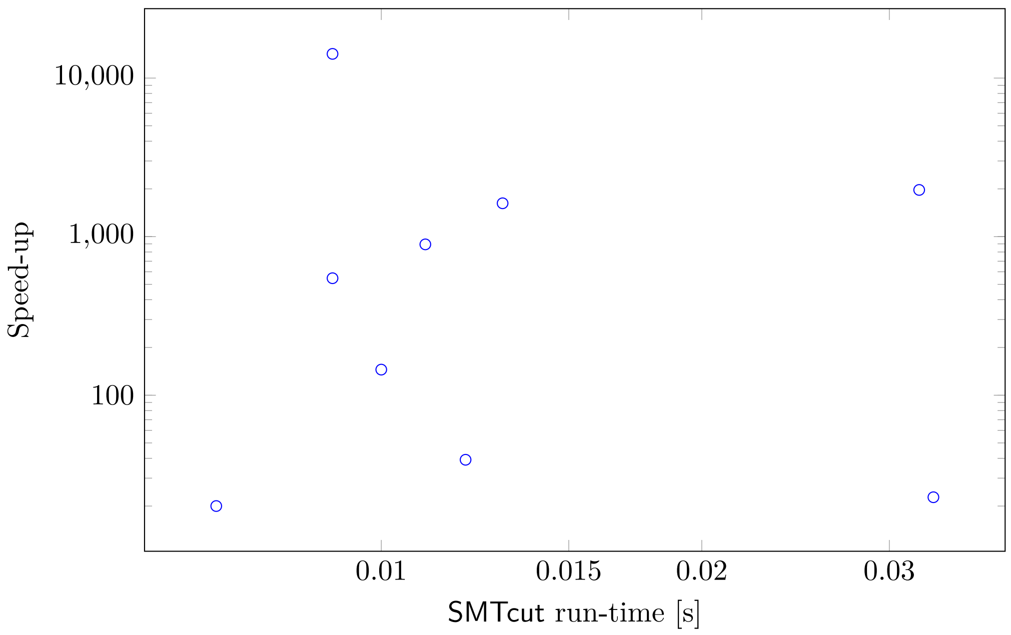


Figure 5.4: Redlog run-times vs. SMTcut run-times

finished within seconds with SMTcut where Redlog stopped due to memory exhaustion. Only one computation, derived from `BIOMD0000000014`, could not be finished at all, neither by SMTcut nor by Redlog.

5.2 Comparison of SMTcut with Prior Work

Now we compare SMTcut with results from Samal [92]. They also had written algorithms to compute the tropical equilibration, namely their Algorithm 2 in Section 4.3. We have communicated with the author of [92] and obtained a copy of the data set of run-times that led to their Table 4.2. They used $\varepsilon = \frac{1}{5}$, while a value for p could not be set in their algorithms, implying $p = 1$. Their benchmarks were performed on an Intel Core 2 Quad with 2.66 GHz and 4 GiB of memory.

For a balanced comparison, we have to use their kind of tropicalization that is slightly different from ours and is explained in their work. The differences amount to how the logarithms of parameter values are computed and whether literal constants in the polynomials are ignored. Regarding the computation of the tropical equilibration from already tropicalized polynomials, their and our algorithms return equivalent results.

Furthermore, the run-times for SMTcut in this section are not comparable with those of the last section, since in this section the time for tropicalization is included in the run-time, whereas in the last section only the time required to compute a DNF from a CDC-form was measured. Moreover, we obtained the original ODE systems that were used in [92], which sometimes differ from the data sets from ODEbase used in the last section.

We used Yices 2.6.2 as SMT solver for our benchmarks here. No run-time or memory limit was set. Our benchmarks were done on a 2.9 GHz Intel i5-9400 CPU with 6 cores

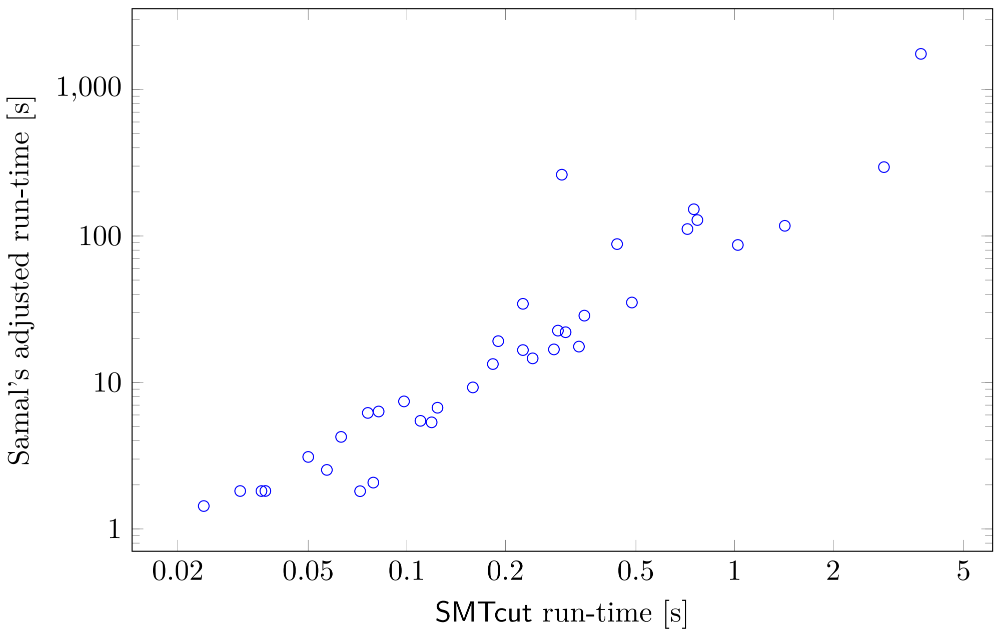


Figure 5.5: Run-times from Samal [92] vs. SMTcut run-times

and 32 GiB of memory. The computations in [92] were performed in 2016 and the machine used was slower than the one used for our tests here. To compensate for that fact, we consulted the PassMark CPU benchmark overview[1] to get relative speed ratings of the CPUs in question. Their ratings are a good indicator for the speed of a CPU. For our i5-9400 we find a single-thread rating of 2486, while Samal's Core 2 Quad has a rating of 1122–1153, depending on the exact model. Therefore, to make up for the enhanced speed of our more recent machine, we divide their run-times by a factor of $\frac{2486 \cdot 2}{1122+1153} \approx 2.19$.

Table A.4 shows the run-times for each BioModel under consideration for the two algorithms and Figure 5.5 has a plot of run-times in comparison. All times are CPU times.

The geometric mean of all adjusted run-times for Samal is 16.338 seconds, and the geometric mean of run-times for SMTcut is 0.201 seconds. The average speed-up between their adjusted run-time and SMTcut's run-time is 114.

5.3 Overview of Reduction Run-Times

Recall the pipeline of reduction Algorithms 1–12 as depicted in Figure 2.1. We would like to give an impression about the run-time of the full chain of algorithms. Our algorithms can be applied to all models from ODEbase with polynomial ODEs where some additional technical constraints are satisfied as described in Section 5.1. See Figure 5.1 for a breakdown of models and the reasons why our reduction could not be applied to some of them. We used our reduction procedure on all 86 models

[1]`https://www.cpubenchmark.net/`

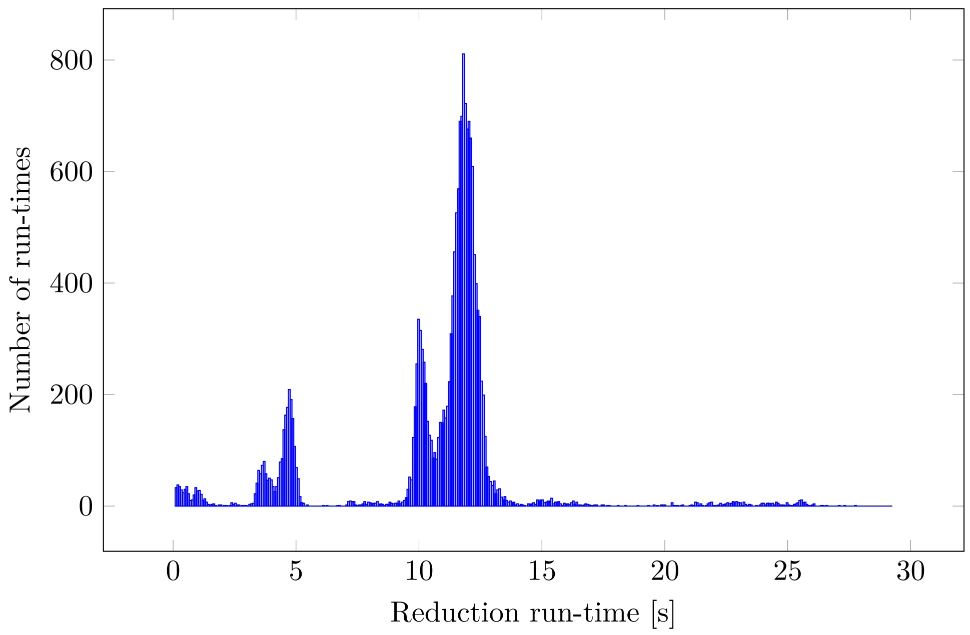


Figure 5.6: Histogram of run-times for the reduction process excluding Algorithm 11. Only run-times less than 30 seconds are displayed.

of the category "polynomial OK", except for `BIOMD0000000014`, where the tropical equilibration could not be computed within the time limit of 2 hours. As values for ε_* we used $\frac{1}{2}$, $\frac{1}{5}$, $\frac{1}{11}$, and $\frac{1}{101}$.

As discussed in the first section of this chapter, Algorithm 11 is a potential bottleneck for computation and its timings have been discussed separately before. Since Algorithm 11 is often the dominant part of the computation, the run-time of the rest of the reduction is hard to assess when the overall sum of run-times is considered. This is why here we have used SMTcut to compute and save the tropical equilibration before the reduction procedure started. Thus, when benchmarking the whole reduction procedure there was no call to Algorithm 11, but its pre-computed results were read instead.

Algorithm 11 returns the tropical equilibration, which describes one or multiple polyhedra. Yet, the number of polyhedra vastly differs from one model to another and also with different values for ε_*. For example, the tropical equilibration for `BIOMD0000000103` with $\varepsilon_* = \frac{1}{5}$ and $p = 1$ has 2889 polyhedra, whereas `BIOMD0000000647` with $\varepsilon_* = \frac{1}{5}$ and $p = 1$ has only one polyhedron as its tropical equilibration. Since there is no a priori "right" polyhedron to choose a value for D from, we chose up to three different values from every polyhedron for every model and ran the reduction process with each of them. This resulted in 17580 values for run-times of successful runs for different models, values of ε_*, and values for D.

Figure 5.6 shows a histogram of the 17510 run-times for the reduction procedure excluding Algorithm 11, where the run-time was less than 30 seconds.

If we additionally exclude the run-time of Algorithm 4 from the sum of run-times, a histogram of the 17507 run-times where the run-time was less than 20 seconds is shown in Figure 5.7.

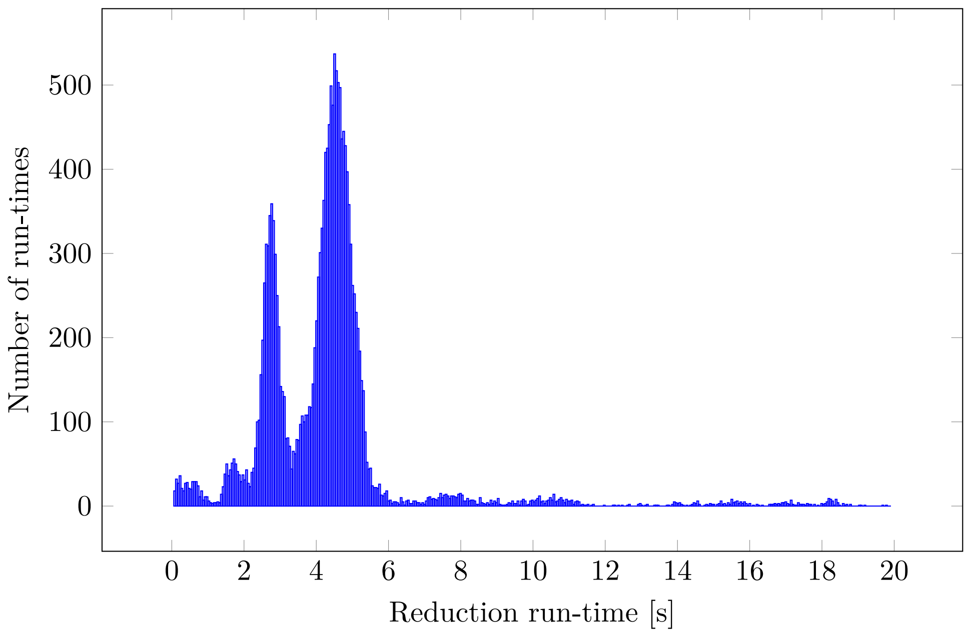


Figure 5.7: Histogram of run-times for the reduction process excluding Algorithms 4 and 11. Only run-times less than 20 seconds are displayed.

