Simulation and optimization of logical and kinetic biochemical models

Simulation and optimization of logical and kinetic biochemical models

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Dipl.-Inform. Roland Keller

aus Speyer

Tübingen

2016

Tag der mündlichen Qualifikation:	11.05.2016
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. rer. nat. Andreas Zell
2. Berichterstatter:	Prof. Dr.-Ing. Oliver Kohlbacher

Abstract

During the last years the impact of systems biology has grown drastically. In contrast to traditional biology, this interdisciplinary field comprises the investigation of biological processes from a systems perspective. An example for a systems biology project is the Virtual Liver Network, in which metabolic liver function is modeled computationally. A mathematical model in systems biology provides a hypothesis that is testable by biological experiments. Data obtained from model simulation can thereby be compared to experimental data possibly leading to the adaptation of the model. Experimental data is also used to optimize a model, e.g., to estimate certain model parameters or to identify connections between model components.

From the different kinds of computational models used in systems biology logical models and kinetic ordinary differential equation (ODE) models are covered in this thesis. While logical models enable to describe biological processes qualitatively, kinetic ODE models allow the dynamic description of these processes. Methods for the simulation and optimization of both model types were developed and applied here.

The first part of the thesis contains the application of a specific logical modeling technique called fuzzy logic modeling. A previously published method based on prior knowledge and experimental data was adapted to identify regulatory events responsible for the downregulation of drug metabolism during inflammation. Further experiments backed the hypothesis suggested by the model. The respective study conducted in collaboration with biologists is a relevant part of the Virtual Liver Network.

In the following part of the thesis an algorithm for the simulation of models given in the Systems Biology Markup Language (SBML) is described. SBML is the most important standard for storing and exchanging systems biology models. It enables to describe ODE models that can also contain other elements, such as rules for model components and events representing sudden changes of components. Because of these additional elements, simulation of SBML models is difficult and only few software tools support this standard completely. The Systems Biology Simulation Core Library (SBSCL), which contains an implementation of the developed algorithm, supports SBML completely. Benchmark tests were used to prove the correctness of the library. The SBSCL can be easily integrated into larger software tools. One example for this is SBMLsimulator, which connects the SBSCL and the optimization toolbox EvA2 and is described in the following chapter of this thesis. SBMLsimulator enables simulation and optimization of SBML models and can be very helpful in systems biology studies.

At the end of the thesis SBMLsimulator is applied in a large study investigating the influence of experimental noise on the estimation of kinetic parameters for three published

SBML models. The study shows the usefulness of the tool, but also suggests to take the noise into account during estimation, which was previously only tested for a small toy model. Furthermore, the chosen approach is a way of robustness analysis that can be used to estimate how reliable the parameter estimates for a certain model are. Like the other methods applied and developed in this thesis, it can be used for further systems biology research.

Kurzfassung

In den letzten Jahren hat die Systembiologie drastisch an Bedeutung gewonnen. Im Unterschied zur traditionellen Biologie werden biologische Prozesse in diesem interdisziplinären Feld aus einer Systemperspektive heraus untersucht. Das Netzwerk Virtuelle Leber, in dem die Funktionen der Leber am Computer modelliert werden, ist ein Beispiel für ein großes systembiologisches Projekt. Ein solches mathematisches Modell in der Systembiologie enthält eine Hypothese, die mit biologischen Experimenten überprüft werden kann. Die aus einer Modellsimulation erhaltenen Daten können dabei mit experimentellen Daten verglichen werden, was möglicherweise zu einer Anpassung des Modells führt. Experimentelle Daten werden auch zur Optimierung eines Modells verwendet, was die Schätzung bestimmter Modellparameter oder die Bestimmung von Verbindungen zwischen Modellkomponenten umfassen kann.

Von den verschiedenen für die Systembiologie geeigneten Modelltypen wurden in dieser Arbeit logische Modelle und kinetische Differenzialgleichungsmodelle (ODE-Modelle) benutzt. Während mit logischen Modellen biologische Prozesse qualitativ beschrieben werden können, ermöglichen kinetische ODE-Modelle die dynamische Beschreibung solcher Prozesse. Methoden für die Simulation und Optimierung beider Modelltypen wurden im Rahmen dieser Arbeit entwickelt und angewendet.

Der erste Teil dieser Arbeit enthält die Anwendung einer speziellen Art der logischen Modellierung, die als Fuzzy-Logik-Modellierung bezeichnet wird. Eine bereits publizierte Methode, die auf Vorwissen und experimentellen Daten basiert, wurde hier angepasst. Damit konnten dann regulatorische Ereignisse, die den Wirkstoffmetabolismus herunterregulieren, identifiziert werden. Weitere Experimente stützten die im Modell enthaltene Hypothese. Die Arbeit, die in Kooperation mit Biologen durchgeführt wurde, war Teil des Netzwerks Virtuelle Leber.

Im folgenden Teil der Arbeit wird ein Algorithmus zur Simulation von Modellen, die in der Beschreibungssprache SBML gegeben sind, beschrieben. SBML ist der wichtigste Standard zur Speicherung und zum Austasch systembiologischer Modelle. Die Sprache ermöglicht die Beschreibung von ODE-Modellen, die zusätzliche Elemente wie Regeln für Modellkomponenten und Events zur Repräsentation plötzlicher Veränderung solcher Komponenten enthalten können. Aufgrund dieser zusätzlichen Elemente ist die Simulation von SBML-Modellen schwierig und nur wenige Programme unterstützen den Standard vollständig. Die *Systems Biology Simulation Core Library* (SBSCL) enthält eine Implementierung des entwickelten Algorithmus und unterstützt alle SBML-Elemente. Mit Benchmark-Tests wurde die Korrektheit der Bibliothek bewiesen. Die SBSCL kann leicht in größere Programme integriert werden. Der SBMLsimulator, der die SBSCL

mit der Optimierungsbibliothek EvA2 verbindet, ist ein Beispiel dafür, das im folgenden Kapitel der Arbeit beschrieben wird. Er ermöglicht die Simulation und Optimierung von SBML-Modellen und kann damit in der systembiologischen Forschung sehr hilfreich sein.

Am Schluss der Arbeit wird der SBMLsimulator in einer größeren Studie angewendet, um den Einfluss von experimentellem Rauschen auf die Schätzung kinetischer Parameter in drei SBML-Modellen zu untersuchen. Zusätzlich zur Nützlichkeit des Programms zeigt die Studie, dass es sinnvoll ist, das experimentelle Rauschen bei der Parameterschätzung einzubeziehen. Dies wurde vorher nur für ein kleines Testmodell überprüft. Außerdem ist der verwendete Ansatz eine Art Robustheitsanalyse, mit der die Zuverlässigkeit einer Parameterschätzung für ein bestimmtes Modell untersucht werden kann. Wie die anderen Methoden, die in dieser Arbeit angewendet und entwickelt wurden, kann der Ansatz für viele weitere systembiologische Untersuchungen verwendet werden.

Danksagung

Zunächst möchte ich meiner Familie und meinen Freunden danken, die mich während meiner Zeit als Doktorand begleitet haben. Außerdem danke ich Dr. Elke Menkel-Conen, die mich auf die Idee gebracht hat Bioinformatik zu studieren.

Mein Dank gilt Prof. Dr. Andreas Zell für die Möglichkeit, in seiner Arbeitsgruppe zu promovieren, und für seine Unterstützung. Weiterhin danke ich dem Zweitgutachter der Arbeit Prof. Dr. Oliver Kohlbacher. Allen Kollegen im Lehrstuhl Kognitive Systeme danke ich für die gute Atmosphäre und für den Zusammenhalt, insbesondere Dr. Andreas Dräger, Alexander Dörr und Stephanie Hoffmann für die gute Zusammenarbeit. Mein besonderer Dank geht an unsere Sekretärin Vita Serbakova und unseren Rechneradministrator Klaus Beyreuther, die mir bei allen Problemen behilflich waren. Auch den Bachelorstudenten Robin Fähnrich, Stefan Fischer und Matthias Rall danke ich.

Ohne die Zusammenarbeit mit externen Kooperationspartnern wäre diese Arbeit nicht möglich gewesen. Hier danke ich insbesondere Dr. Marcus Klein, Dr. Maria Thomas und Prof. Dr. Uli Zanger vom IKP Stuttgart, aber auch Dr. Ute Metzger, Dr. Markus Templin und Dr. Thomas Joos vom NMI Reutlingen. Außerdem gilt mein spezieller Dank Prof. Dr. Peter Wills, mit dem ich bei der in dieser Arbeit enthaltenen Parameterschätzungsstudie zusammengearbeitet habe.

Zuletzt geht mein Dank an das Bundesministerium für Bildung und Forschung (BMBF), da diese Doktorarbeit Teil des vom BMBF geförderten Netzwerks Virtuelle Leber (Fördernummern 0315755 und 0315756) war.

Roland Keller
Tübingen, April 2015

Contents

Chapter 1

Introduction

During the last decades researchers have increasingly tried to understand biological processes from a systems perspective. The emerging field of systems biology does not primarily focus on single biochemical reactions or interactions, but on the complex interplay between different biological components like genes or proteins. Several biological components interacting or reacting with each other constitute a biological network, which can, e.g., process biological signals or transform toxic substances. A key question in systems biology is how certain biological networks functioning properly in healthy humans are modified during a disease. Ultimately, such research can lead to new strategies for disease treatment.

New experimental techniques that enable to measure many protein concentrations or gene expression values in parallel have greatly enhanced the possibilities of biological research. However, the resulting high-throughput data are usually complex and difficult to interpret. Computational methods are therefore required to extract important information from these data. In systems biology mathematical models that can simulate biological network function are developed based on such data. Each mathematical model provides a hypothesis for biological behavior that can be validated with adequate experimental data. Models are usually developed in a bottom-up fashion, which means that different model components are connected based on existing scientific knowledge from several sources (literature or databases). Afterwards, the model is optimized with respect to the available data. Dependent on the type of model, this optimization can involve changing the interactions between the model components or the estimation of certain model parameters. For large networks, in which many interactions are unclear, logical modeling is often applied. A logical model qualitatively describes how different model components interact with each other. In contrast to that, kinetic ordinary differential equation (ODE) models are used to investigate dynamic behavior of biological systems over time.

Standardized model formats have been developed to facilitate exchange and reuse of models in systems biology. The most important format in this respect is the Systems Biology Markup Language (SBML). SBML models can contain several model elements, which makes full support of this format difficult to implement. Therefore, many systems biology software solutions do not support the SBML standard completely.

The work covered in this thesis involves the development and application of meth-

ods for the simulation and optimization of systems biology models. First an existing method for the optimization of a logical model was adapted in order to identify important regulatory events modifying detoxification in the human liver. Next an algorithm for the simulation of SBML models that supports all SBML elements was developed and implemented in the Systems Biology Simulation Core Library. This simulation library was then connected with the optimization toolbox EvA2 in the program SBMLsimulator, which enables optimization of SBML models. Finally, SBMLsimulator was applied in a study to investigate the robustness of three published models against different magnitudes of experimental noise.

1.1 Contributions of the thesis

In order to obtain valid models in systems biology, routines for the simulation and optimization of such models are necessary. This thesis focuses on the development and application of such methods considering logical models as well as kinetic ODE models. While for logical models an existing optimization method was adapted to solve a specific biological question, for ODE models described by the SBML format a simulation algorithm was developed, connected with optimization routines, and applied in a study investigating the robustness of models against experimental noise. The work was part of the Virtual Liver Network (Holzhütter *et al.*, 2012), in which the main functions of the human liver are investigated (grant numbers 0315755 and 0315756). Next the different contributions of the thesis are described in more detail.

Application of fuzzy logic modeling for identification of regulatory events responsible for an altered hepatic gene expression upon IL-6

Fuzzy logic modeling is a specific type of logical modeling. In contrast to the often applied Boolean models, which only allow states of 0 or 1 for model components, fuzzy logic models enable continuous states for these components. We adapted a method to develop an optimized fuzzy logic model based on scientific knowledge and experimental data (Morris *et al.*, 2011), which is implemented in the software CNORfuzzy (Terfve *et al.*, 2012). With this adapted routine we constructed a fuzzy logic model describing downregulation of hepatic genes responsible for drug transformation during inflammation in the body (Keller *et al.*, 2016). Such a downregulation might cause drug overdosing, which is why a better understanding of the regulatory events behind this downregulation is of clinical relevance. The model suggested that one regulatory event is mainly causing the downregulation of genes. This hypothesis was supported by further experiments.