There were 1628 runs where no reduction could be computed within 20 seconds.

Chapter 6

Computational Examples

Based on our algorithms in Chapters 2 and 3, we have developed a software prototype in Python realizing all methods described in this thesis. The prototype uses SymPy [69] for symbolic computation and PySMT [33] as an interface to the SMT solver Z3 [71] that was used with Algorithms 3, 6, and 7. We used Python 3.8.10, PySMT 0.9.0, gmpy2 2.0.8, SymPy 1.7.1, and Z3 4.8.7. As with the benchmarks, we have conducted our computations on a standard desktop computer with an Intel i5-9400 CPU, 2.9 GHz, 6 cores, and 32 GB of main memory. The machine was running Ubuntu Linux 20.04.3 in 64-bit mode with kernel version 5.13.0. Computation times listed are CPU times. For a general overview of computation times for our algorithms, see Chapter 5. Parts of this chapter were previously published in [51, Section 7].

In the next section, we discuss in detail the computations for `BIOMD0000000716`. The symbolic computation input is taken from ODEbase and originates from the BioModels database [77], as described in Chapter 4. Subsequent sections showcase several further examples in a more concise style. The focus here is on biological results. For an illustration of our algorithms, we discuss in Section 6.8 examples where the reduction stops at $\ell < m$ for various reasons.

6.1 An Epidemic Model of Avian Influenza H5N6

We consider `BIOMD0000000716`, which is related to the transmission dynamics of subtype H5N6 of the influenza A virus in the Philippines in August 2017 [56]. The model specifies four species: *Susceptible birds* (`S_b`), *Infected birds* (`I_b`), *Susceptible humans* (`S_h`), and *Infected humans* (`I_a`), the concentrations of which over time we map to differential variables $x_1, \ldots, x_4$, respectively. The input system is given by

$$\begin{aligned} S = [\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 &= -\tfrac{9137}{2635182}x_1x_2 - \tfrac{1}{730}x_1 + \tfrac{412}{73}, \\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_2 &= \tfrac{9137}{2635182}x_1x_2 - \tfrac{4652377}{961841430}x_2, \\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 &= -\tfrac{1}{6159375000}x_2x_3 - \tfrac{1}{25258}x_3 + \tfrac{40758549}{3650000}, \\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 &= \tfrac{1}{6159375000}x_2x_3 - \tfrac{112500173}{2841525000000}x_4]. \end{aligned}$$

We choose $\varepsilon_* = \frac{1}{5}$, $p = 1$, and Algorithm 3 nondeterministically selects $D = (-1, -4, -7, -3)$ from the tropical equilibration. Algorithm 1 then yields the following scaled and truncated system with three time scales:

$$\begin{aligned}
T_1 &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_1 = 1 \cdot \left(-\tfrac{5710625}{2635182}y_1y_2 + \tfrac{412}{365}\right)\right],\\
T_2 &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_2 = \delta^3 \cdot \left(\tfrac{5710625}{2635182}y_1y_2 - \tfrac{116309425}{192368286}y_2\right)\right],\\
T_3 &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_3 = \delta^6 \cdot \left(-\tfrac{15625}{25258}y_3 + \tfrac{40758549}{18250000}\right),\right.\\
&\quad\ \left.\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_4 = \delta^6 \cdot \left(\tfrac{15625}{15768}y_2y_3 - \tfrac{112500173}{181857600}y_4\right)\right].
\end{aligned}$$

Notice that the lexicographic order of the differential variables is coincidence. From this input, Algorithm 5 produces the following reduced systems:

$$\begin{aligned}
M_0 &= [\,],\\
T_1 &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_1 = 1 \cdot \left(-\tfrac{5710625}{2635182}y_1y_2 + \tfrac{412}{365}\right)\right],\\
R_1 &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}\tau}y_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}\tau}y_4 = 0\right],
\end{aligned}$$

$$\begin{aligned}
M_1 &= \left[y_1y_2 = \tfrac{1085694984}{2084378125}\right],\\
T_2 &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_2 = \delta^3 \cdot \left(\tfrac{5710625}{2635182}y_1y_2 - \tfrac{116309425}{192368286}y_2\right)\right],\\
R_2 &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}\tau}y_4 = 0\right],
\end{aligned}$$

$$\begin{aligned}
M_2 &= \left[y_1y_2 = \tfrac{1085694984}{2084378125}, \quad 16675025y_1y_2 - 4652377y_2 = 0\right],\\
T_3 &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_3 = \delta^6 \cdot \left(-\tfrac{15625}{25258}y_3 + \tfrac{40758549}{18250000}\right),\right.\\
&\quad\ \left.\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_4 = \delta^6 \cdot \left(\tfrac{15625}{15768}y_2y_3 - \tfrac{112500173}{181857600}y_4\right)\right],\\
R_3 &= [\,].
\end{aligned}$$

In that course, Algorithm 6 confirms hyperbolic attractivity for all three scaled systems. Furthermore, Algorithm 7 applies the sufficient smoothness test with

$$\ell = 3, \quad b_1 = 3, \quad b_2 = 3, \quad P_1 = 1 \cdot (-\delta^4 \cdot \tfrac{125}{146}y_1), \quad P_2 = \delta^6 \cdot (-\delta^4 \cdot \tfrac{15625}{15768}y_2y_3).$$

This yields $E = \{4\}$, where 4 cannot be expressed as an integer multiple of 3. Thus the test fails, which causes a warning in Algorithm 5.

Algebraic simplification through Algorithm 8 yields the simplified reduced systems

$$\begin{aligned}
M_0' &= [\,],\\
T_1' &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_1 = 1 \cdot \left(-\tfrac{5710625}{2635182}y_1y_2 + \tfrac{412}{365}\right)\right],\\
R_1' &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}\tau}y_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}\tau}y_4 = 0\right],
\end{aligned}$$

$$\begin{aligned}
M_1' &= \left[y_1y_2 = \tfrac{1085694984}{2084378125}\right],\\
T_2' &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_2 = \delta^3 \cdot \left(-\tfrac{116309425}{192368286}y_2 + \tfrac{412}{365}\right)\right],\\
R_2' &= \left[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}\tau}y_4 = 0\right],
\end{aligned}$$

$$
\begin{aligned}
M_2' &= \bigl[y_1 = \tfrac{4652377}{16675025}, \quad y_2 = \tfrac{10856694984}{581547125}\bigr],\\
T_3' &= \bigl[\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_3 = \delta^6 \cdot \left(-\tfrac{15625}{25258}y_3 + \tfrac{40758549}{18250000}\right),\\
&\qquad \tfrac{\mathrm{d}}{\mathrm{d}\tau}y_4 = \delta^6 \cdot \left(\tfrac{1884887125}{1018870563}y_3 - \tfrac{112500173}{181857600}y_4\right)\bigr],\\
R_3' &= [\,].
\end{aligned}
$$

Notice that our implementation conveniently rewrites equational constraints as monomial equations with numerical right hand sides when possible. This supports readability but is not essential for the simplifications applied here, which are based on Gröbner basis theory. Comparing T_2' with T_2, we see that the equation for y_1y_2 in M_1' is plugged in. Similarly, M_2 is simplified to M_2', which is in turn used to reduce T_3 to T_3'.

The back-transformed reduced systems as computed by Algorithm 9 read as follows:

$$
\begin{aligned}
M_0^* &= [\,],\\
T_1^* &= \bigl[\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = 1 \cdot \left(-\tfrac{9137}{2635182}x_1x_2 + \tfrac{412}{73}\right)\bigr],\\
R_1^* &= \bigl[\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0\bigr],
\end{aligned}
$$

$$
\begin{aligned}
M_1^* &= \bigl[x_1x_2 = \tfrac{10856694984}{667001}\bigr],\\
T_2^* &= \bigl[\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = \tfrac{1}{125} \cdot \left(-\tfrac{116309425}{192368286}x_2 + \tfrac{51500}{73}\right)\bigr],\\
R_2^* &= \bigl[\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0\bigr],
\end{aligned}
$$

$$
\begin{aligned}
M_2^* &= \bigl[x_1 = \tfrac{4652377}{3335005}, \quad x_2 = \tfrac{5428474920}{4652377}\bigr],\\
T_3^* &= \bigl[\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = \tfrac{1}{15625} \cdot \left(-\tfrac{15625}{25258}x_3 + \tfrac{203792745}{1168}\right),\\
&\qquad \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = \tfrac{1}{15625} \cdot \left(\tfrac{15079097}{5094352815}x_3 - \tfrac{112500173}{181857600}x_4\right)\bigr],\\
R_3^* &= [\,].
\end{aligned}
$$

We compare T_1^*, …, T_3^* to the input system S: In the equation for $\frac{\mathrm{d}}{\mathrm{d}t}x_1$, the monomial in x_1 is identified as a higher order term with respect to δ and discarded by Algorithm 1. In the equation for $\frac{\mathrm{d}}{\mathrm{d}t}x_2$, the monomial in x_1x_2 has been Gröbner-reduced to a constant modulo the defining equation in M_1'. Similarly, the equation for $\frac{\mathrm{d}}{\mathrm{d}t}x_3$ loses its monomial in x_2x_3 by truncation of higher order terms, and in the equation for $\frac{\mathrm{d}}{\mathrm{d}t}x_4$, the monomial in x_2x_3 is Gröbner-reduced to a monomial in x_3.

Notice the explicit constant factors on the right hand sides of the differential equations in T_1^*, …, T_3^*. They originate from factors δ^{b_k} in the respective scaled systems T_1, …, T_3, corresponding to (2.16). They are left explicit to make the time scale of the differential equations apparent. We see that the system $T_2^* \circ R_2^*$ is 125 times slower than $T_1^* \circ R_1^*$, and $T_3^* \circ R_3^*$ is another 125 times slower.

Figure 6.1 visualizes the direction fields of $T_1^* \circ R_1^*$, …, $T_3^* \circ R_3^*$ on their respective manifolds $\mathcal{M}_0^*$, …, $\mathcal{M}_2^*$ along with their respective critical manifolds $\mathcal{M}_1^*$, …, $\mathcal{M}_3^*$, where $\mathcal{M}_3^*$ can be derived from $\mathcal{M}_2^*$ by additionally equating the vector field of $T_3^* \circ R_3^*$ to zero:

$$
M_3^* = \bigl[x_1 = \tfrac{4652377}{3335005},\ x_2 = \tfrac{5428474920}{4652377},\ x_3 = \tfrac{7051228977}{25000},\ x_4 = \tfrac{441466240042010928888}{327120760850763125}\bigr].
$$