Development of an algorithm for the simulation of SBML models and its implementation

The most important format for biological models is SBML, which is mainly used to describe kinetic ODE models and in which model components (referred to as species) are connected via reactions. Those reactions can contain kinetic rate laws, which describe the dynamic behavior of the model. However, SBML models are also allowed to comprise other elements influencing model dynamics, e.g., events leading to sudden changes of model components. Those other elements make simulation of SBML models a very complicated task, which is why most simulation tools do not support all SBML elements. Therefore, we developed the Systems Biology Simulation Core Algorithm for simulation of SBML models with full support of all elements (Keller *et al.*, 2013). This algorithm was implemented in the Systems Biology Simulation Core Library (SBSCL), which can be easily integrated into other programs. Benchmark tests showed the correctness of the algorithm.

Graphical simulation and optimization of SBML models with SBMLsimulator

Kinetic parameters of ODE models are often estimated with respect to biological data. This estimation involves repeated model simulation with different parameter values. In order to enable such an optimization as well as a graphical display of the simulation results obtained with the SBSCL, SBMLsimulator (Dörr *et al.*, 2014) was developed. SBMLsimulator comprises the SBSCL for simulation of SBML models and the optimization toolbox EvA2 for parameter estimation. In a small experiment with a published model, the capability of SBMLsimulator to identify biochemical parameters was demonstrated.

Applying SBMLsimulator to three published models and testing their robustness to experimental noise in a parameter estimation study

The quality of experimental data has a great impact on the estimation of biochemical parameters in dynamic models. How well such parameters can be identified depends on the model structure and the quality of the experimental data such as the measured time points. Those data are usually very noisy, which is why we used SBMLsimulator in a simulation study to investigate how robust three published models with fixed measured time points are to different magnitudes of experimental noise. For all three models a mean noise of 20% led to several parameters not being reliably identifiable. By taking the noise in the data into account during parameter estimation, which was previously tested for a small toy model (Raue *et al.*, 2013), we could significantly improve the estimation results.

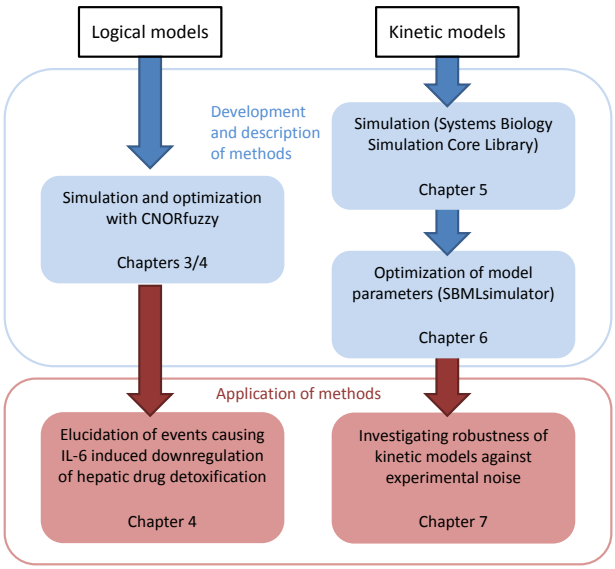

Figure 1.1: Structure of this thesis. The thesis contains chapters describing the development and implementation of methods for simulation and optimization of models as well as chapters covering the application of such methods. CNORfuzzy, which was developed elsewhere and only slightly adapted for this thesis, is integrated into this structure for better clarity, as it was applied in Chapter 4.

1.2 Thesis organization

The thesis is subdivided into eight chapters. In Chapter 2 the fundamentals of logical and kinetic modeling, such as model formats and simulation of a model are explained. Chapter 3 shows important steps during the construction of a model especially focusing on methods for model optimization. The main results of this thesis are presented in detail in Chapters 4 to 7. Besides the presentation of the applied methods and the results, each of these chapters comprises a motivation of the topic as well as a discussion, a summary, and a conclusion section. In Chapter 4 the elucidation of regulatory events responsible for downregulation of hepatic drug detoxification upon IL-6 is described. Chapter 5 contains the explanation of the algorithm for the simulation of SBML models and its implementation in the Systems Biology Simulation Core Library. This library was connected to the optimization toolbox EvA2 in the tool SBMLsimulator, which is covered in Chapter 6. The application of SBMLsimulator in a parameter estimation study investigating the robustness of models against experimental noise is presented in Chapter 7. Chapter 8 concludes this thesis with a summarization of the main results and a discussion. Figure 1.1 shows the structure of the thesis. Chapters 4 to 7 can be subdivided into chapters describing methods and their implementation (Chapters 5 and 6) and chapters covering application of methods (Chapters 4 and 7).

Chapter 2

Computational modeling approaches in systems biology

Understanding complex biological networks requires integrated experimental and computational research (Kitano, 2002). Computational models in systems biology can help here by providing hypotheses for the explanation of biological behavior. Models predict a certain behavior of a network, which can be compared to experimental data.

The biological networks modeled computationally often belong to one of the following three types:

- cell signaling networks (Janes and Lauffenburger, 2013) comprising the processing of certain signals, such as stimulation of the cell with a certain substance,
- metabolic networks (e.g., the citric acid cycle), which contain metabolites (i.e., intermediate products) connected via metabolic reactions and are necessary for cell function (Kaplan *et al.*, 2009), and
- gene regulatory networks, which contain genes and regulatory proteins interacting with each other (Hecker *et al.*, 2009).

If the prediction of a model differs substantially from experimental data, adaptations of the model are necessary. Apart from explaining experimental data, models can also suggest further experiments in order to refine a certain hypothesis. This might, e.g., involve a more detailed investigation of a certain dynamic process. In systems biology an iterative cycle of model refinements and new experiments is common, during which the understanding of the studied system is continuously improved.

Depending on the experimental data available and the biological network which is investigated, different types of computational models are developed. Two of the most frequently used model types are logical models and ordinary differential equation (ODE) models. This chapter gives an introduction on both modeling approaches, model simulation, and the modeling formats used for both approaches.

2.1 Logical models

Logical models can solve qualitative questions like whether stimulation of a specific signaling cascade activates a certain transcription factor (Samaga and Klamt, 2013). In such models logical species representing biological entities (e.g., proteins) are connected via logic gates. A gate is associated with a combination of the logic functions AND, OR, and NOT (Morris *et al.*, 2010). Thus a gate describes how the state of a node depends on the values of its input nodes.

Boolean models are the most widely used logical models. In them the species can only have two values (also called states): 0 (meaning inactive) and 1 (active). An AND function in a Boolean model returns 1, if all input nodes are active. For an OR function to be 1, it suffices that one of the input nodes is 1. A NOT function has only one incoming node and returns 0, if the input node is 1, and vice versa.

Boolean models are a big simplification of biological reality, but have in several cases enabled to improve understanding of large signaling networks (Samaga *et al.*, 2009; Saez-Rodriguez *et al.*, 2009; Ryll *et al.*, 2011; Saez-Rodriguez *et al.*, 2011).

2.1.1 Logical steady state

A typical question in logical modeling is the determination of the states of all the species in a logical networks given that some species are stimulated or inactivated. This stimulation or inhibition often involves the nodes in the network which are not activated or inhibited by other nodes (i.e, the so-called source nodes). In order to compute the states of the other nodes, the determination of a so-called logical steady state introduced previously (Klamt *et al.*, 2006) is a straightforward method.

The first step of the algorithm is the fixing of the values of activated species to 1 and those of inhibited species to 0. Then in each following step the states of nodes that are activated/inhibited by the processed species are computed. If the state of one node has just been determined for the first time or changed due to a new computation, the states of nodes activated/inhibited by this node are recomputed. The algorithm proceeds until the recomputations of all states do not cause changes any more.

If the states of all source nodes of the network are fixed and the network does not contain feedbacks, the logical steady state can be determined uniquely. An example for the computation of a logical steady state is shown for a toy network in Figure 2.1.

2.1.2 The SIF format

One important format for working with logical models is SIF (simple interaction file). It is one of the main formats of the important software Cytoscape for visualization of biological networks (Shannon *et al.*, 2003).

The SIF format enables to set a relationship between two nodes of a network. Each line of a SIF file contains such a relationship between two nodes, in which the first node

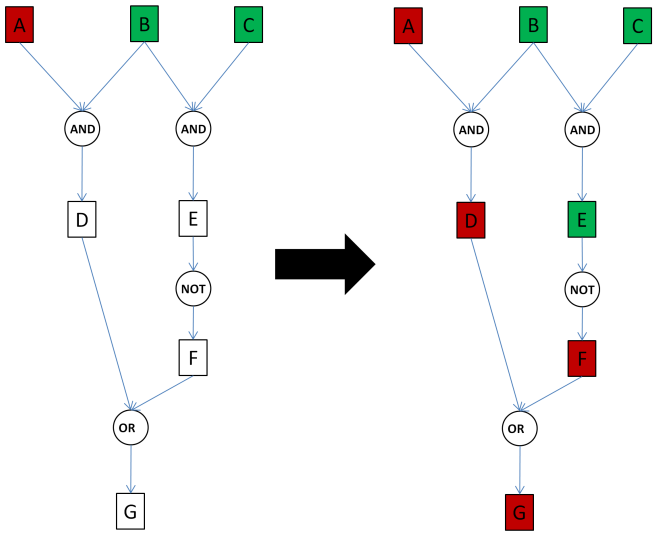

Figure 2.1: Example for the determination of the logical steady state of a toy network. The network contains species (rectangular nodes) as well as AND, OR and NOT gates (circular nodes). We now assume that the value of node A is 0 (red color) and the values of B and C are 1 (green color). Based on those settings the logical steady state can be computed in a step-wise fashion (right part of the figure). Beginning with the fixing of the source nodes, in each following step the states of the successors of the currently processed nodes are determined.

influences the second node. In this thesis we use "activates" (1) and "inactivates" (-1) as relationships. If one node is activated/inhibited by several other nodes, we assume that these other nodes are connected by the OR function. In order to define AND connections, we introduce a so-called dummy node before the influenced node. This shows that the SIF format is suitable for defining Boolean networks.

For the network in Figure 2.1 the SIF definition is as follows:

A	1	AND1
B	1	AND1
B	1	AND2
C	1	AND2
AND1	1	D
AND2	1	E
E	-1	F
D	1	G
F	1	G

`AND1` and `AND2` are the dummy nodes for defining the AND connections. The input nodes of D (A and B) as well as the input nodes of E (B and C) are thus by definition AND connected In contrast, the input nodes of G (D and F) are OR connected.

2.2 ODE models in systems biology

With ordinary differential equations (ODEs) one can describe how the derivatives of biological network variables with respect to time can be calculated based on the values of the variables. The ODEs of all variables are also referred to as an ODE system. The variables in an ODE model in systems biology are most often species representing, e.g., proteins or biological metabolites. Those species are connected by reactions, whose velocities are described by rate laws.

The ODE referring to a species can be determined from the velocities of all reactions the species takes part in, which is explained below. If all ODEs as well as the initial values of the variables are known, numerical integration routines can solve the ODE system. This means that the values of all variables are determined over time leading to, e.g., concentration curves of proteins.

2.2.1 The stoichiometric matrix

The stoichiometry of a species in a reaction is its relative turnover rate during the reaction. In the reaction $2\mathrm{A} + \mathrm{B} \rightarrow 2\mathrm{C}$ A and C have the stoichiometry 2, whereas B has stoichiometry 1. The stoichiometric matrix $\mathbf{N}$ of a model comprises for all reactions $\vec{R}$ (in columns) of a biological system the stoichiometries of the species $\vec{S}$ (in rows). If a species is not affected by some reaction, the corresponding stoichiometry in the matrix is 0.

Given the ODEs $\vec{v}$ for all reactions and the stoichiometric matrix, one can derive the ODEs for all species $\vec{S}$:

$$\frac{\mathrm{d}\vec{S}}{\mathrm{d}t} = \mathbf{N}\vec{v} \tag{2.1}$$

This means that for a specific species S_k the ODE is determined as follows:

$$\frac{\mathrm{d}S_k}{\mathrm{d}t} = \sum_{i=1}^{|\vec{R}|} (n_{i,k} \cdot v_i) \tag{2.2}$$

2.2.2 Rate laws

A rate law is an ODE that describes how to determine the velocity of a reaction based on species values and certain kinetic parameters. Based on the type of reaction, different types of rate laws are used:

- Generalized mass-action kinetics are derived from the mass action law (Heinrich and Schuster, 1996) and can be used to describe the kinetics of different types of reactions.