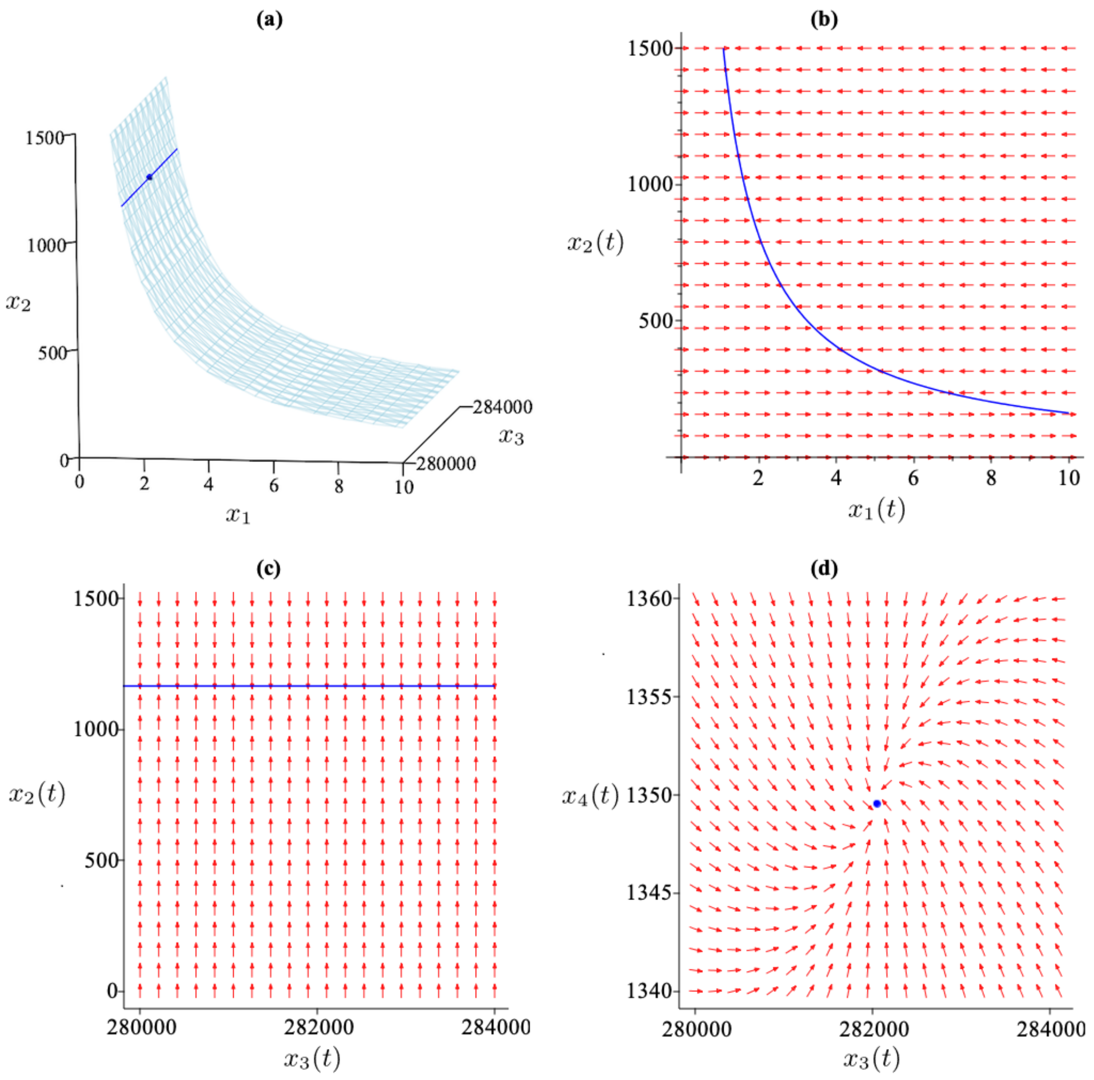


Figure 6.1: Critical manifolds and direction fields of our reduced systems for `BIOMD0000000716`. **(a)** The surface is the critical manifold $\mathcal{M}_1^* \subseteq \mathcal{M}_0^* = \mathcal{U}$ projected from $\mathbb{R}^4$ into real (x_1, x_2, x_3)-space. The line located at $(x_1, x_2) \approx (1.4, 1166.8)$ is the critical submanifold $\mathcal{M}_2^* \subseteq \mathcal{M}_1^*$. The dot located at $(x_1, x_2, x_3) \approx (1.4, 1166.8, 282049.2)$ is the critical submanifold $\mathcal{M}_3^* \subseteq \mathcal{M}_2^*$. Both $\mathcal{M}_1^*$ and $\mathcal{M}_2^*$ extend to $\pm\infty$ in both x_3 and x_4 direction, and $\mathcal{M}_3^*$ is located near $(1.4, 1166.8, 282049.2, 1349.6)$. **(b)** The direction field of $T_1^* \circ R_1^*$ on $\mathcal{M}_0^* = \mathcal{U}$ projected from $\mathbb{R}^4$ into real (x_1, x_2)-space. The curve is the critical submanifold $\mathcal{M}_1^* \subseteq \mathcal{M}_0^*$. **(c)** The direction field of $T_2^* \circ R_2^*$ on $\mathcal{M}_1^*$ projected from $\mathbb{R}^4$ into real (x_3, x_2)-space. The line is the critical submanifold $\mathcal{M}_2^* \subseteq \mathcal{M}_1^*$. The system here is slower than the one in (b) by a factor of 125. **(d)** The direction field of $T_3^* \circ R_3^*$ on $\mathcal{M}_2^*$ projected from $\mathbb{R}^4$ into real (x_3, x_4)-space. The dot is the critical submanifold $\mathcal{M}_3^* \subseteq \mathcal{M}_2^*$. The system here is slower than the one in (c) by another factor of 125.

This list M_3^* does not explicitly occur in the output. However, its preimage M_3 is constructed in Algorithm 5 and justifies the presence of (M_2, T_3, R_3) in the output there. The total computation time was 0.464 seconds.

This multiple-time scale reduction of the bird flu model emphasizes a cascade of successive relaxations of model variables. First, the population of susceptible birds relaxes. This relaxation is illustrated in Fig. 6.1(b). Then, the population of infected birds relaxes as shown in Fig. 6.1(c). Finally, the populations of susceptible and infected humans relax to a stable steady state as shown in Fig. 6.1(d), following a reduced dynamics described by T_3^*.

6.2 TGF-β Pathway

`BIOMD0000000101` is a simple representation of the TGF-β signaling pathway that plays a central role in tissue homeostasis and morphogenesis, as well as in numerous diseases such as fibrosis and cancer [105]. Concentrations over time of species *Receptor 1* (`RI`), *Receptor 2* (`RII`), *Ligand receptor complex-plasma membrane* (`lRIRII`), *Ligand receptor complex-endosome* (`lRIRII_endo`), *Receptor 1 endosome* (`RI_endo`), and *Receptor 2 endosome* (`RII_endo`), are mapped to differential variables $x_1, \ldots, x_6$, respectively. The original `BIOMD0000000101` has a change of `ligand` concentration at time $t = 2500$ from $3 \cdot 10^{-5}$ to 0.01 to reach the steady state, as in the paper. For our computation here, we use this changed value of 0.01. The input system is given by

$$
\begin{aligned}
S = [\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 &= -\tfrac{1}{100}x_1x_2 - \tfrac{3611111111333}{10000000000000}x_1 + \tfrac{333333333333333}{10000000000000000}x_4 + \tfrac{333333333333333}{10000000000000000}x_5 + 8,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 &= -\tfrac{1}{100}x_1x_2 - \tfrac{3611111111333}{10000000000000}x_2 + \tfrac{333333333333333}{10000000000000000}x_4 + \tfrac{333333333333333}{10000000000000000}x_6 + 4,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 &= \tfrac{1}{100}x_1x_2 - \tfrac{6111111111333}{10000000000000}x_3,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 &= \tfrac{3333333333333}{10000000000000}x_3 - \tfrac{333333333333333}{10000000000000000}x_4,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 &= \tfrac{3333333333333}{10000000000000}x_1 - \tfrac{333333333333333}{10000000000000000}x_5,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_6 &= \tfrac{3333333333333}{10000000000000}x_2 - \tfrac{333333333333333}{10000000000000000}x_6]
\end{aligned}
$$

We choose $\varepsilon_* = \frac{1}{5}$, $p = 1$, and select $D = (0, -4, -1, -2, -1, -5)$ from the tropical equilibration. Our back-transformed reduced systems read as follows:

$$
\begin{aligned}
M_0^* &= [\,],\\
T_1^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = 5 \cdot \left(-\tfrac{1}{500}x_1x_2 + \tfrac{8}{5}\right)],\\
R_1^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_6 = 0],\\[1ex]
M_1^* &= [x_1x_2 = 800],\\
T_2^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 1 \cdot \left(-\tfrac{6111111111333}{10000000000000}x_3 + 8\right)],\\
R_2^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_6 = 0],
\end{aligned}
$$

Table 6.1: Mapping of species concentrations to differential variables for the model BIOMD0000000102

Species	Species variable	Diff. variable
APAF-1	A	x_1
Caspase 9	C9	x_2
Caspase 9-XIAP complex	C9X	x_3
XIAP	X	x_4
APAF-1-caspase 9-XIAP complex	AC9X	x_5
APAF-1-caspase 9 complex	AC9	x_6
Caspase 3	C3	x_7
Caspase 3 cleaved	C3_star	x_8
Caspase 3 cleaved-XIAP complex	C3_starX	x_9
Caspase 9 cleaved-XIAP complex	C9_starX	x_{10}
Caspase 9 cleaved	C9_star	x_{11}
APAF-1-caspase 9 cleaved complex	AC9_star	x_{12}
APAF-1-caspase 9 cleaved-XIAP complex	AC9_starX	x_{13}

$$\begin{aligned}
M_2^* &= \left[x_1 x_2 = 800, \quad x_3 = \tfrac{80000000000000}{6111111111333}\right],\\
T_3^* &= \left[\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = \tfrac{1}{5}\cdot\left(-\tfrac{3611111111333}{2000000000000}x_2 + \tfrac{333333333333333}{2000000000000000}x_6\right)\right],\\
R_3^* &= \left[\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_6 = 0\right].
\end{aligned}$$

The smoothness criteria were verified. Notice the Gröbner reduction of the equation for $\frac{\mathrm{d}}{\mathrm{d}t}x_3$. The total computation time was 0.648 seconds.

The multiple-time scale reduction of the TGF-β model emphasizes a cascade of successive relaxations of concentrations of different species. First, the concentration of receptor 1 relaxes rapidly. Then follows the membrane complex, and, even slower, the relaxation of receptor 2.

6.3 Caspase Activation Pathway

BIOMD0000000102 is a quantitative kinetic model that examines the intrinsic pathway of caspase activation that is essential for apoptosis induction by various stimuli including cytotoxic stress [57]. Species concentrations over time are mapped to differential variables $x_1, \ldots, x_{13}$ as described in Table 6.1. The input system is given by

$$\begin{aligned}
S = [\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_1 &= -\tfrac{1}{500}x_1x_2 - \tfrac{1}{500}x_1x_3 - \tfrac{1}{500}x_1x_{10} - \tfrac{1}{500}x_1x_{11} - \tfrac{1}{1000}x_1 + \tfrac{1}{10}x_5 + \tfrac{1}{10}x_6\\
&\quad + \tfrac{1}{10}x_{12} + \tfrac{1}{10}x_{13} + \tfrac{1}{50},\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 &= -\tfrac{1}{500}x_1x_2 - \tfrac{1}{1000}x_2x_4 - \tfrac{1}{5000}x_2x_8 - \tfrac{1}{1000}x_2 + \tfrac{1}{1000}x_3 + \tfrac{1}{10}x_6 + \tfrac{1}{50},\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 &= -\tfrac{1}{500}x_1x_3 + \tfrac{1}{1000}x_2x_4 - \tfrac{1}{500}x_3 + \tfrac{1}{10}x_5,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 &= -\tfrac{1}{1000}x_2x_4 + \tfrac{1}{1000}x_3 - \tfrac{1}{1000}x_4x_6 - \tfrac{3}{1000}x_4x_8 - \tfrac{1}{1000}x_4x_{11} - \tfrac{1}{1000}x_4x_{12}\\
&\quad - \tfrac{1}{1000}x_4 + \tfrac{1}{1000}x_5 + \tfrac{1}{1000}x_9 + \tfrac{1}{1000}x_{10} + \tfrac{1}{1000}x_{13} + \tfrac{1}{25},
\end{aligned}$$

$$\begin{aligned}
\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 &= \tfrac{1}{500}x_1x_3 + \tfrac{1}{1000}x_4x_6 - \tfrac{51}{500}x_5,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_6 &= \tfrac{1}{500}x_1x_2 - \tfrac{1}{1000}x_4x_6 + \tfrac{1}{1000}x_5 - \tfrac{1}{5000}x_6x_8 - \tfrac{101}{1000}x_6,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_7 &= -\tfrac{1}{200000}x_2x_7 - \tfrac{7}{20000}x_6x_7 - \tfrac{1}{20000}x_7x_{11} - \tfrac{7}{2000}x_7x_{12} - \tfrac{1}{1000}x_7 + \tfrac{1}{5},\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_8 &= \tfrac{1}{200000}x_2x_7 - \tfrac{3}{1000}x_4x_8 + \tfrac{7}{20000}x_6x_7 + \tfrac{1}{20000}x_7x_{11} + \tfrac{7}{2000}x_7x_{12}\\
&\quad - \tfrac{1}{1000}x_8 + \tfrac{1}{1000}x_9\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_9 &= \tfrac{3}{1000}x_4x_8 - \tfrac{1}{500}x_9,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_{10} &= -\tfrac{1}{500}x_1x_{10} + \tfrac{1}{1000}x_4x_{11} - \tfrac{1}{500}x_{10} + \tfrac{1}{10}x_{13},\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_{11} &= -\tfrac{1}{500}x_1x_{11} + \tfrac{1}{5000}x_2x_8 - \tfrac{1}{1000}x_4x_{11} + \tfrac{1}{1000}x_{10} - \tfrac{1}{1000}x_{11} + \tfrac{1}{10}x_{12},\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_{12} &= \tfrac{1}{500}x_1x_{11} - \tfrac{1}{1000}x_4x_{12} + \tfrac{1}{5000}x_6x_8 - \tfrac{101}{1000}x_{12} + \tfrac{1}{1000}x_{13},\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_{13} &= \tfrac{1}{500}x_1x_{10} + \tfrac{1}{1000}x_4x_{12} - \tfrac{51}{500}x_{13}].
\end{aligned}$$