- Michaelis-Menten can describe reactions catalyzed by enzymes (Johnson and Goody, 2011), for which mass-action kinetics are often not appropriate.
- Convenience kinetics are a generalization of Michaelis-Menten kinetics especially dedicated to enzyme-catalyzed reactions involving several substrates and products (Liebermeister and Klipp, 2006).

2.2.3 Model simulation

We now assume that the ODEs $\frac{\mathrm{d}x(i)}{\mathrm{d}t} = f_i(x(1),\ldots,x(n))$ of all variables $x(i)$ of a biological system as well as the initial values of the variables are given. Then one key problem in ODE modeling is the determination of the values of all variables over time. The simulation of such an ODE system solves this problem. Most ODE systems cannot be simulated exactly, but several so-called integration routines (also referred to as solvers) exist for approximating the values of the variables precisely.

A straightforward way to solve an ODE system is to continuously increase the value of time by a certain step size h starting with the initial time. At each time point slopes can then be determined based on the current values of the variables, starting with the initial values at time point 0. The values of the variables $x_{\mathrm{pred}}(i,t_{k+1})$ at the following time point t_{k+1} are computed with the current values of the variables $x_{\mathrm{pred}}(i,t_k)$ and the slopes $f_i(x_{\mathrm{pred}}(1,t_k),\ldots,x_{\mathrm{pred}}(n,t_k))$:

$$x_{\mathrm{pred}}(i,t_{k+1}) = x_{\mathrm{pred}}(i,t_k) + h \cdot f_i(x_{\mathrm{pred}}(1,t_k),\ldots,x_{\mathrm{pred}}(n,t_k)) \tag{2.3}$$

This algorithm is called Euler method (Press *et al.*, 1993).

However, the Euler method with its constant step size h is not suitable for many kinds of ODE systems. There are stiff ODE systems, for which an extremely small step size is necessary within some (usually very short) periods of time in order to solve those systems with a tolerable error. As an extremely small step size throughout the whole simulation process would lead to an unacceptably high running time, integration methods need to be able to adapt their step size for such time periods. Therefore, applying the Euler method with a very small h to those ODE systems is not indicated. One method with an adaptive step size that is able to solve stiff ODE systems is Rosenbrock's method (Press *et al.*, 1993). It is used in this thesis. We leave out the complex details of Rosenbrock's method here, as they are not necessary for understanding the thesis.

2.2.4 The model format SBML

SBML (Hucka *et al.*, 2004) is the most important format for describing and storing ODE models in systems biology. There are several levels and versions of the SBML specification, level 3 version 1 being the latest of them (Hucka *et al.*, 2010). SBML is derived from XML and also comprises parts of MathML 2.0 (Carlisle *et al.*, 2001). Like XML

elements, the elements of SBML are hierarchically structured. SBML comprises several elements for the description of models in systems biology:

- species representing proteins or metabolites,
- reactions between those species,
- kinetic laws for these reactions allowing to declare the rate laws (in MathML),
- compartments (e.g., the cell), in which reactions can take place,
- parameters for defininining variables in the system other than species,
- function definitions for defining functions that can be referred to in mathematical expressions, such as, e.g., kinetic laws,
- initial assignments for defining how to determine the initial values of variables,
- assignment rules for defining how the values of variables are determined based on other variables during the whole simulation time,
- rate rules for stating how the derivatives of certain variables (e.g., species not taking part in any reaction) are computed based on other variables,
- algebraic rules for expressing relationships between variables that have to be valid during the whole simulation process, and
- events for stating sudden changes of variables of the biological system.

While SBML is still most often used for describing ODE models, during the last years several extensions have been specified that enable to use SBML for other purposes. How a model is supposed to be drawn can, e.g., be specified with the SBML layout extension (Gauges *et al.*, 2006). The SBMLqual extension (Chaouiya *et al.*, 2013b) facilitates the definition of qualitative models, which also involve logical models.

2.2.5 JSBML

SBML models need to be read in, modified and stored by systems biology software. To this end, libraries were created that provide an application programming interface (API) which facilitates the processing of SBML models. Those libraries can easily be integrated into software supporting SBML.

The oldest library that completely supports all levels and versions of SBML is libSBML (Bornstein *et al.*, 2008). libSBML is written in C and C++, but there are also wrappers for including libSBML into software tools written in other programming languages, such as Java.

Using libSBML in software implemented in Java does not lead to fully platform independent tools due to the internal C/C++ code. Furthermore, it is difficult to implement Java Web Start applications with libSBML. Therefore, JSBML was developed, which is completely written in Java (Dräger *et al.*, 2011; Rodriguez *et al.*, 2015). Its type hierarchy closely resembles the hierarchy of SBML elements. JSBML contains a parser for MathML, which transforms a formula into an abstract syntax tree that can be processed, e.g., by simulation software.

2.3 Comparison of the two modeling approaches

ODE models explain dynamic mechanisms behind a behaviour of a biological system, whereas logical models usually only describe the order, in which a certain process (e.g., a signal transduction through a certain pathway) proceeds. In order to construct an ODE model, knowledge about some reactions and their rate laws is usually a pre-requisite.

Logical models are often adequate for the investigation of interactions between many proteins (Morris *et al.*, 2010). Prior knowledge about mechanistic details is not necessary for constructing such models. Therefore, logical models are usually much larger than ODE models.

Chapter 3

The process of model construction

The creation of a logical model or an ODE model proceeds in several steps. This involves obtaining information about model structure at first. At the end of the construction process the model is often optimized with respect to biological data. To this end, heuristic optimization routines can be applied. This chapter first gives an introduction about those heuristic optimization routines. Then the model construction process is described for logical models and ODE models. The most important repository for SBML models, the BioModels database, is also introduced as well as the *path2models* project, which involved generation of numerous pathway models in SBML.

3.1 Heuristic optimization methods for model optimization

Many optimization problems involve a large solution space, which greatly hampers the search for an optimal solution in acceptable running time. Therefore, heuristic optimization methods are used that cannot guarantee to find the best global solution, but are able to find a good solution of the problem within a tolerable amount of time. Many of these algorithms involve random steps and are therefore stochastic. Several stochastic optimization methods are derived from natural processes. Important examples are ant colony optimization (Dorigo *et al.*, 2006), which is based on the behaviour of ant colonies, and particle swarm optimization (Kennedy and Eberhart, 1995; Clerc and Kennedy, 2002; Clerc, 2005), which is inspired by movements of birds in a flock. Some nature-inspired optimization algorithms have proven to be suitable for solving complex optimization problems in systems biology (Dräger *et al.*, 2009). In this thesis differential evolution, which is a special type of evolutionary algorithm, is used for estimation of model parameters. Therefore, evolutionary algorithms in general and differential evolution are explained in the following sections.

3.1.1 Evolutionary algorithms

Evolutionary algorithms are based on the principles of biological evolution (Freitas, 2007). At the beginning of an evolutionary algorithm, a population of candidate solutions is randomly initiated. Then the individual solutions of the population undergo recombination, mutation, and selection in each so-called generation of the algorithm. The quality of a solution is determined by a fitness function, whose value the algorithm tries to improve over time. The algorithm terminates, e.g., after a certain number of fitness evaluations or when the algorithm has converged to some fitness value.

Depending on the type of evolutionary algorithm, recombination, mutation, selection, and the representation of the individual solutions are defined differently. For the widely used genetic algorithm (Holland, 1975) the solution is represented by a vector of numbers of fixed size. The definition of recombination and mutation is straightforward then: A recombination between two candidate solutions is defined as a random exchange of a part of the two respective vectors similar to chromosomal crossover during meiosis. A mutation is a random change of a number in the vector of a candidate solution.

3.1.2 The toolbox EvA2

EvA2 (Kronfeld *et al.*, 2010; Becker and Kronfeld, 2014) is a Java framework that comprises a large number of evolutionary algorithms as well as other heuristic optimization routines. It has been successfully applied to optimization of biological systems in several studies (Dräger *et al.*, 2007b,a, 2009; Kronfeld *et al.*, 2009; Raue *et al.*, 2013).

The user can choose between a large number of algorithms, such as

- differential evolution (Storn, 1996; Storn and Price, 1997),
- hill climbing (Tovey, 1985),
- simulated annealing (Kirkpatrick *et al.*, 1983),
- evolution strategies (Rechenberg, 1973; Schwefel, 1975),
- genetic algorithms (Holland, 1975), and
- particle swarm optimization (Kennedy and Eberhart, 1995).

Many of these methods contain specific parameters, which the user can specify in EvA2 (see Figure 3.1).

3.1.3 Differential evolution

Differential evolution (Storn, 1996; Storn and Price, 1997) was one of the best-performing methods in a study comparing the estimation of biochemical model parameters with different heuristic optimization algorithms (Dräger *et al.*, 2009). In differential evolution a

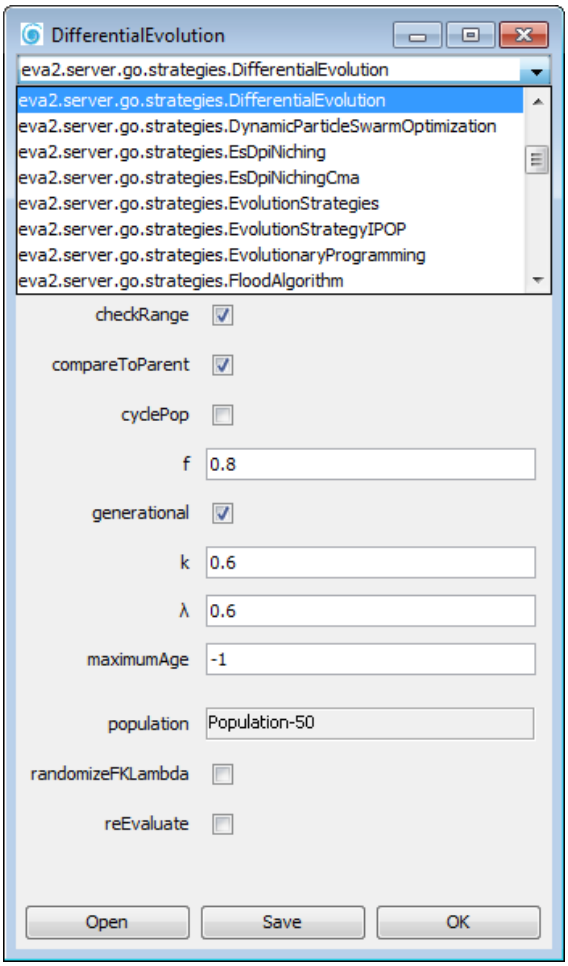

Figure 3.1: EvA2 window for selecting the optimization routine as well as specific and general parameters. This window lets the user select one of numerous optimization methods provided by EvA2. For a selected method EvA2 then enables setting several parameters. In this case differential evolution is selected and the user can, e.g., set the values F and λ, which are specific for differential evolution. General parameters, such as the population size, can also be specified in this window.

candidate solution consists of a vector of parameters. In the variant used in this thesis (named current-to-best), in each generation G a vector v is generated for every candidate solution $x_{i,G}$ as follows (Storn and Price, 1997):

$$v = x_{i,G} + \lambda \cdot (x_{best,G} - x_{i,G}) + F \cdot (x_{r1,G} - x_{r2,G}) \tag{3.1}$$

$x_{r1,G}$ and $x_{r2,G}$ are vectors from the population which are different from each other as well as from $x_{i,G}$. $x_{best,G}$ is the currently best candidate solution, whereas F and λ are constants of the algorithm set by the user.

After v has been created, it is recombined with $x_{i,G}$ based on a crossover probability $CR \in [0,1]$ yielding a vector u. If the fitness of u is better than that of $x_{i,G}$, it is included in the population of the following generation. Otherwise this population still consists of $x_{i,G}$.

3.2 Retrieving information about model structure

Structural information for constructing logical models or dynamic ODE models can be collected from a large number of literature sources. Especially for large logical models this literature search can be a very tedious process. Some models are based on hundreds of different literature sources (e.g., Ryll *et al.* (2011)).

An alternative to a literature search is the extraction of the structure of a model from existing models. One example for that is HepatoNet1 (Gille *et al.*, 2010), which is a structural model containing large parts of the hepatocyte metabolism but lacking kinetic information. If one has the aim to model parts of this metabolism dynamically, the respective sub-model of HepatoNet1 can be a good starting point.