We choose $\varepsilon_* = \frac{1}{2}$, $p = 1$, and select $D = (-4, 2, 3, 5, 5, 4, -6, -8, -4, -2, -2, 0, 0)$ from the tropical equilibration. Our back-transformed reduced systems read as follows:

$$\begin{aligned}
M_0^* &= [\,],\\
T_1^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 1\cdot\left(-\tfrac{3}{1000}x_4x_8 + \tfrac{1}{25}\right)],\\
R_1^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_6 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_{12} = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_{13} = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0,\\
&\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_{10} = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_{11} = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_9 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_8 = 0],
\end{aligned}$$

$$\begin{aligned}
M_1^* &= [x_4x_8 = \tfrac{40}{3}],\\
T_2^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = \tfrac{1}{8}\cdot\left(\tfrac{2}{125}x_1x_3 - \tfrac{102}{125}x_5\right),\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_6 = \tfrac{1}{8}\cdot\left(\tfrac{2}{125}x_1x_2 - \tfrac{101}{125}x_6\right),\\
&\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_{12} = \tfrac{1}{8}\cdot\left(\tfrac{2}{125}x_1x_{11} - \tfrac{101}{125}x_{12}\right),\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_{13} = \tfrac{1}{8}\cdot\left(\tfrac{2}{125}x_1x_{10} - \tfrac{102}{125}x_{13}\right)],\\
R_2^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_{10} = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_{11} = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0,\\
&\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_9 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_8 = 0],
\end{aligned}$$

$$\begin{aligned}
M_2^* &= [x_4x_8 = \tfrac{40}{3},\quad x_1x_3 - 51x_5 = 0,\quad 2x_1x_2 - 101x_6 = 0,\\
&\quad 2x_1x_{11} - 101x_{12} = 0,\quad x_1x_{10} - 51x_{13} = 0],\\
T_3^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = \tfrac{1}{16}\cdot\left(-\tfrac{2}{625}x_2x_8 + \tfrac{8}{25}\right)],\\
R_3^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_{10} = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_{11} = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_9 = 0,\\
&\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_8 = 0].
\end{aligned}$$

The smoothness criteria were verified. Gröbner reduction did not simplify any polynomial. The total computation time was 4.16 seconds, of which Algorithm 4 took 2.838 seconds.

The multiple-time scale reduction of the caspase activation model emphasizes a cascade of successive relaxations. First, the inhibitor of apoptosis XIAP binds rapidly to the cleaved caspase. Then, the four APAF complexes are formed. Finally, the caspase 9 is recruited to the apoptosome.

6.4 Avian Influenza Bird-To-Human Transmission

`BIOMD0000000709` describes bird-to-human transmission of different strains of avian influenza A viruses, such as H5N1 and H7N9 [58]. Species concentrations over time of *Susceptible avians* (`S_a`), *Infected avians* (`I_a`), *Susceptible humans* (`S_h`), *Infected humans* (`I_h`), and *Recovered humans* (`R_h`) are mapped to differential variables x_1, …, x_5, respectively. The input system is given by

$$\begin{aligned}
S = \big[&\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = -\tfrac{1}{8000000000}x_1^3 + \tfrac{127}{20000000}x_1^2 - \tfrac{9}{500000000}x_1x_2 - \tfrac{1}{200}x_1,\\
&\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = \tfrac{9}{500000000}x_1x_2 - \tfrac{37123}{50000000}x_2,\\
&\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = -\tfrac{3}{500000000}x_2x_3 - \tfrac{391}{10000000}x_3 + 30,\\
&\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = \tfrac{3}{500000000}x_2x_3 - \tfrac{4445391}{10000000}x_4,\\
&\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = \tfrac{1}{10}x_4 - \tfrac{391}{10000000}x_5\big].
\end{aligned}$$

We choose $\varepsilon_* = \frac{1}{5}$, $p = 1$, and select $D = (-7, 0, -8, 3, -2)$ from the tropical equilibration. Our back-transformed reduced systems read as follows:

$$\begin{aligned}
M_0^* &= [\,],\\
T_1^* &= \big[\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = 1 \cdot \big(-\tfrac{1}{8000000000}x_1^3 + \tfrac{127}{20000000}x_1^2\big)\big],\\
R_1^* &= \big[\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0\big],\\[1ex]
M_1^* &= \big[x_1^3 - 50800x_1^2 = 0\big],\\
T_2^* &= \big[\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = \tfrac{1}{5} \cdot \big(\tfrac{3}{100000000}x_2x_3 - \tfrac{4445391}{2000000}x_4\big)\big],\\
R_2^* &= \big[\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0\big].
\end{aligned}$$

The smoothness criteria were verified. Gröbner reduction did not simplify any polynomial. The total computation time was 0.317 seconds.

The multiple-time scale reduction of this avian influenza model emphasizes a cascade of successive relaxations of different model variables. First, the susceptible bird population relaxes rapidly. The reduced equation T_1 and manifold M_1 suggest that the bird population dynamics is of the Allee type and evolves toward the stable extinct state. It follows the relaxation of the infected human population that also evolves toward the extinct state, the end of the epidemics.

6.5 Spontaneous Oscillations in Excitable Cells of *Dictyostelium*

`BIOMD0000000099` describes interacting proteins that account for spontaneous oscillations in adenylyl cyclase activity that are observed in homogenous populations of Dictyostelium cells [54]. Species concentrations over time are mapped to differential variables x_1, …, x_7 as described in Table 6.2. The input system is given by:

Table 6.2: Mapping of species concentrations to differential variables for the model BIOMD0000000099

Species	Species variable	Diff. variable
Extracellular cAMP	Ex_cAMP	x_1
Intracellular cAMP	In_cAMP	x_2
Protein kinase PKA	PKA	x_3
Cytoplasmic phosphodiesterase	REGA	x_4
Adenylyl cyclase	ACA	x_5
cAMP receptor	CAR1	x_6
Protein kinase ERK2	ERK2	x_7

$$\begin{aligned} S = [&\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = -\tfrac{31}{10}x_1 + \tfrac{3}{5}x_5,\\ &\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = -x_2x_4 + \tfrac{29}{100}x_5,\\ &\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = \tfrac{5}{2}x_2 - \tfrac{3}{2}x_3,\\ &\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = -\tfrac{13}{10}x_4x_7 + 2,\\ &\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = -\tfrac{9}{10}x_5 + \tfrac{7}{5}x_7,\\ &\tfrac{\mathrm{d}}{\mathrm{d}t}x_6 = 33x_1 - \tfrac{9}{2}x_3x_6,\\ &\tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = -\tfrac{4}{5}x_3x_7 + \tfrac{3}{5}x_6] \end{aligned}$$

We choose $\varepsilon_* = \frac{1}{2}$, $p = 1$, and select $D = \left(\frac{7}{4}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, -\frac{5}{4}, -\frac{7}{4}, -\frac{5}{4}\right)$ from the tropical equilibration. Our back-transformed reduced systems read as follows:

$$\begin{aligned} M_0^* &= [\,],\\ T_1^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = 4 \cdot \left(-\tfrac{31}{40}x_1 + \tfrac{3}{20}x_5\right)],\\ R_1^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_6 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0],\\[1ex] M_1^* &= [31x_1 - 6x_5 = 0],\\ T_2^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_6 = 2^{\frac{3}{2}} \cdot \left(\tfrac{9}{8}\sqrt{2}x_3x_6 + \tfrac{99}{62}\sqrt{2}x_5\right)],\\ R_2^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0],\\[1ex] M_2^* &= [31x_1 - 6x_5 = 0, \quad 31x_3x_6 - 44x_5 = 0],\\ T_3^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 2^{\frac{5}{4}} \cdot \left(-\tfrac{13}{40}\sqrt[4]{8}x_4x_7 + \tfrac{1}{2}\sqrt[4]{8}\right)],\\ R_3^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0],\\[1ex] M_3^* &= [31x_1 - 6x_5 = 0, \quad 31x_3x_6 - 44x_5 = 0, \quad x_4x_7 = \tfrac{20}{13}],\\ T_4^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 2 \cdot \left(\tfrac{5}{4}x_2 - \tfrac{3}{4}x_3\right)],\\ R_4^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0], \end{aligned}$$

$$
\begin{aligned}
M_4^* &= \big[31x_1 - 6x_5 = 0, \quad 155x_2x_6 - 132x_5 = 0, \quad x_4x_7 = \tfrac{20}{13}, \quad 5x_2 - 3x_3 = 0\big],\\
T_5^* &= \big[\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 1 \cdot \Big(-\tfrac{9}{10}x_5 + \tfrac{7}{5}x_7\Big)\big],\\
R_5^* &= \big[\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0\big],\\
M_5^* &= \big[93x_1 - 28x_7 = 0, \quad 465x_2x_6 - 616x_7 = 0, \quad x_4x_7 = \tfrac{20}{13}, \quad 5x_2 - 3x_3 = 0,\\
&\qquad 9x_5 - 14x_7 = 0\big],\\
T_6^* &= \big[\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 2^{-\frac{1}{4}} \cdot \Big(-\sqrt[4]{2}x_2x_4 + \tfrac{203}{450}\sqrt[4]{2}x_7\Big)\big],\\
R_6^* &= \big[\tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0\big],\\
M_6^* &= \big[93x_1 - 28x_7 = 0, \quad 2640x_4 - 899x_6 = 0, \quad x_4x_7 = \tfrac{20}{13},\\
&\qquad 5400x_3 - 2639x_7^2 = 0, \quad 9x_5 - 14x_7 = 0, \quad 9000x_2 - 2639x_7^2 = 0\big],\\
T_7^* &= \big[\tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 2^{-\frac{1}{2}} \cdot \Big(\tfrac{1584}{899}\sqrt{2}x_4 - \tfrac{2639}{6750}\sqrt{2}x_7^3\Big)\big],\\
R_7^* &= [\ ].
\end{aligned}
$$

Notice the irrational coefficients in the scaled systems $T_1^*, \dots, T_7^*$. In the course of the computation, Algorithm 1 sets $q = 4$ in l.13 because of the denominators of the d_k. This in turn causes $\delta = \varepsilon^q = \frac{1}{2}^{\frac{1}{4}} = 2^{-\frac{1}{4}}$ and the powers of δ reappear in our equations. Furthermore, Gröbner reduction has not only simplified the manifold-defining equations M_4^*, M_5^*, and M_6^*, but also the equations for $\frac{\mathrm{d}}{\mathrm{d}t}x_2$, $\frac{\mathrm{d}}{\mathrm{d}t}x_6$, and $\frac{\mathrm{d}}{\mathrm{d}t}x_7$. The smoothness criteria were verified. The total computation time was 1.192 seconds.

The multiple-time scale reduction of the caspase activation model emphasizes a cascade of successive relaxations. The concentration of the species relax in the following order: extracellular cAMP, cAMP receptor, cytoplasmic phosphodiesterase, protein kinase PKA, adenylyl cyclase, intracellular cAMP, and protein kinase ERK2.