Alternatively, the structure of a pathway can be downloaded from KEGG (Kyoto Encyclopedia of Genes and Genomes) (Kanehisa and Goto, 2000). Numerous pathways are contained in the PATHWAY section of this database (Kanehisa *et al.*, 2012). The standard format of downloaded KEGG pathways is KEGG Markup Language (KGML). KGML can be converted into other formats like SBML and SBMLqual with a dedicated software tool called KEGGtranslator (Wrzodek *et al.*, 2011). Information from KEGG can then be used for logical as well as for ODE modeling.

3.3 Optimization of logical models with CellNetOptimizer

Having information about a possible model structure given, a logical model is usually optimized with respect to experimental data. These experimental data usually comprise different experimental conditions (i.e., stimulation or inhibition of certain components

contained in the respective model). The stimulation and inhibition of components is named perturbation in general.

The optimization of the network can involve computing a logical steady state (see 2.1.1) under each experimental condition and then comparing the predicted states of the species to the data. As the model states are discrete, the data usually need to be transformed to discrete values prior to comparison. If the model prediction differs from the experimental data, one can think of model adaptations such as addition or removal of certain species or the alteration of logical gates. The described approach was used, e.g., to create an IL-1 and an IL-6 signaling model (Ryll *et al.*, 2011).

Instead of manually optimizing a logical model, a model can also be calibrated automatically. Such a routine suitable for the optimization of Boolean models is implemented in the MATLAB software CellNetOptimizer (CNO) (Saez-Rodriguez *et al.*, 2009) and in its variant in R CellNOptR (Terfve *et al.*, 2012). The experimental data are transformed into the interval $[0,1]$ with a dedicated normalization function. Besides the experimental data, the possible network structure has to be provided to CNO in the form of interactions between model species. For CellNOptR this structure needs to be given in the SIF format (see 2.1.2). This network structure is also called prior knowledge network (PKN) by the creators of CNO.

CNO first compresses the PKN by trying to remove nodes that are neither experimentally measured nor perturbed and fusing respective interactions. In the simplest case a node with one incoming interaction edge from a measured or perturbed node and one outgoing interaction edge to such a node is removed and the two interactions are fused. Besides the measured or perturbed nodes the compression finally retains only nodes that are necessary for preserving logical consistency. This involves nodes having several incoming and several outgoing interaction edges to measured or perturbed nodes.

After the compression of the PKN in an expansion step, Boolean models that fit to this compressed PKN are combined into a superstructure. This set of Boolean models should contain all combinations of possible logic gates resulting from the PKN. For a node in the network with two incoming activating interactions, the logic gate can, e.g., be an AND connection of the respective two states or an OR connection. A NOT gate is used for an inhibitory interaction. If a Boolean model comprises more then one incoming interaction for some node, these inputs are by default OR connected (compare the definition of SIF in Section 2.1.2). Therefore, CNO adds AND gates to the superstructure. In order to limit the size of the superstructure, only all AND gates consisting of some fixed maximum number of interactions are created. This maximum number is by default 2 for CellNOptR.

In the calibration step CNO now tries to find the model variant (i.e., a Boolean model comprising a certain combination of logic gates) from the superstructure that has the minimum fitness value with respect to the experimental data. The training is done with a genetic algorithm, during which the fitness $\theta(P)$ of a model variant P is computed as

follows (Saez-Rodriguez *et al.*, 2009):

$$\theta(P) = \theta_f(P) + \alpha \cdot \theta_s(P) \tag{3.2}$$

In this formula $\theta_f(P)$ stands for the mean squared error (MSE) between the normalized experimental data and the predicted logical steady states for the different experimental conditions. $\theta_s(P)$ is a term which penalizes large models. This MSE can be computed as follows (adapted from Morris *et al.* (2011)):

$$MSE = \frac{1}{N} \sum_{i=1}^{N_c} \sum_{j=1}^{Ns} (x_{\text{pred}}(i,j) - x_m(i,j))^2 \tag{3.3}$$

$x_{\text{pred}}(i,j)$ and $x_m(i,j)$ represent the predicted value for experimental condition i and species j or the respective normalized measured value, respectively. N stands for the total number of measurements, N_c for the number of experimental conditions, and N_s for the number of measured model species. If a measurement is missing for some experimental condition and species, the corresponding quadratic term is usually ignored in the computation of the MSE.

After the calibration step, the model variant with the best fitness is further processed. In a reduction step CNO removes single interactions, if this does not lead to a worse fitness value of the model.

3.4 Construction of dynamic ODE models

In contrast to logical models, for dynamic ODE models kinetic rate laws have to be added to reactions. These rate laws can be taken from literature or dedicated databases (like SABIO-RK). Another possibility is the automatic generation of kinetic equations of a certain type with SBMLsqueezer. Some kinetic parameters in these equations are often unknown, which makes their estimation necessary.

3.4.1 SABIO-RK: a database with kinetic information

SABIO-RK (Wittig *et al.*, 2012) is an online database updated and maintained by the Heidelberg Institute for Theoretical Studies. The database is continuously updated with literature information, whereby the literature data are included manually into SABIO-RK. SABIO-RK comprises kinetic equations for reactions as well as values of kinetic parameters in those equations. The kinetic entries in the database are associated with further information like the identifier of the reaction in the KEGG database (Kanehisa *et al.*, 2012), the organism the data came from, and environmental conditions (such as pH and temperature) under which the kinetic parameters were determined. A modeller can e.g., search in the SABIO-RK web interface for a specific reaction enzyme and an

organism. Then the web interface returns the rate laws fitting the query (i.e., rate laws for the reaction with kinetic parameters determined in the specified organism). The found kinetic equations can be downloaded in an SBML file. SABIO-RK contains more than 45,000 entries. With the increasing number of kinetic data integrated, the database will become even more important.

3.4.2 Automatic generation of kinetic equations with SBMLsqueezer

The addition of rate laws to SBML models by hand is very time-consuming and error-prone. Therefore, SBMLsqueezer (Dräger *et al.*, 2008, 2015) has been developed, which facilitates this process. SBMLsqueezer enables to search for rate laws in SABIO-RK and provides a dedicated wizard for this (see Figure 3.2). Alternatively, one can add specified rate laws of different types with default kinetic parameters. SBMLsqueezer supports several types of rate laws including mass-action kinetics, Michaelis Menten, convenience kinetics (Liebermeister and Klipp, 2006), and modular rate laws (Liebermeister *et al.*, 2010).

A straightforward way to use SBMLsqueezer is as follows: For all reactions in an SBML model an adequate rate equation is searched for in SABIO-RK. Typical search criteria for a reaction could be its KEGG identifier and the organism to which the model is associated. After this database search, the reactions lacking a rate law from SABIO-RK can be equipped with appropriate rate equations by SBMLsqueezer. To this end, SBMLsqueezer chooses a rate law for each reaction based on user settings and properties of the reaction like the number of reactants and products as well as the reversibility of the reaction. At the end of this process the user can look over the rate equations of all reactions. If one of those rate equations does not seem appropriate, SBMLsqueezer enables to process this single reaction individually. A search in SABIO-RK with new search criteria can be started or a different type of rate law can be chosen for the reaction.

3.4.3 BioModels database and the *path2models* project

An important repository for biological models is the BioModels database (Le Novère *et al.*, 2006; Li *et al.*, 2010), which contains hundreds of published models. These models can be downloaded in the SBML format and used for further research. Apart from published models, the BioModels database also comprises a large collection of models based on KEGG Pathway (Kanehisa *et al.*, 2012). Those models were created in the *path2models* project (Büchel *et al.*, 2013).

The *path2models* project involved downloading metabolic and non-metabolic models from KEGG, which were then processed with KEGGtranslator. The models were converted into SBML with KEGGtranslator (compare 3.2). Non-metabolic pathways were converted into qualitative models using the SBMLqual extension, whereas the metabolic pathways were transformed into ODE models. The metabolic models converted with KEGGtranslator were further processed with SBMLsqueezer, which first involved a

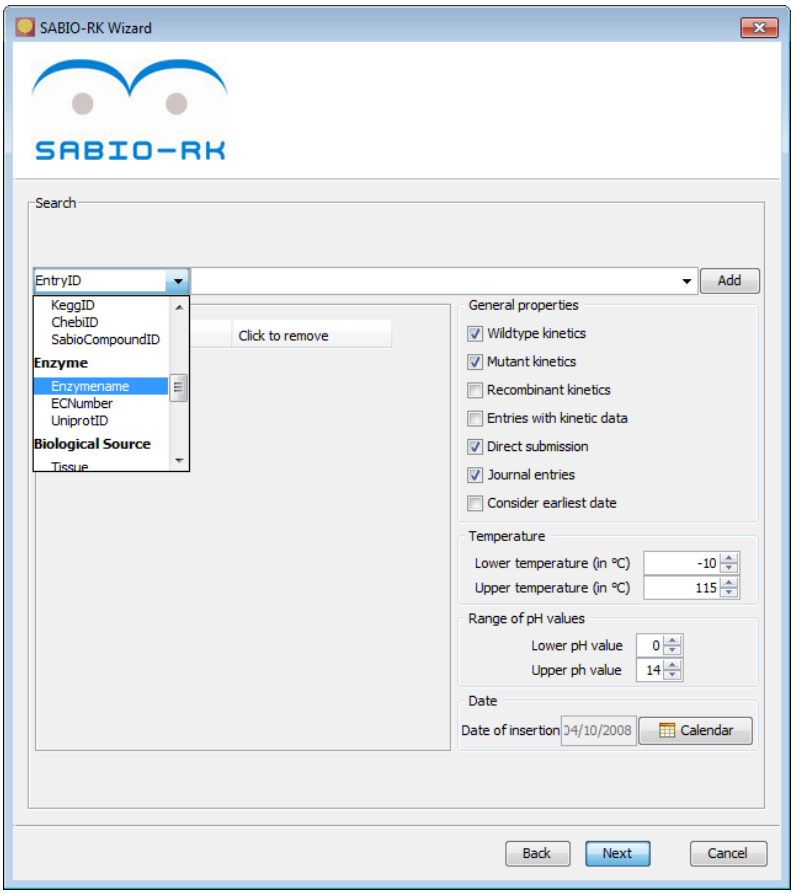

Figure 3.2: Wizard for SABIO-RK search in SBMLsqueezer. SBMLsqueezer lets the user specify different terms available in SABIO-RK for searching rate laws for a certain reaction. These terms comprise the KEGG identifier of the reaction, the name of the enzyme catalyzing the reaction, and the pH value as well as the temperature, for which the kinetic parameters of the reaction were determined.

search for kinetic rate laws in SABIO-RK. Modular rate laws with default parameter values were added to those reactions for which no kinetic law could be found in SABIO-RK. So the *path2models* project is an example how large scale ODE models can be constructed automatically.

3.4.4 Estimation of unknown parameters

Databases with kinetic information like SABIO-RK can help to derive kinetic parameters for biochemical models. However, a common problem in systems biology is that kinetic parameters are often not known or uncertain (Costa *et al.*, 2011; Chen *et al.*, 2012). Then estimation or adjustment of these parameters with respect to time-resolved experimental data is necessary (Dräger and Planatscher, 2013). This estimation of model parameters is also called model fitting. As the search space for parameter estimation is usually very large and complex, heuristic optimization routines like evolutionary algorithms are applied for this task. They are able to find optima in the parameter space in acceptable time while being robust to noisy environments (Sun *et al.*, 2012).

The fitness function that is minimized in parameter estimation represents the distance between the measured data x_m and the data x_{pred} obtained from simulation of the model with the current parameter values. The end time of this simulation should be similar to the latest time point in the experimental data. In order to enable such a simulation, all initial values of variables need to be known. The initial value of a variable can either be set to a specific value or it needs to be estimated together with the kinetic parameters. If the variable is measured in the data at time point 0, the initial value can be taken from the experimental data. Alternatively, there are often assumptions about an initial value (e.g., the value is assumed to be 0 at time point 0).

A simple way to compute the distance of x_{pred} and x_m is adding the squared distances of all similar data points (i.e., data points representing the same time point and model variable in x_{pred} and x_m). However, this gives large measurement values a high impact on the value of the fitness function, which is why the distances of the data points are usually divided by some weighting factor to circumvent this problem.

3.4.5 The problem of parameter identifiability

When estimating unknown parameters with respect to experimental data, a major problem is often that the values of several parameters cannot be identified. This unidentifiability can be due to the structure of the biochemical model (structural unidentifiability). Another possible reason for unidentifiable parameters are the experimental data used for parameter estimation (practical unidentifiability) (Raue *et al.*, 2009). These data are usually noisy and do not always contain a sufficient number of data points.