6.6 Potential HIV-1 Therapy With Modified Viruses

BIOMD0000000707 describes a mathematical model for a double infection with HIV-1 and a genetically engineered virus [86]. Species concentrations over time for *normal cells* (Normal_Th_cells), *pathogen virus* (Pathogen_Virus), *single-infected cells* (Single_Infected_Th_Cells), *recombinant virus* (Recombinant_Virus), and *doubled-infected cells* (Double_Infected_Th_Cells) are mapped to differential variables x_1, ..., x_5, respectively. The input system is given by:

$$
\begin{aligned}
S = \big[\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 &= -\tfrac{1}{250}x_1x_2 - \tfrac{1}{100}x_1 + 2,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 &= -2x_2 + 50x_3,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 &= \tfrac{1}{250}x_1x_2 - \tfrac{1}{250}x_3x_4 - \tfrac{33}{100}x_3,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 &= -2x_4 + 2000x_5,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 &= \tfrac{1}{250}x_3x_4 - 2x_5\big]
\end{aligned}
$$

We choose $\varepsilon_* = \frac{1}{2}$, $p = 1$, and select $D = (-5, -4, 1, -10, 0)$ from the tropical equilibration. Our back-transformed reduced systems read as follows:

$$\begin{aligned}
M_0^* &= [\,],\\
T_1^* &= \left[\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 4\cdot\left(\tfrac{1}{1000}x_1x_2 - \tfrac{1}{1000}x_3x_4\right)\right],\\
R_1^* &= \left[\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = 0\right],
\end{aligned}$$

$$\begin{aligned}
M_1^* &= [x_1x_2 - x_3x_4 = 0],\\
T_2^* &= \left[\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 2\cdot(-x_2 + 25x_3)\,,\quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 2\cdot(-x_4 + 1000x_5)\,,\right.\\
&\qquad \left.\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 2\cdot\left(\tfrac{1}{500}x_1x_2 - x_5\right)\right],\\
R_2^* &= \left[\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = 0\right],
\end{aligned}$$

$$\begin{aligned}
M_2^* &= [x_2 - 25x_3 = 0,\quad 2x_2x_5 - 25x_5 = 0,\quad x_1x_2 - 500x_5 = 0,\\
&\qquad x_4 - 1000x_5 = 0,\quad x_1x_5 - 40x_5^2 = 0],\\
T_3^* &= \left[\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = \tfrac{1}{16}\cdot(-32x_5 + 32)\right],\\
R_3^* &= [\,].
\end{aligned}$$

The smoothness criteria were verified. Notice that the Gröbner simplification reduced the equations for $\frac{\mathrm{d}}{\mathrm{d}t}x_1$ and $\frac{\mathrm{d}}{\mathrm{d}t}x_5$. It also reduced M_2^* and introduced one additional term in the process. Before the reduction, we had

$$\begin{aligned}
M_2 = [&x_1x_2 - x_3x_4 = 0,\quad 32x_2 - 25x_3 = 0,\\
&128x_4 - 125x_5 = 0,\quad 128x_3x_4 - 125x_5 = 0]
\end{aligned}$$

and obtained through simplification

$$\begin{aligned}
M_2^* = [&32x_2 - 25x_3 = 0,\quad 32x_2x_5 - 25x_5 = 0,\\
&128x_1x_2 - 125x_5 = 0,\quad 128x_4 - 125x_5 = 0,\quad 4x_1x_5 - 5x_5^2 = 0].
\end{aligned}$$

The computation time was 0.755 seconds.

The multiple-time scale reductions of the double virus infection model emphasizes a cascade of successive relaxations. First, the number of single infected cells relaxes, next are the pathogen virus, recombinant virus, and double-infected cells. Lastly, the number of normal cells relaxes.

6.7 Innate Immune Response to Influenza Virus

`BIOMD0000000710` describes a mathematical model based on interferon-induced resistance to infection of respiratory epithelial cells and the clearance of infected cells by natural killers [40]. Species concentrations over time are mapped to differential variables $x_1, \ldots, x_7$ as described in Table 6.3.

Table 6.3: Mapping of species concentrations to differential variables for the model `BIOMD0000000710`

Species	Species variable	Diff. variable
Healthy cells	`U_H`	x_1
Partially infected cells	`U_E`	x_2
Infected cells	`U_I`	x_3
Resistant cells	`U_R`	x_4
Virus particles	`V`	x_5
Interferon	`IFN`	x_6
Natural killers	`K`	x_7

$$\begin{aligned}
S = [\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 &= -\tfrac{3}{1000}x_1x_5 - 7x_1x_6 - \tfrac{1}{100}x_1 + 5000000,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 &= \tfrac{3}{1000}x_1x_5 - \tfrac{3}{1000000}x_2x_7 - \tfrac{1}{2}x_2,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 &= \tfrac{1}{2}x_2 - \tfrac{3}{1000000}x_3x_7 - \tfrac{1}{100}x_3,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 &= 7x_1x_6 - \tfrac{1}{100}x_4,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 &= \tfrac{1}{100}x_3 - \tfrac{26}{5}x_5,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_6 &= \tfrac{3}{1000000}x_3 - 4x_6,\\
\tfrac{\mathrm{d}}{\mathrm{d}t}x_7 &= \tfrac{1}{1000}x_3 - \tfrac{1}{25}x_7 + 32000]
\end{aligned}$$

We choose $\varepsilon_* = \frac{1}{5}$, $p = 1$, and select $D = (-8, -10, -10, -13, -6, -1, -8)$ from the tropical equilibration. Our back-transformed reduced systems read as follows:

$$\begin{aligned}
M_0^* &= [\,],\\
T_1^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = 25 \cdot \left(-\tfrac{3}{25000}x_1x_5 - \tfrac{7}{25}x_1x_6 + 200000\right)],\\
R_1^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_6 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0],\\
M_1^* &= [3x_1x_5 + 7000x_1x_6 - 5000000000 = 0],\\
T_2^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 5 \cdot \left(\tfrac{1}{500}x_3 - \tfrac{26}{25}x_5\right), \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_6 = 5 \cdot \left(\tfrac{3}{5000000}x_3 - \tfrac{4}{5}x_6\right)],\\
R_2^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0],\\
M_2^* &= [x_1x_3 = \tfrac{260000000000000}{573}, \quad x_3 - 520x_5 = 0, \quad 3x_3 - 4000000x_6 = 0],\\
T_3^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = 1 \cdot \left(-\tfrac{3}{1000000}x_2x_7 - \tfrac{1}{2}x_2 + \tfrac{500000000}{191}\right),\\
&\qquad \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = 1 \cdot \left(\tfrac{1}{2}x_2 - \tfrac{3}{1000000}x_3x_7\right)],\\
R_3^* &= [\tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = 0, \quad \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0],
\end{aligned}$$

$$\begin{aligned}
M_3^* = [&12500000x_1 - 39x_7^2 - 6500000x_7 = 0, \quad x_3 - 520x_5 = 0,\\
&3x_3 - 4000000x_6 = 0, \quad 500000x_2 - 3x_3x_7 = 0,\\
&1719x_3x_7^2 + 286500000x_3x_7 - 250000000000000000000 = 0],\\
T_4^* = [&\tfrac{\mathrm{d}}{\mathrm{d}t}x_7 = \tfrac{1}{25} \cdot \Big(\tfrac{1}{40}x_3 - x_7 + 800000\Big)],\\
R_4^* = [&\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = 0],\\
M_4^* = [&12500000x_1 - 39x_7^2 - 6500000x_7 = 0, \quad 13x_5 - x_7 + 800000 = 0,\\
&100000x_6 - 3x_7 + 2400000 = 0, \quad 12500x_2 - 3x_7^2 + 2400000x_7 = 0,\\
&x_3 - 40x_7 + 32000000 = 0, \quad 1719x_7^3 - 1088700000x_7^2\\
&\quad - 229200000000000x_7 - 6250000000000000000 = 0],\\
T_5^* = [&\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = \tfrac{1}{125} \cdot \Big(-\tfrac{5}{4}x_4 + \tfrac{56875000000}{191}\Big)],\\
R_5^* = [&\,].
\end{aligned}$$

The smoothness criteria were verified. Notice the Gröbner simplification of M_2^*, M_3^*, and M_4^* as well as the equations for $\frac{\mathrm{d}}{\mathrm{d}t}x_2$, $\frac{\mathrm{d}}{\mathrm{d}t}x_3$, $\frac{\mathrm{d}}{\mathrm{d}t}x_4$, and $\frac{\mathrm{d}}{\mathrm{d}t}x_7$. The computation time was 3.445 seconds.

The multiple-time scale reduction of this influenza model emphasizes a cascade of successive relaxations of different model variables. First the concentration of healthy cells relaxes. Then follow the virus particles and interferon. On the next, even slower, time scale the concentration of the partially infected and infected cells relaxes. Then the concentration of natural killers relaxes, and lastly the concentration of resistant cells.

6.8 Further Computational Examples

In the last sections we have discussed computations for several systems from ODEbase. While the focus was on biological results, we discuss here examples where reduction stops at $\ell < m$ for various reasons.

6.8.1 Hypertoxicity of a Painkiller

BIOMD0000000609 describes the metabolism and the related hepatotoxicity of acetaminophen, a painkiller [85]. The species concentrations over time of *Sulphate PAPS*, *GSH*, *NAPQI*, *Paracetamol APAP*, and *Protein adducts* are mapped to differential variables x_1, …, x_5, respectively. The input system is given by

$$\begin{aligned}
S = [&\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = -226000000000000x_1x_4 - 2x_1 + \tfrac{53}{2000000000000000},\\
&\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = -1600000000000000000x_2x_3 - 2x_2 + \tfrac{687}{50000000000000000},\\
&\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = -1600000000000000000x_2x_3 - \tfrac{220063}{2000}x_3 + \tfrac{63}{200}x_4,\\
&\tfrac{\mathrm{d}}{\mathrm{d}t}x_4 = -226000000000000x_1x_4 + \tfrac{63}{2000}x_3 - \tfrac{661}{200}x_4,\\
&\tfrac{\mathrm{d}}{\mathrm{d}t}x_5 = 110x_3].
\end{aligned}$$

Table 6.4: Mapping of species concentrations to differential variables for the model BIOMD0000000726

Species	Species variable	Differential variable
Susceptible dogs	S_d	x_1
Exposed dogs	E_d	x_2
Infectious dogs	I_d	x_3
Recovered dogs	R_d	x_4
Susceptible humans	S_h	x_5
Exposed humans	E_h	x_6
Infectious humans	I_h	x_7
Recovered humans	R_h	x_8

Since there is only one monomial on the right hand side of the equation for $\frac{\mathrm{d}}{\mathrm{d}t}x_5$, equilibration is impossible. This causes Algorithm 4 to return in l.23 a disjunctive normal form Π equivalent to "false," which describes the empty set. Hence Algorithm 3 returns $\perp$, and Algorithm 1 returns the empty list. The total computation time was 0.046 seconds.

6.8.2 Transmission Dynamics of Rabies

BIOMD0000000726 examines the transmission dynamics of rabies between dogs and humans [89]. Species concentrations over time are mapped to differential variables x_1, …, x_8 as described in Table 6.4. The input system is given by

$$\begin{aligned} S = [\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 &= -\tfrac{79}{500000000}x_1x_3 - \tfrac{17}{100}x_1 + \tfrac{18}{5}x_2 + x_4 + 3000000,\\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_2 &= \tfrac{79}{500000000}x_1x_3 - \tfrac{617}{100}x_2,\\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 &= \tfrac{12}{5}x_2 - \tfrac{27}{25}x_3,\\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_4 &= \tfrac{9}{100}x_1 + \tfrac{9}{100}x_2 - \tfrac{27}{25}x_4,\\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_5 &= -\tfrac{229}{100000000000000}x_3x_5 - \tfrac{3}{1000}x_5 + \tfrac{18}{5}x_6 + x_8 + 15400000,\\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_6 &= \tfrac{229}{100000000000000}x_3x_5 - \tfrac{6543}{1000}x_6,\\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_7 &= \tfrac{12}{5}x_6 - \tfrac{1343}{1000}x_7,\\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_8 &= \tfrac{27}{50}x_6 - \tfrac{1003}{1000}x_8]. \end{aligned}$$

We choose $\varepsilon_* = \frac{1}{5}$, $p = 1$, and $D = (-10, -10, -11, -9, -14, -7, -8, -7)$ is selected by Algorithm 3 from the tropical equilibration. Algorithm 1 then yields the following scaled and truncated system with three time scales:

$$\begin{aligned} T_1 = [\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_1 &= 1 \cdot \left(-\tfrac{395}{256}y_1y_3 + \tfrac{18}{25}y_2\right),\\ \tfrac{\mathrm{d}}{\mathrm{d}\tau}y_2 &= 1 \cdot \left(\tfrac{395}{256}y_1y_3 - \tfrac{617}{500}y_2\right),\\ \tfrac{\mathrm{d}}{\mathrm{d}\tau}y_6 &= 1 \cdot \left(\tfrac{28625}{16384}y_3y_5 - \tfrac{6543}{5000}y_6\right)], \end{aligned}$$

$$\begin{aligned}T_2 = [&\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_3 = \delta\cdot\left(\tfrac{12}{25}y_2 - \tfrac{27}{25}y_3\right),\\ &\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_4 = \delta\cdot\left(\tfrac{9}{20}y_1 + \tfrac{9}{20}y_2 - \tfrac{27}{25}y_4\right),\\ &\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_7 = \delta\cdot\left(\tfrac{12}{25}y_6 - \tfrac{1343}{1000}y_7\right),\\ &\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_8 = \delta\cdot\left(\tfrac{27}{50}y_6 - \tfrac{1003}{1000}y_8\right)],\\ T_3 = [&\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_5 = \delta^5\cdot\left(\tfrac{4928}{3125} - \tfrac{15}{8}y_5\right)].\end{aligned}$$

Equating the right hand sides F_1 of the differential equations in T_1 to zero equivalently yields

$$\begin{aligned}M_1 = [&-9875y_1y_3 + 4608y_2 = 0,\quad 49375y_1y_3 - 39488y_2 = 0,\\ &17890625y_3y_5 - 13400064y_6 = 0].\end{aligned}$$

In l.1 of Algorithm 6, $U \circ M_1$ is tested for satisfiability. This fails, which means that the corresponding manifold $\mathcal{M}_1$ is empty over the positive first orthant. Consequently, Algorithms 5, 8, and 9 return empty lists. The total computation time was 0.921 seconds.