For detecting unidentifiable parameters, several methods are available (Bellu *et al.*, 2007; Quaiser and Mönnigmann, 2009; Balsa-Canto *et al.*, 2010; Kreutz *et al.*, 2013),

of which only the latter is dedicated to practical unidentifiability. Apart from these approaches, one can also assess the identifiability of parameters by repeating the parameter estimation multiple times (Schilling *et al.*, 2009). In order to exclude results for which only a local fitness optimum has been found only some of the best parameter sets (e.g., the 50% of all solutions with best fitness) are considered further. With those parameter sets, one calculates the standard deviations for all parameters. Parameters with extremely high standard deviations (e.g., more than 50% of the mean estimated parameter value) can then be considered to be unidentifiable. The described approach approach will also be applied in this work.

Chapter 4

Modeling IL-6 induced hepatic gene regulation using fuzzy logic

The aim of the Virtual Liver Network (VLN) is the development of a model involving the central functions of the human liver (Holzhütter *et al.*, 2012). These functions comprise synthesis and storage of essential substances, but also the detoxification of various substances such as, e.g., drugs. This drug detoxification can be impaired during an inflammation in the body, which might cause drug overdosing. In this chapter, which is mainly based on Keller *et al.* (2016), work in the VLN is described that involved a special type of logical modeling called fuzzy logic modeling. After an extensive literature search and analysis of different experimental data, a fuzzy logic model was finally created in order to identify regulatory events leading to a decreased drug detoxification. The main hypothesis suggested by the model was backed by further experimentation.

4.1 Impaired drug clearance during acute phase response in the liver

The human liver is the most important organ for detoxification of substances, such as prescribed drugs, in the body. 60 to 80% of all drugs are extensively metabolized (i.e., chemically modified) in the liver (Zanger *et al.*, 2008). Several proteins are responsible for drug metabolization and transport:

- Cytochromes P450 (CYPs) catalyze phase I transformation of drugs, which mainly involves oxidation of the drug molecules leading to a higher solubility (Zanger *et al.*, 2008; Zanger and Schwab, 2013).

- Phase II drug metabolizing enzymes (DMEs) consist of UDP-glucuronosyltransferases (e.g., UGT1A1, UGT2B7), sulfotransferases (e.g., SULT1A1, SULT1B1), N-acetyltransferases (e.g., NAT1, NAT2), glutathione S-transferases (e.g., GSTP1, GSTA2) and methyltransferases (e.g., TPMT) and transfer certain functional groups to the drug molecules (Jancova *et al.*, 2010).

- ABC (ATP-binding cassette) transporters (e.g., MDR1/ABCB1, ABCC2, ABCG2) are able to export drugs from cells (Petrovic *et al.*, 2007).
- SLC transporters (e.g., SLC10A1, SLC22A7, SLCO1B1) are associated with the uptake of drugs in cells (Petrovic *et al.*, 2007).

Upon infection or injury and during many chronic diseases the acute phase response (APR), which is part of the early defense system of the body, is initiated (Cray *et al.*, 2009; Gruys *et al.*, 2005). It involves the release of pro-inflammatory cytokines (i.e., mediators of inflammation) like IL-6, IL-1β, and TNFα, which leads to a drastically changed expression of several genes. Large amounts of acute phase proteins (APP) like C-reactive protein (CRP) or Serum amyloid A (SAA) are eventually synthesized.

Under inflammatory conditions the drug clearance capacity of the liver can be decreased, because many genes encoding drug metabolizing enzymes and transporters (DMET) are downregulated by the released pro-inflammatory cytokines (Aitken *et al.*, 2006; Morgan *et al.*, 2008; Klein *et al.*, 2014). This downregulation of DMET genes might cause overdosing of a drug and lead to undesirable effects such as damage of liver tissue (Morgan, 2009; Slaviero *et al.*, 2003; Renton, 2005). Some drugs, e.g., many chemotherapeutics are especially sensitive to overdosing, as their therapeutic indexes (i.e., the ratios between the toxic drug doses and the doses for the proper drug reaction) are comparably small (Harvey and Morgan, 2014). Therefore, the investigation of the mechanisms behind the downregulation of DMET genes during acute phase response is of clinical relevance. We focused on examining the regulation of DMET genes by IL-6, which is the most important mediator of the acute phase response.

4.2 Regulation of DMET genes and IL-6 signaling: previous knowledge

Transcription of genes is regulated by proteins called transcription factors. A specific type of transcription factors are nuclear receptors, which bind to DNA and whose activities are modulated by ligands binding to them (Perissi and Rosenfeld, 2005). An important role in the regulation of DMET genes was suggested for several nuclear receptors involving the aryl hydrocarbon receptor (AhR), the constitutive androstane receptor (CAR), the pregnane X receptor (PXR), and the peroxisome proliferator-activated receptor-α (PPARα) (Pascussi *et al.*, 2008; Xie, 2009). These transcription factors often act as sensors for xenobiotics (i.e., foreign substances, such as drugs), which leads to a modified transcription factor activity and ultimately to an increased gene expression. In contrast to this induced gene expression, other transcription factors like HNF-1α, HNF-4α, and CCAAT-enhancer binding proteins (C/EBPs) are responsible for a basal expression of DMET genes (Jover *et al.*, 2009; Zanger and Schwab, 2013).

There is evidence that the downregulation of DMETs by pro-inflammatory cytokines involves several of the mentioned transcription factors for human as well as for mouse

liver (Jover *et al.*, 2002; Gu *et al.*, 2006; Sun Kim *et al.*, 2007; Congiu *et al.*, 2009; Wang *et al.*, 2011). The respective studies suggested that a strong crosstalk between signaling pathways and transcription factors is responsible for this downregulation. Whether one specific signaling pathway is mainly causing the downregulation of many DMET genes was not clear. However, a coordinated mechanism with a major role of the retinoid X receptor (RXRα), which dimerizes with several nuclear receptors such as CAR, FXR, LXR, PPAR, and PXR, was proposed (Ghose *et al.*, 2004; Lefebvre *et al.*, 2010).

While the signaling pathways contributing to a modified DMET activity were not clearly identified, it is, however, known that IL-6 stimulates the JAK/STAT, MAPK/ERK, and PI3K/AKT pathways (Eulenfeld *et al.*, 2012; Ryll *et al.*, 2011). According to previous work, the downregulation of the important drug metabolizing enzyme CYP3A4 by IL-6 does not depend on the JAK/STAT pathway (Jover *et al.*, 2002), but this independence of JAK/STAT was not shown for other DMET genes. MAP kinases can presumably phosphorylate nuclear receptors, which could cause an inhibiting relocalization of the receptors (Ghose *et al.*, 2004; Burgermeister *et al.*, 2003) ultimately influencing DMET gene expression. Furthermore, PI3K/AKT might be responsible for relocalization of NF-κB, which can inactivate nuclear receptor function, but also bind to promoters (i.e., the regulatory regions of the DNA near which transcription of a gene starts) of DMET genes (Zordoky and El-Kadi, 2009). Figure 4.1 summarizes the current knowledge about possible DMET regulation induced by IL-6.

4.3 Constructing a prior knowledge network from literature

Based on literature knowledge, a prior knowledge network (PKN) involving IL-6 signal transduction and following gene regulation was developed. Figure 4.2 shows the structure of the PKN, which is available in the SIF format (see 2.1.2). The signal transduction part of the PKN is based on a Boolean model by Ryll *et al.* (2011), which contains several signaling pathways. For the sake of simplification, all feedback loops were removed from this model, as our investigation was not targeted to secondary regulatory effects. Additionally, several input and output nodes which were not relevant for us were eliminated. In order to simplify the PKN for further processing, we converted the AND, OR, and NOT gates of the network into activating or inhibiting interactions from the respective input species of the gate to the output species. Afterwards, a gene regulation module was added to the PKN by integrating biological knowledge from databases (such as BIOBASE TRANSFAC) and literature. A list of all the references the PKN is based on is provided as an appendix of this thesis (Appendix A). The resulting PKN comprises all DMET genes as well as the transcription factors STAT3, NF-κB, AhR, HNF-1α, HNF-4α, ELK1, GR, and cFOS. The complexes between RXRα and any of the nuclear receptors known to dimerize with RXRα are summarized in the network by a species

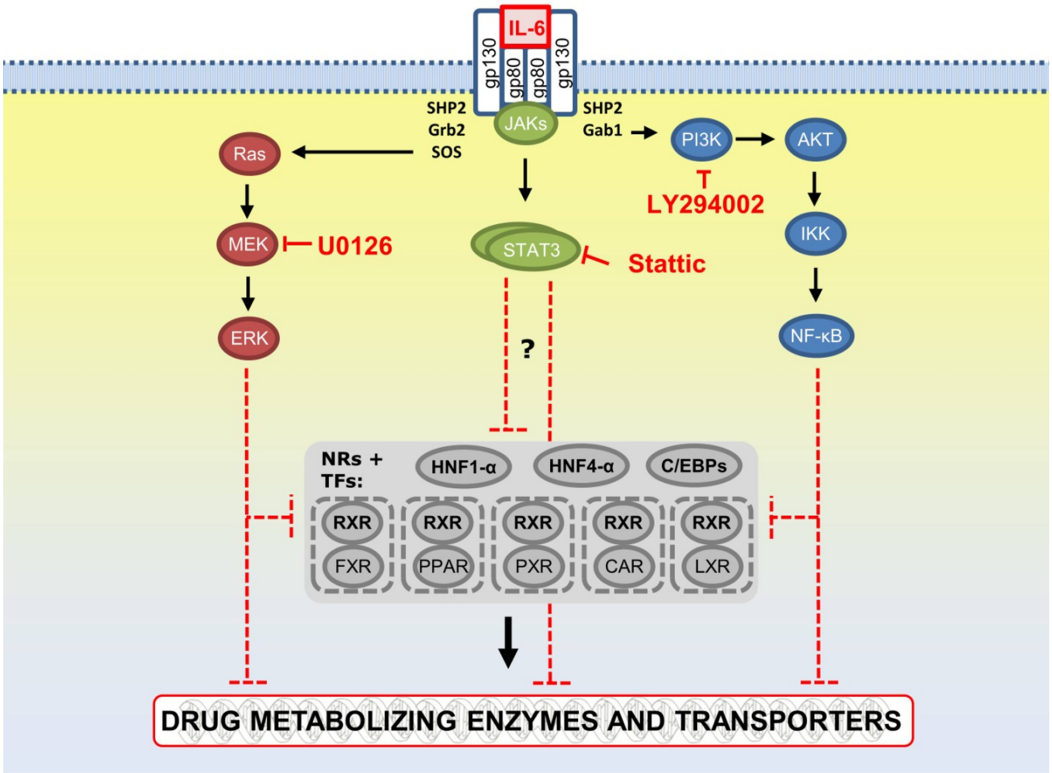

Figure 4.1: IL-6-stimulated pathways and their inhibition by selective inhibitors. The signal transduction by IL-6 proceeds via a receptor complex consisting of glycoprotein 130 (gp130) and gp80 (IL-6Rα) and stimulates the Janus kinase (JAK)/signal transducer and activator of transcription (STAT) pathway (Eulenfeld *et al.*, 2012). Additionally, IL-6 also causes activation of the mitogen activated protein kinase (MAPK)/ extracellular regulated kinase 1 and 2 (ERK1/2) (MAPK/ERK) pathway as well as of the phosphatidyl-inositol-3-kinase (PI3K) pathway (Eulenfeld *et al.*, 2012). PI3K stimulation then activates AKT (PKB) serine/threonine kinases (Cox and Der, 2002). AKT could then cause activation of IκB kinase (IKK) leading to stimulation of the NF-κB pathway (Ozes *et al.*, 1999; Romashkova and Makarov, 1999). This activation of NF-κB is, however, controversial (Delhase *et al.*, 2000). For the connection of the signaling pathways to DMET regulation, which is still not clear, several mechanisms were suggested. In the figure the PI3K/AKT pathway is indicated by blue, the JAK/STAT pathway by green, and the MAPK pathway by red color. Nuclear receptors (NRs) and transcription factors (TFs) regulating DMET genes are contained in the grey box. Positive stimulation is represented by arrows, while flat-ended arrows stand for inhibition. In experiments the pathways were inhibited by LY294002 (PI3K), Stattic (STAT3), and U0126 (MAPK), which is indicated by red flat-ended arrows. The figure was modified from Eulenfeld *et al.* (2012) as well as from Castellano and Downward (2011) and is also based on several other literature sources (see Appendix A).

RXR/NR. This is due to the fact that the influence of those complexes between RXRα and nuclear receptors on DMET genes is often similar. Furthermore, elucidating which of those complexes has the biggest influence on DMET downregulation is only possible with very specific experiments, which were beyond the scope of this investigation.