6.8.3 Negative Feedback Loop Between Tumor Suppressor and Oncogene

`BIOMD0000000156` describes the dynamics of a negative feedback loop between the tumor suppressor protein p53 and the oncogene protein Mdm2 in human cells [34]. The species concentrations over time for *P53* (`x`), *Mdm2* (`y`), and *Precursor Mdm2* (`y0`) are mapped to differential variables $x_1, \dots, x_3$, respectively. The input system is given by

$$\begin{aligned}S = [&\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 = -\tfrac{37}{10}x_1x_2 + 2x_1,\\ &\tfrac{\mathrm{d}}{\mathrm{d}t}x_2 = -\tfrac{9}{10}x_2 + \tfrac{11}{10}x_3,\\ &\tfrac{\mathrm{d}}{\mathrm{d}t}x_3 = \tfrac{3}{2}x_1 - \tfrac{11}{10}x_3].\end{aligned}$$

We choose $\varepsilon_* = \frac{1}{2}$, $p = 1$, and select $D = (2, 1, 1)$. Algorithm 1 then yields the following scaled and truncated system with two time scales:

$$\begin{aligned}T_1 = [&\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_1 = 1\cdot\left(-\tfrac{37}{40}y_1y_2 + y_1\right)],\\ T_2 = [&\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_2 = \delta\cdot\left(-\tfrac{9}{10}y_2 + \tfrac{11}{10}y_3\right),\\ &\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_3 = \delta\cdot\left(\tfrac{3}{4}y_1 - \tfrac{11}{10}y_3\right)].\end{aligned}$$

Analogously to the previous section we obtain $M_1 = [-37y_1y_2 + 40y_1 = 0]$, for which we find $\mathcal{M}_1$ to be non-empty over the positive first orthant in l.1 of Algorithm 6. However, the test for hyperbolic attractivity in l.20 fails with M_1 and the Hurwitz conditions

$$\Gamma = \left\{\tfrac{37}{40}y_2 - 1 > 0\right\},$$

so that "false" is returned. Therefore, Algorithm 5 breaks the for-loop in l.10 with $k = 1$ and returns the empty list in l.19. Obviously, the simplified and back-translated systems are empty lists as well. The total computation time was 0.172 seconds.

6.8.4 CD4 T-Cells in the Spread of HIV

`BIOMD0000000663` describes how CD4 T-cells can influence the spread of the HIV infection [108]. Species concentrations over time for *Infected T-cells* (`x`), *Uninfected T-cells* (`y`), and *Free viruses* (`v`) are mapped to variables $x_1, \ldots, x_3$, respectively. The input system is given by

$$\begin{aligned} S = [\tfrac{\mathrm{d}}{\mathrm{d}t}x_1 &= -\tfrac{1}{10}x_1^2x_3 - \tfrac{1}{10}x_1x_2x_3 + \tfrac{4}{5}x_1x_3 - \tfrac{1}{10}x_1, \\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_2 &= -\tfrac{1}{10}x_1x_2x_3 + \tfrac{1}{5}x_1x_3 - \tfrac{1}{10}x_2^2x_3 + x_2x_3 - \tfrac{1}{5}x_2, \\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_3 &= x_2 - \tfrac{1}{2}x_3]. \end{aligned}$$

We choose $\varepsilon_* = \frac{1}{2}$, $p = 5$, and $D = (1, 4, 3)$. The choice of $p = 5$ causes fractional powers of numbers in the scaled and truncated system

$$\begin{aligned} T_1 &= [\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_3 = 1 \cdot (y_2 - y_3)], \\ T_2 &= [\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_2 = \delta^7 \cdot \left(\tfrac{4}{5}\sqrt[5]{4}\,y_1y_3 - \tfrac{4}{5}\sqrt[5]{4}\,y_2\right)], \\ T_3 &= [\tfrac{\mathrm{d}}{\mathrm{d}\tau}y_1 = \delta^{12} \cdot \left(\tfrac{4}{5}\sqrt[5]{4}\,y_1y_3 - \tfrac{4}{5}\sqrt[5]{4}\,y_1\right)]. \end{aligned}$$

However, such input is not accepted with the SMT logic `QF_NRA` used in Algorithm 6. As discussed in Sect. 2.3.6, we catch the corresponding error from the SMT solver, convert to floats, and restart, which solves the problem.

Similarly to the previous example, the Hurwitz test in l.20 of Algorithm 6 succeeds for $k = 1$ but fails for $k = 2$ in Algorithm 5. Since there are fewer than two reduced systems, we return the empty list. Consequently, the list of simplified reduced systems and the corresponding list of back-transformed systems are empty as well. The total computation time was 0.426 seconds.

Chapter 7

Conclusion and Outlook

We introduced a symbolic method for the reduction of systems of ordinary differential equations that describe the species kinetics of biochemical reaction networks. Our method is applicable to systems with three or more time scales, in contrast to established approaches that are limited to only two time scales. For the first time, hyperbolic attractivity conditions are verified, which are required for the reductions to be valid. Our theoretical framework is accompanied by detailed algorithms and a prototypical implementation.

To test our new method, we have compiled ODEbase, a database of symbolic computation input of biochemical processes. It has been populated with data from BioModels, a leading database of biochemical processes [77]. ODEbase is freely available at `https://odebase.org`. From this database we have taken numerous examples and tested our algorithms with this real-world data. Several examples were presented to illustrate the feasibility of our reduction method.

As part of the scaling of ODE systems to be reduced, tropical equilibrations have to be computed. As a constituent part, the computation of a DNF of constraints over the reals was a potential bottleneck for computation. To overcome this, we presented a new algorithm together with a theoretical description in the form of a calculus. Both rely on Satisfiability Modulo Theories solvers for their calculations. This new algorithm was also prototypically implemented and benchmarked against Redlog [24]. The results indicate that the new method performs up to 10 000 times faster than Redlog. Moreover, the algorithm was able to complete computations where Redlog failed due to memory exhaustion.

Finally, we point out some problems and questions for further work. The process of verification of hyperbolic attractivity of the scaled ODE system sometimes stops early. One possible reason is the existence of linear conservation laws for the fast system that necessarily cause a zero eigenvalue of the Jacobian in Algorithm 6, l.9–10. A theorem by Schneider and Wilhelm [96] may in some cases reduce this to the hyperbolically attractive case. Furthermore, linear conservation laws must be considered subject to scalings, as is discussed by Lax and Walcher [55] for the case of two time scales. Since about half of all systems from ODEbase contain some sort of algebraic constraint, pursuing this would enhance the applicability of our method and is left for future work.

It is interesting to consider cases where hyperbolic attractivity fails, but hyperbolicity still holds. According to [18], invariant manifolds exist in these cases. This can be tested algorithmically as well, see e.g. the approach by Routh [32, Chapter V, §4], but the computations become more involved.

Moreover, it is interesting to be able to handle parametric input systems, that is, systems where the reaction constants $k_{c \to c'}$ of (1.8) are not replaced by numeric values but remain symbolic. This requires the use of quantifier elimination techniques instead of SMT solving, which is however well understood and supported [107, 26, 49]. Unfortunately, our use of tropical geometry here is not compatible with this setup, since it introduces logarithms of polynomials in the parametric coefficients in (1.30). Different tropicalization methods, which however do not fit our abstract view on scaling in Section 1.5, require only logarithms of individual parametric coefficients [94]. Such a more special form would allow the use of abstraction in the logic engine. A workaround for reaction constants close to the given numeric values however seems possible and allows to re-introduce parameters into our reduced systems.

From the perspective of a user of our algorithms, it would be desirable to automatically choose good values for ε_* and D in (1.26).

The calculus of Chapter 3 could be useful in other domains as well. Since it removes one junctor alteration by converting a CDC-form to a DNF it should be possible to apply it recursively to reduce a junctor alteration of arbitrary depth to a single DNF.

Appendix A

Tables With Run-Times

Table A.1: SMTcut run-times in seconds with different solvers for runs that took at least 0.1 seconds, sorted by minimal run-time. 'Combos' is the number of combinations; bold numbers mark the lowest run-time. 'c', 't', and 'm' signify crash, timeout, and memory exceeded, respectively.

Model	Combos	CVC4	MathSAT	Yices	Z3
BIOMD0000000030	$\approx 10^{10}$	0.147	0.070	**0.031**	0.056
BIOMD0000000002	$\approx 10^{11}$	0.177	0.109	**0.043**	0.076
BIOMD0000000500	$\approx 10^{11}$	0.209	0.160	**0.079**	0.160
BIOMD0000000011	$\approx 10^{11}$	0.223	0.185	**0.096**	0.258
BIOMD0000000028	$\approx 10^{9}$	0.441	0.361	**0.147**	0.380
BIOMD0000000637	$\approx 10^{12}$	0.644	0.402	**0.216**	0.390
BIOMD0000000599	$\approx 10^{16}$	0.976	0.875	**0.355**	0.916
BIOMD0000000365	$\approx 10^{23}$	1.697	0.995	**0.490**	0.649
BIOMD0000000147	$\approx 10^{16}$	1.931	1.588	**0.733**	1.586
BIOMD0000000492	$\approx 10^{25}$	7.245	5.068	**2.247**	6.240
BIOMD0000000102	$\approx 10^{10}$	6.076	7.501	**2.690**	6.844
BIOMD0000000407	$\approx 10^{16}$	5.990	8.331	**3.606**	14.421
BIOMD0000000431	$\approx 10^{16}$	10.491	9.472	**4.107**	9.329
BIOMD0000000491	$\approx 10^{24}$	14.922	11.217	**5.227**	13.420
BIOMD0000000638	$\approx 10^{27}$	17.172	**9.694**	10.364	10.415
BIOMD0000000501	$\approx 10^{25}$	94.272	168.461	**40.220**	85.869
BIOMD0000000103	$\approx 10^{18}$	182.722	c	205.993	**128.633**
BIOMD0000000014	$\approx 10^{67}$	t	m	m	m

Table A.2: Redlog run-times in seconds for completed runs that took at least 0.1 seconds, sorted by minimal run-times. 'Combos' is the number of combinations; bold numbers mark the lowest run-time; 't' signifies timeout.

Model	Combos	`rldnf`	`rldnf` `sm`	`rldnf` `sac`	`rldnf` `sac+sm`	`rlgsn`
`BIOMD0000000692`	19 200	**0.05**	0.11	0.06	0.06	0.06
`BIOMD0000000770`	1050	0.07	0.37	**0.06**	0.32	0.08
`BIOMD0000000871`	4608	0.11	5.82	0.08	0.78	**0.06**
`BIOMD0000000647`	30 720	**0.14**	7.11	0.35	3.66	0.35
`BIOMD0000000361`	20 736	1.00	148.84	**0.47**	4.63	0.49
`BIOMD0000000769`	31 360	**0.75**	52.36	1.27	26.03	1.31
`BIOMD0000000759`	88 000	**1.45**	3.04	1.49	2.26	1.56
`BIOMD0000000940`	73 728	**4.92**	1607.01	25.58	1325.86	25.15
`BIOMD0000000447`	$\approx 10^6$	**9.83**	t	70.15	t	73.78
`BIOMD0000000082`	$\approx 10^6$	55.42	221.58	**21.10**	40.98	21.25
`BIOMD0000000026`	$\approx 10^6$	90.67	t	63.85	1236.03	**63.05**
`BIOMD0000000163`	153 600	**127.87**	t	347.00	t	349.17

Table A.3: SMTcut vs. Redlog run-times in seconds for completed runs that took at least 0.1 seconds. 'Combos' is the number of combinations; bold numbers mark the lowest run-time; 'm' signifies memory exceeded.

Model	Combos	Redlog	SMTcut	Speed-up
`BIOMD0000000647`	30 720	0.14	**0.007**	20
`BIOMD0000000940`	73 728	4.92	**0.009**	547
`BIOMD0000000163`	153 600	127.87	**0.009**	14208
`BIOMD0000000759`	88 000	1.45	**0.010**	145
`BIOMD0000000447`	$\approx 10^6$	9.83	**0.011**	894
`BIOMD0000000361`	20 736	0.47	**0.012**	39
`BIOMD0000000082`	$\approx 10^6$	21.10	**0.013**	1623
`BIOMD0000000026`	$\approx 10^6$	63.05	**0.032**	1970
`BIOMD0000000769`	31 360	0.75	**0.033**	23
`BIOMD0000000028`	$\approx 10^9$	m	**0.147**	—
`BIOMD0000000637`	$\approx 10^{12}$	m	**0.216**	—
`BIOMD0000000599`	$\approx 10^{16}$	m	**0.355**	—
`BIOMD0000000365`	$\approx 10^{23}$	m	**0.490**	—
`BIOMD0000000147`	$\approx 10^{16}$	m	**0.733**	—
`BIOMD0000000492`	$\approx 10^{25}$	m	**2.247**	—
`BIOMD0000000102`	$\approx 10^{10}$	m	**2.690**	—
`BIOMD0000000407`	$\approx 10^{16}$	m	**3.606**	—
`BIOMD0000000431`	$\approx 10^{16}$	m	**4.107**	—
`BIOMD0000000491`	$\approx 10^{24}$	m	**5.227**	—
`BIOMD0000000638`	$\approx 10^{27}$	m	**9.694**	—
`BIOMD0000000501`	$\approx 10^{25}$	m	**40.220**	—
`BIOMD0000000103`	$\approx 10^{18}$	m	**128.633**	—

Table A.4: Run-times for SMTcut vs. run-times from Samal [92]. Run-times are in seconds.

Model	Samal [s]	Adjusted [s]	SMTcut [s]	Speed-up
BIOMD0000000233	3.139	1.433	**0.024**	131
BIOMD0000000125	3.975	1.815	**0.031**	128
BIOMD0000000156	3.973	1.814	**0.036**	110
BIOMD0000000159	3.977	1.816	**0.037**	107
BIOMD0000000289	6.794	3.102	**0.050**	136
BIOMD0000000040	5.538	2.529	**0.057**	97
BIOMD0000000150	9.307	4.250	**0.063**	148
BIOMD0000000460	3.963	1.810	**0.072**	55
BIOMD0000000194	13.568	6.195	**0.076**	179
BIOMD0000000459	4.538	2.072	**0.079**	57
BIOMD0000000072	13.874	6.335	**0.082**	169
BIOMD0000000077	16.259	7.424	**0.098**	166
BIOMD0000000199	11.979	5.470	**0.110**	109
BIOMD0000000101	11.698	5.342	**0.119**	98
BIOMD0000000198	14.720	6.721	**0.124**	119
BIOMD0000000035	20.258	9.250	**0.159**	127
BIOMD0000000361	29.244	13.353	**0.183**	160
BIOMD0000000193	41.923	19.143	**0.190**	221
BIOMD0000000046	75.383	34.421	**0.226**	334
BIOMD0000000080	36.428	16.634	**0.226**	161
BIOMD0000000082	31.983	14.604	**0.242**	132
BIOMD0000000287	36.821	16.813	**0.281**	131
BIOMD0000000226	49.464	22.586	**0.289**	171
BIOMD0000000038	573.775	261.998	**0.297**	1932
BIOMD0000000163	48.235	22.025	**0.305**	158
BIOMD0000000257	38.489	17.575	**0.335**	115
BIOMD0000000026	62.607	28.588	**0.348**	180
BIOMD0000000270	192.741	88.010	**0.438**	440
BIOMD0000000001	76.953	35.138	**0.486**	158
BIOMD0000000122	244.158	111.488	**0.718**	340
BIOMD0000000123	333.315	152.199	**0.751**	444
BIOMD0000000009	281.302	128.448	**0.770**	365
BIOMD0000000028	190.101	86.804	**1.023**	186
BIOMD0000000030	256.953	117.330	**1.423**	181
BIOMD0000000002	646.381	295.151	**2.856**	226
BIOMD0000000102	3833.583	1750.495	**3.703**	1035

Bibliography

[1] Herbert Amann. *Ordinary Differential Equations: An Introduction to Nonlinear Analysis.* de Gruyter, 1990.

[2] Alessandro Armando, Jacopo Mantovani, and Lorenzo Platania. Bounded model checking of software using SMT solvers instead of SAT solvers. In *International SPIN Workshop on Model Checking of Software*, pages 146–162. Springer, 2006.

[3] Dennis S. Arnon, George E. Collins, and Scott McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM J. Comput.*, 13(4):865–877, November 1984.

[4] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. CAV 2011*, volume 6806 of *LNCS*, pages 171–177. Springer, 2011.

[5] Clark Barrett, Leonardo De Moura, and Pascal Fontaine. Proofs in satisfiability modulo theories. In B. W. Paleo and D. Delahaye, editors, *All About Proofs, Proofs for All*, volume 55 of *Studies in logic*, pages 23–44. College Publications, 2015.

[6] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The SMT-LIB standard: Version 2.6. Technical report, Department of Computer Science, The University of Iowa, 2017.

[7] Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Model Checking*, pages 305–343. Springer, 2018.

[8] Thomas Becker, Volker Weispfenning, and Heinz Kredel. *Gröbner Bases, a Computational Approach to Commutative Algebra*, volume 141 of *Graduate Texts in Mathematics*. Springer, 1993.

[9] Paul Bernays. Beiträge zur axiomatischen Behandlung des Logik-Kalküls, 1918. Habilitationsschrift, unpublished.

[10] Paul Bernays. Axiomatische Untersuchung des Aussagen-Kalkuls der "Principia Mathematica". *Math. Z.*, pages 305–320, 1926.

[11] Nikolaj Bjørner, Kenneth L. McMillan, and Andrey Rybalchenko. Program verification as satisfiability modulo theories. In P. Fontaine and A. Goel, editors, *SMT 2012. 10th International Workshop on Satisfiability Modulo Theories*, volume 20 of *EPiC Series in Computing*, pages 3–11, 2013.

[12] François Boulier, François Fages, Ovidiu Radulescu, Satya S. Samal, Andreas Schuppert, Werner M. Seiler, Thomas Sturm, Sebastian Walcher, and Andreas Weber. The SYMBIONT project: Symbolic methods for biological networks. *ACM Commun. Comput. Algebra*, 52(3):67–70, December 2018.

[13] Russell Bradford, James H. Davenport, Matthew England, Hassan Errami, Vladimir Gerdt, Dima Grigoriev, Charles Hoyt, Marek Košta, Ovidiu Radulescu, Thomas Sturm, and Andreas Weber. Identifying the parametric occurrence of multiple steady states for some biological networks. *J. Symb. Comput.*, 98:84–119, May 2020.

[14] Robert K. Brayton, Gary D. Hachtel, Curt McMullen, and Alberto L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*, volume 2. Springer Science & Business Media, 1984.

[15] George E. Briggs and John B. S. Haldane. A note on the kinetics of enzyme action. *Biochem. J.*, 19(2):338, 1925.

[16] Erwan Brugallé and Kristin Shaw. A bit of tropical geometry. *CoRR*, abs/1311.2360, 2013.

[17] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* Doctoral dissertation, Mathematical Institute, University of Innsbruck, Innsbruck, Austria, 1965.

[18] Pedro T. Cardin and Marco A. Teixeira. Fenichel theory for multiple time scale singular perturbation problems. *SIAM J. Appl. Dyn. Syst.*, 16(3):1425–1452, January 2017.

[19] Alessandro Cimatti, Alberto Griggio, Bastiaan Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT solver. In N. Piterman and S. A. Smolka, editors, *Proc. TACAS 2013*, volume 7795 of *LNCS*, pages 93–107. Springer, 2013.

[20] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[21] Olivier Coudert. Doing two-level logic minimization 100 times faster. In *SODA*, pages 112–121, 1995.

[22] Haskell B. Curry and Robert Feys. *Combinatory Logic*, volume I of *Studies in Logic and the Foundations of Mathematics.* North Holland Publishing Company, Amsterdam, The Netherlands, 1958.

[23] Alicia Dickenstein. Biochemical reaction networks: An invitation for algebraic geometers. In *Mathematical Congress of the Americas*, volume 656 of *Contemp. Math.*, pages 65–83. 2016.

[24] Andreas Dolzmann and Thomas Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bull.*, 31(2):2–9, June 1997.

[25] Andreas Dolzmann and Thomas Sturm. Simplification of quantifier-free formulae over ordered fields. *J. Symb. Comput.*, 24(2):209–232, 1997.

[26] Andreas Dolzmann, Thomas Sturm, and Volker Weispfenning. Real quantifier elimination in practice. In B. H. Matzat, G.-M. Greuel, and G. Hiss, editors, *Algorithmic Algebra and Number Theory*, pages 221–247. Springer, 1998.

[27] Bruno Dutertre. Yices 2.2. In *Proc. CAV 2014*, volume 8559 of *LNCS*, pages 737–744. Springer, July 2014.

[28] Bruno Dutertre and Leonardo de Moura. A fast linear-arithmetic solver for DPLL(T). In *International Conference on Computer Aided Verification*, pages 81–94. Springer, 2006.

[29] Hassan Errami, Markus Eiswirth, Dima Grigoriev, Werner M. Seiler, Thomas Sturm, and Andreas Weber. Detection of Hopf bifurcations in chemical reaction networks using convex coordinates. *J. Comput. Phys.*, 291:279–302, 2015.

[30] Martin Feinberg. *Foundations of Chemical Reaction Network Theory*, volume 202 of *Applied Mathematical Sciences.* Springer, 2019.

[31] Neil Fenichel. Geometric singular perturbation theory for ordinary differential equations. *J. Differ. Equations*, 31(1):53–98, January 1979.

[32] Feliks R. Gantmacher. *The Theory of Matrices. Vol. 2. Transl. from the Russian by K. A. Hirsch.* Providence, RI: AMS Chelsea Publishing, reprint of the 1959 translation edition, 1998.

[33] Marco Gario and Andrea Micheli. PySMT: A solver-agnostic library for fast prototyping of SMT-based algorithms. In *SMT Workshop 2015. 13th International Workshop on Satisfiability Modulo Theories, Affiliated With the 27th International Conference on Computer Aided Verification*, San Francisco, CA, July 2015.

[34] Naama Geva-Zatorsky, Nitzan Rosenfeld, Shalev Itzkovitz, Ron Milo, Alex Sigal, Erez Dekel, Talia Yarnitzky, Yuvalal Liron, Paz Polak, Galit Lahav, and Uri Alon. Oscillations and variability in the p53 system. *Mol. Syst. Biol.*, 2(1):2006–0033, June 2006.

[35] Dima Grigoriev, Alexandru Iosif, Hamid Rahkooy, Thomas Sturm, and Andreas Weber. Efficiently and effectively recognizing toricity of steady state varieties. *Math. Comput. Sci.*, pages 1–34, 2020.

[36] Elizabeth Gross, Heather A. Harrington, Zvi Rosen, and Bernd Sturmfels. Algebraic systems biology: A case study for the Wnt pathway. *Bull. Math. Biol.*, 78(1):21–51, 2016.

[37] Emilie V. Haynsworth. On the Schur complement. Technical report, Basel Univ (Switzerland) Mathematics Inst, 1968.

[38] Anthony C. Hearn. Reduce—a user-oriented system for algebraic simplification. *ACM SIGSAM Bull.*, 1(6):50–51, May 1967.

[39] Frederick G. Heineken, Henry M. Tsuchiya, and Rutherford Aris. On the mathematical status of the pseudo-steady state hypothesis of biochemical kinetics. *Math. Biosci.*, 1(1):95–113, March 1967.

[40] A. Esteban Hernandez-Vargas and Michael Meyer-Hermann. Innate immune system dynamics to influenza virus. *IFAC Proceedings Volumes*, 45(18):260–265.

[41] Michael Hucka, Andrew Finney, Herbert M. Sauro, Hamid Bolouri, John C. Doyle, Hiroaki Kitano, Adam P. Arkin, Benjamin J. Bornstein, Dennis Bray, Athel Cornish-Bowden, Autumn A. Cuellar, Serge Dronov, Ernst D. Gilles, Martin Ginkel, Victoria Gor, Igor I. Goryanin, Warren J. Hedley, T. Charles Hodgman, Jan-Hendrik Hofmeyr, Peter J. Hunter, Nick S. Juty, J. L. Kasberger, Andreas Kremling, Ursula Kummer, Nicolas Le Novère, Leslie M. Loew, Daniel Lucio, Pedro Mendes, Eric Minch, Eric D. Mjolsness, Yasuaki Nakayama, Melanie R. Nelson, Poul F. Nielsen, Takeshi Sakurada, James C. Schaff, Bruce E. Shapiro, Thomas S. Shimizu, Hugh D. Spence, Joerg Stelling, Koichi Takahashi, Masayuki Tomita, John Wagner, and J. Wang. The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, March 2003.

[42] Adolf Hurwitz. Ueber die Bedingungen, unter welchen eine Gleichung nur Wurzeln mit negativen reellen Theilen besitzt. *Math. Ann.*, 46:273–284, June 1895.

[43] IEEE Std. 754-2019, July 2019. IEEE Standard for Floating-Point Arithmetic.

[44] Wenwen Ju and Chenqi Mou. Exploiting variable sparsity in computing equilibria of biological dynamical systems by triangular decomposition. In *International Conference on Algorithms for Computational Biology*, pages 29–41. Springer, 2021.