4.4 Experiments with primary human hepatocytes

The experiments in this work were conducted by the Dr. Margarete Fischer-Bosch-Institute of Clinical Pharmacology (IKP) Stuttgart and the Natural and Medical Sciences Institute (NMI) at the University of Tübingen with primary human hepatocytes (PHHs) derived from different liver donors. PHHs were used, as they are seen as the "gold standard" model for investigation of drug metabolism and its regulation in liver cells (LeCluyse and Alexandre, 2010; Godoy *et al.*, 2013). The cells were isolated from partial liver resections and cultivated for at least 48 h. Afterwards, the PHHs were treated further, which will be described next.

4.4.1 Treatments of the cells

In order to elucidate which signaling pathways regulate a certain biological process, experiments involving inhibition of these pathways are common. As also shown in Figure 4.1, all three signaling pathways known to be stimulated by IL-6 were thus inhibited by specific chemical substances:

- PI3K (PI3K/AKT pathway) was inhibited by LY294002, which was previously shown to decrease the activity of PI3K by at least 90% at a concentration above 20 $\frac{\mu mol}{l}$ (Blommaart *et al.*, 1997).
- The MAPK/ERK pathway was inhibited by U0126, a selective inhibitor for MEK1/2 at a concentration between 20 and 100 $\frac{\mu mol}{l}$ (Favata *et al.*, 1998; Goueli *et al.*, 1998).
- Inhibition of the JAK/STAT pathway was possible with Stattic, which is an effective inhibitor of STAT3 at a concentration of approximately 5 $\frac{\mu mol}{l}$ (Schust *et al.*, 2006).

In some experiments single inhibitions of the pathways were conducted, whereas in other experiments two of the three pathways were inhibited in combination. The concentrations of the chemical inhibitors in the experiments were 1 $\frac{\mu mol}{l}$, 5 $\frac{\mu mol}{l}$, or 10 $\frac{\mu mol}{l}$ (Stattic), 20 $\frac{\mu mol}{l}$ or 50 $\frac{\mu mol}{l}$ (LY294002 and U0126). In control experiments the cells were treated with dimethyl sulfoxide (DMSO), which is often used in this case, instead of the inhibitors. After the treatment with inhibitors or DMSO, respectively, the cells were then treated with

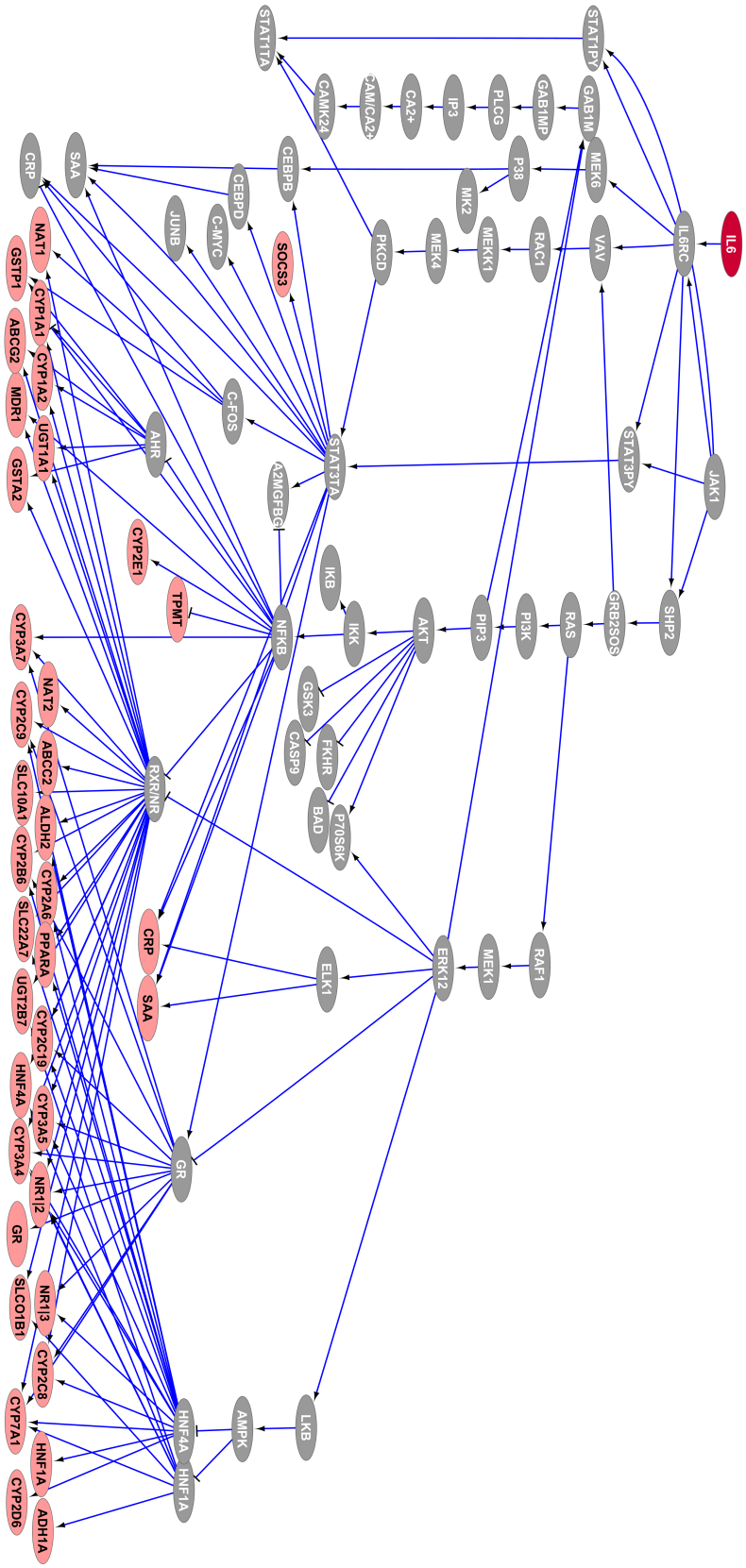

Figure 4.2: (Caption next page.)

Figure 4.2: (*Previous page.*) Prior knowledge network containing IL-6 signaling and following gene regulation. The interactions involving IL-6 signal transduction were taken from the model by Ryll *et al.* (2011). We simplified this model by deleting feedback loops as well as some input and output nodes not relevant for this investigation. Furthermore, AND, OR, and NOT gates were converted into activating or inhibitory interactions. The model was extended with some further signal transduction steps as well as with a gene regulation module based on literature information (see Appendix A). In the figure the light red boxes represent genes, whereas the grey boxes stand for signaling molecules or complexes. IL-6 is displayed in dark red. Interactions with arrows are activating, while inhibiting interactions are represented by flat-ended arrows. The figure was created with Cytoscape (Shannon *et al.*, 2003).

- 10 $\frac{ng}{ml}$ IL-6 dissolved in phosphate buffered saline (PBS) with 0.1% bovine serum albumin (BSA) or
- PBS with 0.1% BSA only for control.

In previous experiments with the specific IL-6 concentration used here STAT3 was activated and expression of the important acute phase response gene CRP was also induced without having a toxing effect on the cells (Campbell *et al.*, 2001; Vee *et al.*, 2009).

In addition to the described treatments involving the inhibition of signaling pathways by chemical inhibitors, other experiments were conducted in which RXRα was knocked-down. This knock-down was based on small interfering RNA (siRNA), which is an RNA binding to a complementary messenger RNA (mRNA). Once the siRNA binds to the mRNA, it prevents the translation of the respective protein, in this case the translation of RXRα. The activity of RXRα is thereby decreased drastically.

4.4.2 Proteomic measurements

Proteins in signaling cascades are usually activated by phosphorylation of specific amino acids contained in the protein. In order to measure protein phosphorylations, the reverse phase protein microarray (RPA) technology was applied by the NMI Reutlingen. RPA involves immobilizing protein mixtures on small arrays and screening with the help of primary antibodies which are highly selective for specific proteins (Poetz *et al.*, 2005). Secondary fluorescently labeled antibodies binding to the primary antibodies are then used to quantify the amount of primary antibody bound to the respective protein. The light intensity of a certain spot in the array is thereby measured and after a background correction with a reference spot relative fluorescence intensities (RFI) are obtained for each protein mixture and specific antibody (Braeuning *et al.*, 2011). With RPA more than 100 unphosphorylated and phosphorylated proteins (also called phosphoproteins) can be measured in parallel (Braeuning *et al.*, 2011).

After RPA, western blot analysis was conducted to confirm some findings. A western blot involves gel electrophoresis for protein separation based on molecular weight, the

transfer of the resulting protein mixture to a membrane (called blotting), and the application of a protein-specific antibody afterwards (Mahmood and Yang, 2012). The presence of a specific (phospho)protein in the protein mixture is then again detected by using a secondary fluorescently labeled antibody. Detection by an imaging system finally produces a visible band in the western blot. RPA is more suited for high-throughput experiments than western blotting, as for western blotting more resources (e.g., a higher amount of protein) are necessary (Tibes *et al.*, 2006).

4.4.3 Real-time quantitative PCR

Real-time quantitative polymerase chain reaction (qPCR) (Heid *et al.*, 1996) is a common technique for measuring the amount of different RNAs in a cell, which represent the activity of the respective genes. In the PCR a complementary DNA (cDNA) is first synthesized from the RNA and afterwards this DNA is duplicated in each PCR cycle. This means that the amount of DNA grows exponentially. In the TaqMan assay applied here a Taq polymerase is used for the synthesis of a new DNA. Furthermore, oligonucleotide probes which bind to a specific part of the DNA sequence and contain a fluorescence label at one end are also necessary. The Taq polymerase cleaves these probes during DNA synthesis thereby releasing the fluorescence label. This leads to an increased fluorescence signal. Therefore, the fluorescence signal for a certain RNA after a specific PCR cycle is dependent on the amount of produced DNA. As the cDNA is synthesized from the RNA and duplicated in each PCR cycle, the amount of DNA thus depends on the amount of RNA present in the sample. The cycle threshold (Ct) value is the PCR cycle in which the fluorescence signal exceeds a certain threshold value. A high gene activity is associated with a low Ct value.

2,304 simultaneous real-time qPCRs are enabled by the Fluidigm platform allowing high-throughput experiments (Spurgeon *et al.*, 2008). Here the expression of 86 genes was quantified for several different treatments and donors by the IKP Stuttgart. The Fluidigm Real-Time PCR Analysis Software enabled automatic determination of Ct values.

4.5 Activation of signaling pathways by IL-6

Activation of different phosphoproteins after IL-6 stimulation in primary human hepatocytes is shown for several time-points in Figure 4.3. Part A of the figure comprises the measurements of 32 phosphoproteins using the reverse phase protein microarray technology (see 4.4.2). The data suggested phosphorylations of AKT S473, ERK1/2 T202/Y204, STAT1 Y701, and STAT3 Y705, where identifiers starting with "S", "T", and "Y" stand for phosphorylations of serine, threonine, and tyrosine residues, respectively. The relative fluorescence intensities (RFI) of these four phosphoproteins upon IL-6 stimulation are also shown in comparison to the respective control experiments (Figure 4.3, B – E, left panel). ERK1/2 and STAT3 were already phosphorylated within 5 minutes after

Table 4.1: Primary hepatocytes and treatments. The different male (m) and female (f) donors and their treatments with IL-6 and specific inhibitors are shown here. Inhibitors for PI3K (PI3Ki), MEK1/2 (MAPKi), and STAT3 (STAT3i) were used for five of the six donors. For one donor RXRα knock-down experiments with siRNA were conducted (siRXRα).

Donor number	Age	Sex	Treatments
1	59	f	IL-6, IL-6 + PI3Ki
2	47	f	IL-6, IL-6 + MAPKi + PI3Ki
3	29	f	IL-6, IL-6 + STAT3i
4	48	m	IL-6, IL-6 + STAT3i + MAPKi
5	71-80	f	IL-6, IL-6 + MAPKi, IL-6 + PI3Ki, IL-6 + STAT3i
6	21-30	m	IL-6, siRXRα

stimulation with IL-6, whereas phosphorylation of AKT and STAT1 occured after 10 - 30 minutes. The increase in phosphorylation was highest for STAT3 (more than 20-fold). The findings with RPA were confirmed by western blot analysis (Figure 4.3, B – E, right panels), as the bands associated with the four phosphoproteins were contained in the western blots. Therefore, the proteomic measurements demonstrate activation of AKT, ERK1/2, STAT1, and STAT3 and thus the stimulation of all three pathways associated with IL-6 signaling.