[45] Maurice Karnaugh. The map method for synthesis of combinational logic circuits. *Trans. Am. Inst. Electr. Eng. Part 1*, 72(5):593–599, 1953.

[46] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[47] Eric Katz. What is tropical geometry? *Not. Am. Math. Soc.*, 64(4), 2017.

[48] Donald E. Knuth. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1.* Addison-Wesley, 2011.

[49] Marek Košta. *New Concepts for Real Quantifier Elimination by Virtual Substitution.* Doctoral dissertation, Saarland University, Germany, December 2016.

[50] Marek Košta, Thomas Sturm, and Andreas Dolzmann. Better answers to real questions. *J. Symb. Comput.*, 74:255–275, May 2016.

[51] Niclas Kruff, Christoph Lüders, Ovidiu Radulescu, Thomas Sturm, and Sebastian Walcher. Algorithmic reduction of biological networks with multiple time scales. *Math. Comput. Sci.*, 15(3):499–534, September 2021.

[52] Niclas Kruff and Sebastian Walcher. Coordinate-independent singular perturbation reduction for systems with three time scales. *Math. Biosci. Eng.*, 16(5):5062–5091, June 2019.

[53] Aless Lasaruk and Thomas Sturm. Weak quantifier elimination for the full linear theory of the integers. A uniform generalization of Presburger arithmetic. *Appl. Algebra Eng. Commun. Comput.*, 18(6):545–574, December 2007.

[54] Michael T Laub and William F Loomis. A molecular network that produces spontaneous oscillations in excitable cells of dictyostelium. *Mol. biol. cell*, 9(12):3521–3532, 1998.

[55] Christian Lax and Sebastian Walcher. Singular perturbations and scaling. *CoRR*, abs/1807.03107, 2018.

[56] Hanl Lee and Angelyn Lao. Transmission dynamics and control strategies assessment of avian influenza A (H5N6) in the Philippines. *Infect. Dis. Model.*, 3:35–59, March 2018.

[57] Stefan Legewie, Nils Blüthgen, and Hanspeter Herzel. Mathematical modeling identifies inhibitors of apoptosis as mediators of positive feedback and bistability. *PLoS Comput. Biol.*, 2(9):e120, September 2006.

[58] Sanhong Liu, Shigui Ruan, and Xinan Zhang. Nonlinear dynamics of avian influenza epidemic models. *Math. Biosci.*, 283:118–135, January 2017.

[59] Christoph Lüders. PtCut: Calculate tropical equilibrations and prevarieties. `http://www.wrogn.com/ptcut/`.

[60] Christoph Lüders. Computing tropical prevarieties with satisfiability modulo theories (SMT) solvers. In P. Fontaine, K. Korovin, I. S. Kotsireas, P. Rümmer, and S. Tourret, editors, *Proc. SC-Square 2020, Co-Located With IJCAR 2020*, volume 2752 of *CEUR Workshop Proceedings*, pages 189–203. CEUR-WS, June–July 2020.

[61] Christoph Lüders, Thomas Sturm, and Ovidiu Radulescu. ODEbase: A repository of ODE systems for systems biology. *Bioinformatics Advances*, 2(1), April 2022. vbac027.

[62] Diane Maclagan and Bernd Sturmfels. *Introduction to Tropical Geometry*, volume 161 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2015.

[63] Filippo Maggioli, Toni Mancini, and Enrico Tronci. SBML2Modelica: Integrating biochemical models within open-standard simulation ecosystems. *Bioinformatics*, 36(7):2165–2172, May 2020.

[64] Sharad Malik and Lintao Zhang. Boolean satisfiability from theoretical hardness to practical success. *Commun. ACM*, 52(8):76–82, 2009.

[65] Rahuman S. Malik-Sheriff, Mihai Glont, Tung V. N. Nguyen, Krishna Tiwari, Matthew G. Roberts, Ashley Xavier, Manh T. Vu, Jinghao Men, Matthieu Maire, Sarubini Kananathan, Emma L. Fairbanks, Johannes P. Meyer, Chinmay Arankalle, Thawfeek M. Varusai, Vincent Knight-Schrijver, Lu Li, Corina Dueñas-Roca, Gaurhari Dass, Sarah M. Keating, Young M. Park, Nicola Buso, Nicolas Rodriguez, Michael Hucka, and Henning Hermjakob. BioModels—15 years of sharing computational models in life science. *Nucleic Acids Res.*, November 2019.

[66] Allan Marquand. Xxxiii. on logical diagrams for n terms. *Lond. Edinb. Dublin. Philos. Mag. J. Sci.*, 12(75):266–270, 1881.

[67] Edward J. McCluskey. Minimization of Boolean functions. *Bell Syst. Tech. J.*, 35(6):1417–1444, 1956.

[68] Patrick C. McGeer, Jagesh V. Sanghavi, Robert K. Brayton, and Alberto L. Sangiovanni-Vincentelli. ESPRESSO-SIGNATURE: A new exact minimizer for logic functions. *IEEE T. VLSI Syst.*, 1(4):432–440, 1993.

[69] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, Amit Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. SymPy: Symbolic computing in Python. *PeerJ Comput. Sci.*, 3:e103, January 2017.

[70] Leonor Michaelis and Maud L. Menten. Die Kinetik der Invertinwirkung. *Biochem. Z.*, 49:333–369, 1913.

[71] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In C. R. Ramakrishnan and J. Rehof, editors, *Proc. TACAS 2008*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[72] Leonardo de Moura and Nikolaj Bjørner. Satisfiability modulo theories: Introduction and applications. *Commun. ACM*, 54(9):69–77, 2011.

[73] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *J. ACM*, 53(6):937–977, November 2006.

[74] Kaspar Nipp. An algorithmic approach for solving singularly perturbed initial value problems. In U. Kirchgraber and H. Walther, editors, *Dynamics Reported*, volume 1, pages 173–263. John Wiley & Sons and B. G. Teubner, 1988.

[75] Vincent Noel, Dima Grigoriev, Sergei Vakulenko, and Ovidiu Radulescu. Tropical geometries and dynamics of biochemical networks application to hybrid cell cycle models. In J. Feret and A. Levchenko, editors, *Proc. SASB 2011*, volume 284 of *ENTCS*, pages 75–91. Elsevier, 2012.

[76] Vincent Noel, Dima Grigoriev, Sergei Vakulenko, and Ovidiu Radulescu. Tropicalization and tropical equilibration of chemical reactions. In G. L. Litvinov and S. N. Sergeev, editors, *Tropical and Idempotent Mathematics and Applications*, volume 616 of *Contemporary Mathematics*, pages 261–277. AMS, 2014.

[77] Nicolas Le Novère, Benjamin J. Bornstein, Alexander Broicher, Mélanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert M. Sauro, Maria Schilstra, Bruce E. Shapiro, Jacky L. Snoep, and Michael Hucka. BioModels database: A free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic acids res.*, 34(suppl_1):D689–D691, January 2006.

[78] Charles S. Peirce. On the algebra of logic. *Am. J. Math.*, 3:15–57, 1880.

[79] Mojzesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du Premier Congres de Mathematiciens des Pays Slaves*, pages 92–101, Warsaw, Poland, 1929.

[80] Willard V. Quine. The problem of simplifying truth functions. *Am. Math. Mon.*, 59(8):521–531, 1952.

[81] Ovidiu Radulescu, Alexander N. Gorban, Andrei Zinovyev, and Vincent Noel. Reduction of dynamical biochemical reactions networks in computational biology. *Front. Genet.*, 3:131, July 2012.

[82] Ovidiu Radulescu, Sergei Vakulenko, and Dima Grigoriev. Model reduction of biochemical reactions networks by tropical analysis methods. *Math. Model. Nat. Pheno.*, 10(3):124–138, June 2015.

[83] Hamid Rahkooy and Thomas Sturm. First-order tests for toricity. In *International Workshop on Computer Algebra in Scientific Computing*, pages 510–527. Springer, 2020.

[84] Sebastian Rautila. Evaluating the 2015 SMT competition, 2017.

[85] Dennis Reddyhoff, John Ward, Dominic Williams, Sophie Regan, and Steven Webb. Timescale analysis of a mathematical model of acetaminophen metabolism and toxicity. *J. Theor. Biol*, 386:132–146, December 2015.

[86] Tomás Revilla and Gisela García-Ramos. Fighting a virus with a virus: A dynamic model for HIV-1 therapy. *Math. Biosci.*, 185(2):191–203, 2003.

[87] John A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.

[88] Nicolas Rodriguez, Jean-Baptiste Pettit, Piero Dalle Pezze, Lu Li, Arnaud Henry, Martijn P. van Iersel, Gael Jalowicki, Martina Kutmon, Kedar N. Natarajan, David Tolnay, Melanie I. Stefan, Chris T. Evelo, and Nicolas Le Novère. The systems biology format converter. *BMC Bioinform.*, 17(1):1–7, 2016.

[89] Shigui Ruan. Modeling the transmission dynamics and control of rabies in China. *Math. Biosci.*, 286:65–93, April 2017.

[90] Richard L. Rudell. *Logic Synthesis for VLSI Design.* PhD thesis, University of California, Berkeley, 1989.

[91] Steve M. Ruggiero and Ashlee N. Ford Versypt. SBMLtoODEpy: A software program for converting SBML models into ODE models in Python. *J. Open Source Softw.*, 5(53):1643, 2019.

[92] Satya S. Samal. *Analysis of Biochemical Reaction Networks Using Tropical and Polyhedral Geometry Methods.* PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2016.

[93] Satya S. Samal, Dima Grigoriev, Holger Fröhlich, and Ovidiu Radulescu. Analysis of reaction network systems using tropical geometry. In V. P. Gerdt, W. Koepf, W. M. Seiler, and E. V. Vorozhtsov, editors, *Proc. CASC 2015*, volume 9301 of *LNCS*, pages 424–439. Springer, 2015.

[94] Satya S. Samal, Dima Grigoriev, Holger Fröhlich, Andreas Weber, and Ovidiu Radulescu. A geometric method for model reduction of biochemical networks with polynomial rate functions. *Bull. Math. Biol.*, 77(12):2180–2211, October 2015.

[95] Satya S. Samal, Aurélien Naldi, Dima Grigoriev, Andreas Weber, Nathalie Théret, and Ovidiu Radulescu. Geometric analysis of pathways dynamics: Application to versatility of TGF-β receptors. *Biosystems*, 149:3–14, 2016.

[96] Klaus R. Schneider and Thomas Wilhelm. Model reduction by extended quasi-steady-state approximation. *J. Math. Biol.*, 40(5):443–450, May 2000.

[97] Issai Schur. Über Potenzreihen, die im Innern des Einheitskreises beschränkt sind. *J. Reine Angew. Math.*, 1917.

[98] Stefan Schuster and Thomas Höfer. Determining all extreme semi-positive conservation relations in chemical reaction systems: A test criterion for conservativity. *J. Chem. Soc., Faraday Trans.*, 87(16):2561–2566, 1991.

[99] Bruce E. Shapiro, Michael Hucka, Andrew Finney, and John C. Doyle. MathSBML: A package for manipulating SBML-based biological models. *Bioinformatics*, 20(16):2829–2831, 2004.

[100] Jake Spurlock. *Bootstrap: Responsive Web Development*. O'Reilly Media, Inc., 2013.

[101] Thomas Sturm. Linear problems in valued fields. *J. Symb. Comput.*, 30(2):207–219, August 2000.

[102] Andrei N. Tikhonov. Systems of differential equations containing small parameters in the derivatives. *Mat. Sb. (N. S.)*, 73(3):575–586, 1952.

[103] Edward W. Veitch. A chart method for simplifying truth functions. In *Proceedings of the 1952 ACM National Meeting (Pittsburgh)*, pages 127–133, 1952.

[104] Ferdinand Verhulst. *Methods and Applications of Singular Perturbations: Boundary Layers and Multiple Timescale Dynamics*, volume 50. Springer Science & Business Media, 2005.

[105] Jose M. G. Vilar, Ronald Jansen, and Chris Sander. Signal processing in the TGF-β superfamily ligand-receptor network. *PLoS Comput. Biol.*, 2(1):e3, January 2006.

[106] Eberhard O. Voit, Harald A. Martens, and Stig W. Omholt. 150 years of the mass action law. *PLoS Comput. Biol.*, 11(1), 2015.

[107] Volker Weispfenning. Quantifier elimination for real algebra—the quadratic case and beyond. *Appl. Algebr. Eng. Comm.*, 8(2):85–101, February 1997.

[108] Dominik Wodarz and Dean H. Hamer. Infection dynamics in HIV-specific CD4 T cells: Does a CD4 T cell boost benefit the host or the virus? *Math. Biosci.*, 209(1):14–29, September 2007.

[109] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Mag.*, 17(3):73–73, 1996.