4.6 Analyzing the gene expression data

While activation of the three signaling pathways was shown with the proteomic data, the key question which of those pathways contributes to DMET gene regulation cannot be answered by those data. In order to investigate this in depth, gene expression of DMET genes was measured 24 hours after IL-6 stimulation using real-time qPCR (see 4.4.3). Besides DMET gene expression, the expression of genes confirming inflammation or activation of a certain pathway was also measured. Specific chemical inhibitors for STAT3, PI3K, and MAPK signaling were applied (see 4.4.1 and Figure 4.1). Primary human hepatocytes from five liver donors were used for gene expression measurements (Table 4.1, donors 1-5). From the data involving inhibitions of signaling pathways only the fold changes of experiments with the highest concentrations of each chemical inhibitor were considered further. For these inhibitor concentrations we assumed complete inactivation of the respective signaling pathway.

4.6.1 The ΔΔCt method for determining fold changes

Real-time qPCR data comprise cycle threshold (Ct) values (see 4.4.3). For transformation of the Ct values into relative gene expression changes, the ΔΔCt method (Livak and

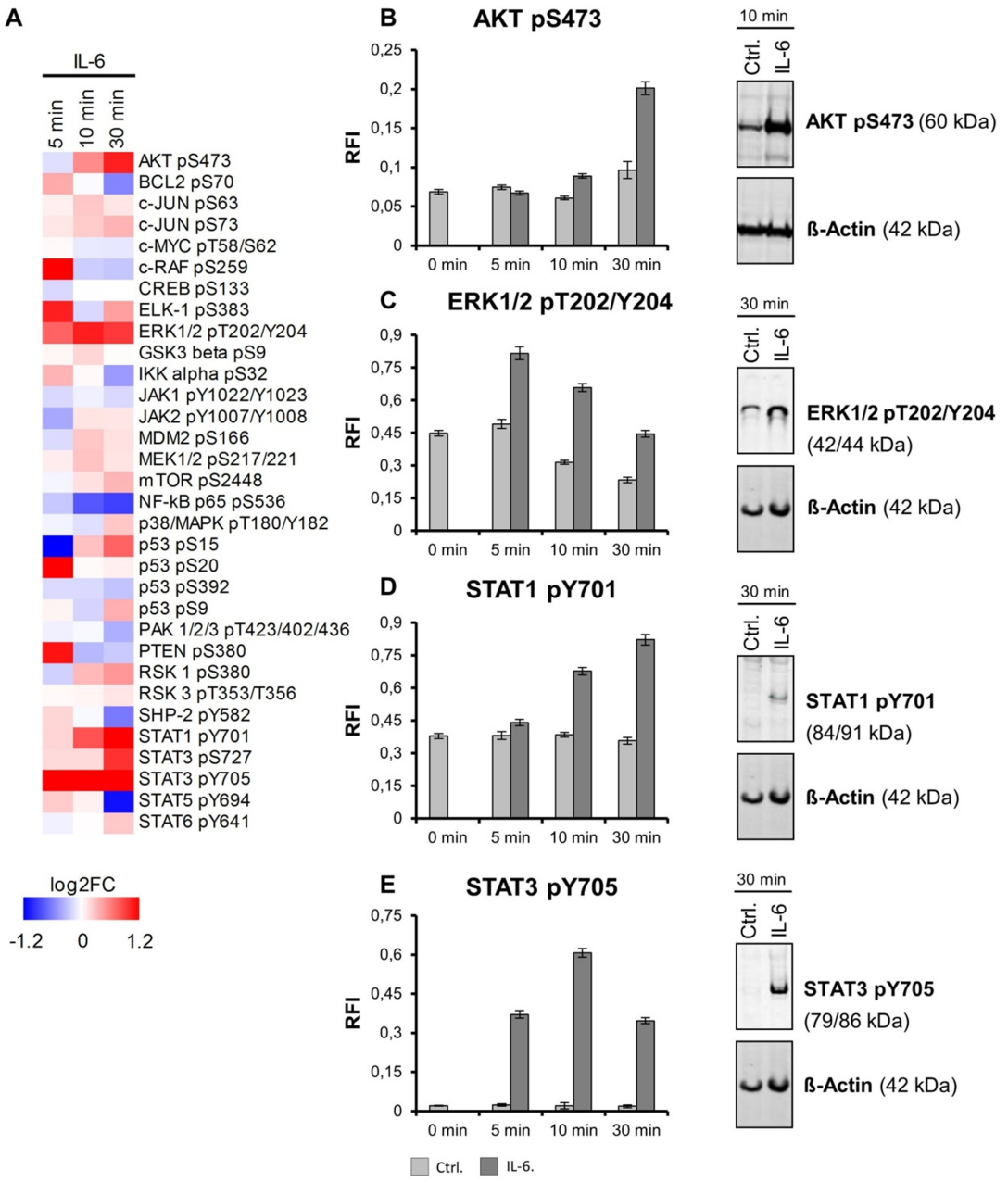

Figure 4.3: Activation of phosphoproteins after IL-6 stimulation in primary human hepatocytes. (A) This heat map comprises the relative intensity changes (IL-6 vs. control) of 32 phosphoproteins at three time points. Reverse phase protein microarray (RPA) was used for measuring the phosphoproteins. A positive log2FC (fold change) is indicated by red color, whereas blue color stands for a negative log2FC. (B-E, left panel) The relative fluorescent intensities (RFIs) from the RPA experiment of AKT pS473, ERK1/2 pT202/Y204, STAT1 pY701, and STAT3 pY705 at different time points after IL-6 stimulation (dark grey) are displayed together with their respective controls (Ctrl, light grey). Standard deviations calculated from four technical replicates are represented by error bars. (B-E, right panel) Here the western blots of the respective phosphoproteins are displayed. The western blot involving β-Actin is a positive control.

Schmittgen, 2001) is often used. The first step involves normalization of the Ct values of all genes to the Ct values of a housekeeping gene (HKG). This HKG, in our case glyceraldehyde 3-phosphate dehydrogenase (GAPDH), is an unregulated gene that is constitutively expressed. From the mean Ct value $\overline{Ct(G,T)}$ of the gene G for treatment T, a ΔCt value was calculated by subtracting the respective mean value for GADPH:

$$\Delta Ct(G,T) = \overline{Ct(G,T)} - \overline{Ct(GAPDH,T)} \tag{4.1}$$

Afterwards, ΔΔCt values were determined by subtraction of the ΔCt value for the respective control treatment $C(T)$ (e.g., PBS, 0.1% BSA-treated) from the ΔCt of a certain treatment T (e.g., IL-6-treated).

$$\Delta\Delta Ct(G,T) = \Delta Ct(G,T) - \Delta Ct(G,C(T)) \tag{4.2}$$

As in each PCR cycle the amount of DNA is duplicated and a higher Ct value represents a lower gene activity, fold changes $fc(G,T)$ were then calculated from the ΔΔCt values with the following formula:

$$fc(G,T) = 2^{-\Delta\Delta Ct(G,T)} \tag{4.3}$$

4.6.2 Clustering of the data

Having applied the ΔΔCt method, each gene can now be represented by a vector of fold changes for the different treatments. In order to compare the fold change values for the genes and treatments, a heat map of the genes and treatments based on the logarithmized fold changes was produced using the R function *heatmap.2* (Warnes *et al.*, 2013). This function involves hierarchical clustering of the genes and the treatments. We used average linkage clustering (Hartigan, 1975) as clustering method. Beginning with clusters containing only a single element, in each step the two clusters having the shortest Euclidean distance are fused to one cluster until only one cluster remains. In the average linkage method the distance $D(C_1,C_2)$ of two clusters C_1 and C_2 is computed from the distances between two elements d with the following formula:

$$D(C_1,C_2) = \frac{1}{|C_1||C_2|} \sum_{c_1 \in C_1, c_2 \in C_2} d(c_1,c_2) \tag{4.4}$$

Figure 4.4 displays the resulting heat map. The heat map shows that with the exception of CYP2E1 all CYPs as well as several ABC and SLC transporters were clearly downregulated by IL-6. CRP and SOCS3, which are known to be upregulated during acute phase response (APR) or by IL-6, respectively, are contained in one major gene cluster. Several measurements are missing for the third APR gene SAA making clustering of this gene not reliable. Another upregulated gene is CYP2E1. Its upregulation was, however, not seen in all donors, which is why this gene is separated from CRP and SOCS3 in the

hierarchical clustering. One major cluster of the treatments on top of Figure 4.4 comprises all the treatments including a MAPK and/or a PI3K inhibitor. In this cluster most of the effects induced by IL-6 were significantly reduced. Treatments involving only an inhibition of STAT3 were clustered together with the IL-6 treatments without inhibitors. Therefore, the data suggest that the JAK/STAT pathway plays a less important role in the IL-6 induced regulation of DMET genes, whereas MAPK and PI3K signaling are strongly involved in this regulation.

4.7 Fuzzy logic modeling for optimization of the prior knowledge network

The clustering results indicate the impact of the signaling pathways on IL-6 induced DMET gene regulation. However, the involved regulatory events remain to be identified. To this end, a fuzzy logic model was developed and calibrated based on the prior knowledge network (Figure 4.2) and the gene expression data (see 4.6).

Different kind of modeling techniques have been used to improve understanding of complex signaling pathways and transcription networks The input-output behavior of signaling pathways can be characterized qualitatively by logical models (Samaga and Klamt, 2013). Boolean models, which allow the model species only to be in an active or inactive state, are the most frequently applied type of logical models. Very large Boolean models can be developed, but they are often not sufficient to describe biological reality (Morris *et al.*, 2011). In contrast to that, the most precise modeling approach are ordinary differential equation (ODE) models, which permit to simulate signaling dynamics over time (Hug *et al.*, 2013). For the calibration of ODE models time-resolved experimental data and knowledge about the involved signal mechanisms are necessary, which is why ODE modeling is often only adequate for small networks. Fuzzy logic modeling is an intermediate approach between Boolean and ODE modeling. It allows species states to be in a continuous interval, but the respective models are usually not dynamic, which is why time-resolved data are not required for model optimization. Fuzzy logic modeling has been used in studies comprising manual calibration of model parameters (Aldridge *et al.*, 2009), whereas in other studies the parameters of fuzzy logic models were estimated with heuristic optimization routines (Morris *et al.*, 2011; Bernardo-Faura *et al.*, 2014). Manual parameter estimation was not applicable for our work, as we did not have prior knowledge about model parameters. Instead we adapted a method to train signal transduction pathways to protein data (Morris *et al.*, 2011) such that it could also be used with respect to gene expression data. The method and the adaptations (concerning data normalization) will be shown next.

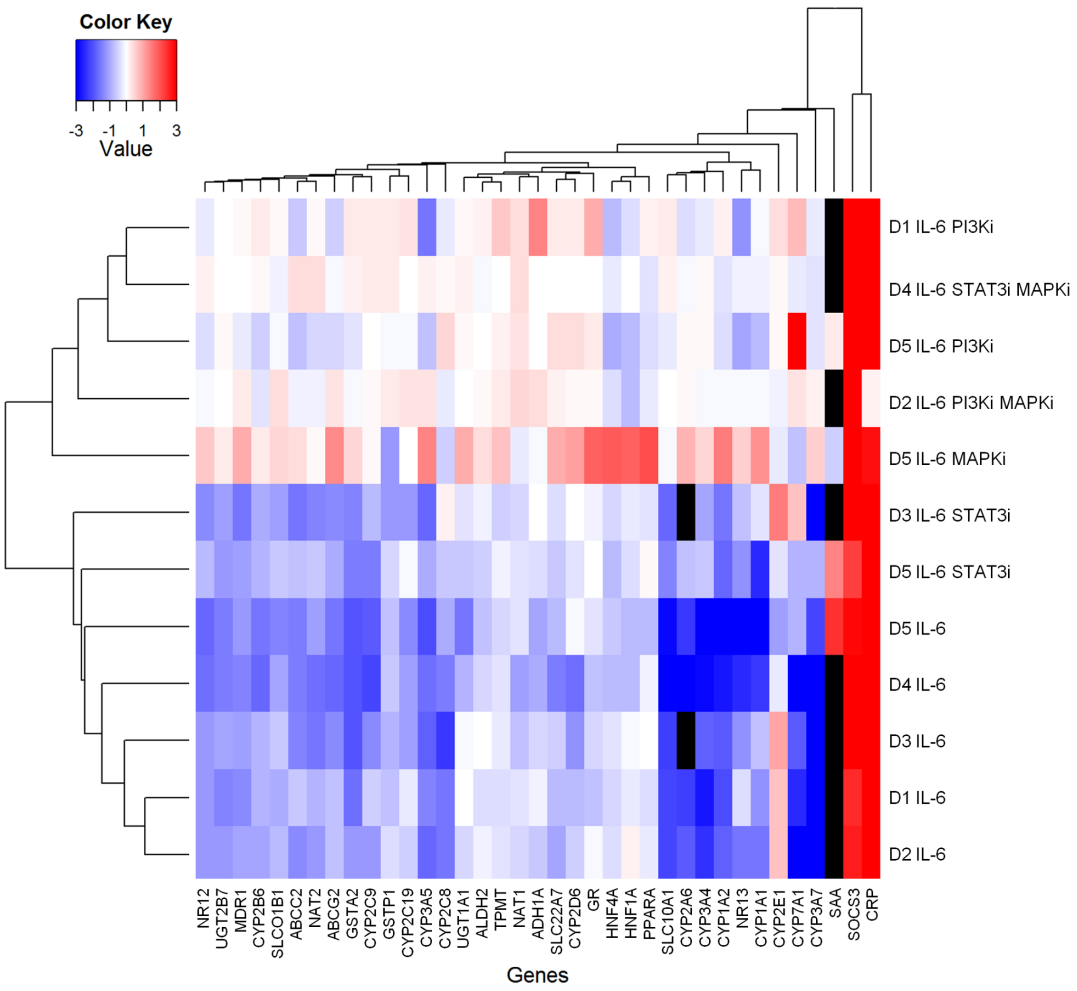

Figure 4.4: Heat map of DMET gene expression data. Both genes (in columns) and treatments (in rows) are hierarchically clustered based on the logarithmized fold change values. The treatments comprise IL-6 stimulation and chemical inhibitions of signaling pathways in five liver donors (Table 4.1, donors 1-5, abbreviated with "D" here). Elements colored in black represent missing measurements, while red stands for upregulation and blue for downregulation. One major gene cluster contains SOCS3 and CRP, which are heavily upregulated by IL-6, whereas the second major gene cluster contains all other genes. Clustering of the treatments produces one major cluster involving all treatments with inhibitions of PI3K and/or MAPK. This heat map has been created with R software package (R Development Core Team, 2011).

4.7.1 State computation and model training in CNORfuzzy

For creation of an optimized model we used CellNOptR (Terfve *et al.*, 2012), the R-variant of CellNetOptimizer (see 3.3). The original method (Saez-Rodriguez *et al.*, 2009) only allows the model species to be in 2 states, 0 and 1. However, CellNOptR comprises CNORfuzzy for fuzzy logic modeling as an add-on, which enables a continuous interval $[0,1]$ for the species states. CNORfuzzy was previously applied to proteomic data (Morris *et al.*, 2011). In contrast to that, we use CNORfuzzy to develop a fuzzy logic model based on normalized gene expression data. Normalization of the data will be discussed later, while computation of the species states of a fuzzy logic model and the optimization routine are explained next.

Computing the species states in a fuzzy logic model

The determination of the species states of a fuzzy logic model can be compared to logical steady state computation for Boolean networks (see 2.1.1). However, as the states in the fuzzy logic model are in the interval $[0,1]$ instead of only being 0 or 1, the Boolean functions need to be generalized. Adapted Hill functions (also called transfer functions), which lead to values in the interval $[0,1]$ for all possible inputs, are used for this purpose. The value c of a node C depending only on node value A having value a is calculated as follows (Morris *et al.*, 2011):

$$c = (k^n + 1)\frac{a^n}{k^n + a^n} \tag{4.5}$$

In this function, which produces values in the interval [0,1] for $a \in [0,1]$, k is the midpoint of the function and n is the Hill coefficient. If in contrast C is inhibited by A, the Hill function is subtracted from 1:

$$c = 1 - (k^n + 1)\frac{a^n}{k^n + a^n} \tag{4.6}$$

If in addition to A B with value b is input of C and A and B are OR connected, the value c is obtained by taking the maximum value of the Hill functions based on A (with index 1) and B (with index 2):

$$c = \max((k_1^{n_1} + 1)\frac{a^{n_1}}{k_1^{n_1} + a^{n_1}}, (k_2^{n_2} + 1)\frac{b^{n_2}}{k_2^{n_2} + b^{n_2}}) \tag{4.7}$$

Correspondingly, in the case of an AND connection of A and B, the result of the logic gate is defined as the minimum of the two Hill functions:

$$c = \min((k_1^{n_1} + 1)\frac{a^{n_1}}{k_1^{n_1} + a^{n_1}}, (k_2^{n_2} + 1)\frac{b^{n_2}}{k_2^{n_2} + b^{n_2}}) \tag{4.8}$$

If the input of those gate functions are only Boolean values (i.e., 0 or 1), the result is the same as in Boolean logic, which is why the functions are proper generalizations of the Boolean functions.

Model optimization routine

The first two steps of the CNORfuzzy optimization method involve network compression as well as expansion and are similar to those in the CNO routine (see 3.3): Initially the network is compressed to all species that are either measured or perturbed in the data or necessary for the preservation of logical consistency. In the next step CNORfuzzy expands the compressed network by possible AND gates comprising a maximum number of inputs (two by default).

The following model calibration step consists of running a genetic algorithm in order to minimize the mean squared error (Equation (3.3)) between the values predicted by the model and the normalized experimental data. In contrast to the CNO routine, the calibration method of CNORfuzzy not only involves the determination of the logic gates contained in the optimized model, but also the optimization of the respective Hill functions (i.e., the parameters k and n). The search space is restricted to seven different combinations of fixed values for k and n for each transfer function, which led to decent fitting results (Morris *et al.*, 2011). In addition to those seven transfer functions, the corresponding logic gate can also be inactive.

As stimulation of input species (IL-6 in our model) is usually assumed to be complete, these species are 1 by definition. This always leads to values of 1 in the Hill functions containing those species (see Equation (4.5)). In order to avoid this, the Hill functions based on those species are substituted by multiplications of the species value by factors from the interval $[0, 1]$. Those factors are then also determined in the genetic algorithm. The number of possible values for such a factor is also seven here and again the corresponding gate can be inactive.

Compared to the CNO routine, in which a gate can be active or inactive (i.e., contained or not contained in the model), the genetic algorithm in CNORfuzzy thus involves a much larger search space. The algorithm can get stuck in a local minimum of the fitness function. Furthermore, there could be several different models yielding a similar fitness. Therefore, multiple runs of the genetic algorithm are necessary for determining the relevance of certain logic gates for fitting the model to the experimental data. A family of fuzzy logic models is the result of such multiple calibration runs. From those models logical redundancies are removed. A logical function "(A AND B) OR A" for a node could, e.g., be substituted by "A", which is why in this case the corresponding AND gate would be removed from the model.

Model reduction and refinement

In the model reduction step gates from the optimized models are removed, if this does not increase the MSE by more than a defined reduction threshold. This adaptation of the models decreases the number of model parameters. Several reduction thresholds are tested yielding a different model family for each reduction threshold. The following refinement step then involves finding a local optimum for the model parameters near the "discrete" optimum obtained from the genetic algorithm. In CNORfuzzy the Subplex algorithm (Rowan, 1990) implemented in the R package *nloptr* (Johnson, 2014) is used for that purpose. In the end a selection threshold is chosen from the tested reduction thresholds such that the number of parameters is as low as possible without significantly increasing the mean MSE between model prediction and normalized experimental data. How this selection threshold is chosen will get obvious in Section 4.8.

4.7.2 Data normalization

The optimization routines of CellNetOptimizer and CNORfuzzy require normalized data in the interval $[0,1]$. This condition is neither fulfilled for the Ct values nor for the calculated fold changes from our gene expression data. The fold change values from our data therefore need to be transformed to the interval $[0,1]$. A normalization method is provided by CNO, which is, however, based on phosphoproteomic data (Saez-Rodriguez *et al.*, 2009). This method involves determining fold changes with respect to the values at time point 0 and applying a Hill function to these fold changes afterwards. For the fold change values computed with the ΔΔCt method, a similar Hill function with adapted parameters can be directly used for transformation of those values (i.e., the ΔΔCt values).

The following function, which depends on the Hill coefficient h and the value m representing the midpoint of the normalization function, was thus applied to each such fold change value fc_i:

$$v_i = \frac{fc_i^h}{m^h + fc_i^h} \tag{4.9}$$

For genes downregulated by IL-6 the fold change values are usually smaller than 1, while these values are greater than 1 for upregulated genes. The fold change values of control experiments are 1 by definition and represent a comparably low gene activity level for upregulated genes, but a high activity level for downregulated genes. If the same midpoint m of the Hill function was chosen for all genes, this control value of 1 would always be transformed to 0.5. The other fold change values would then be transformed either to values in the interval $[0.5,1]$ (for upregulated genes) or to values in the interval $[0,0.5]$ (for downregulated genes). In order to ensure a spread of the fold change values throughout the whole interval $[0,1]$ for different treatments, we therefore chose a different m for both types of genes. The setting of m to 0.5 proved effective for all genes downregulated by IL-6. Correspondingly, for the upregulated gene CYP2E1

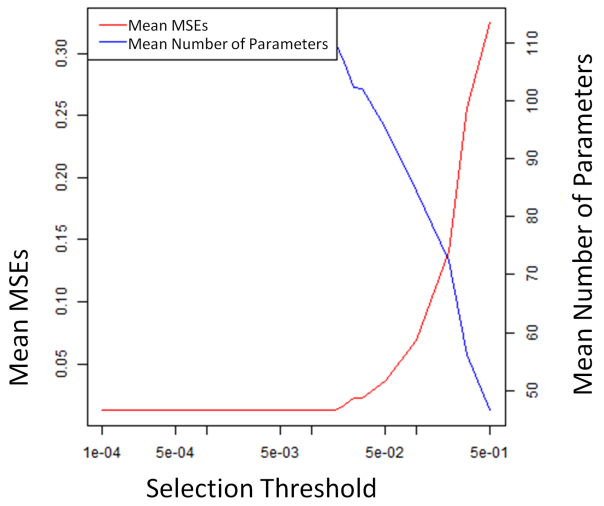

Figure 4.5: Reduction curve for model family resulting from optimization of the PKN. This figure shows how the mean squared error (MSE) and the mean number of parameters of the model family depend on the selection threshold. As the mean MSE heavily increases after 0.01, this value is a good choice as selection threshold. The figure was created with CNORfuzzy (Terfve *et al.*, 2012).

a value of 2 was chosen for m. Because of the large fold change values upon IL-6 stimulation for the other upregulated genes (SAA, CRP, SOCS3) 2 was not a suitable value for the midpoint of the Hill function. In contrast, m was set to half of the mean fold change value of the IL-6 treatments over the data sets. Choosing 4 as value for h led to a transformation of the control fold change to a value near 0 (for the genes upregulated by IL-6) or 1 (for the downregulated genes). As desired, for each gene the transformed values were then spread over the whole interval $[0, 1]$.

4.8 Results of optimizing the prior knowledge network

With the adapted routines of CNORfuzzy (Terfve *et al.*, 2012) described in the previous section we constructed an optimized fuzzy logic model from the gene expression data and the prior knowledge network. The five gene expression data sets containing single inhibitions of STAT3, PI3K, and MAPK as well as combinatorial inhibitions of two pathways (Table 4.1, donors 1-5) were used for model training. We ran the genetic algorithm for optimization and the following reduction procedure 100 times. We thus obtained a family of optimized and reduced models. The selection threshold was set to 0.01 leading to an average MSE for the 100 models of 0.013 (see Figure 4.5).

4.8.1 Calibration results

The predictions of the model family using the described settings were compared with the respective data points (see Figure 4.6). It is obvious from the figure that the normalized fold change values had large standard deviations for some treatments and genes, such as the values of CYP2E1 for the STAT3 inhibition treatment. This demonstrates that there is a variability between the liver donors. For most of the genes and treatments, however, the fold changes were similar throughout different donors.

Only for CYP2C8, CYP7A1, and SOCS3 the deviations between model predictions and normalized experimental data were exceptionally high. This could be due to unknown regulatory events not contained in the prior knowledge network, but influencing expression of the genes.

4.8.2 The resulting optimized model

Figure 4.7 shows the model family after optimization of the PKN. As suggested by the model, several events contribute to the downregulation of many DMET genes by IL-6:

- The most important event for this downregulation is presumably the inhibition of the complexes between RXRα and the nuclear receptors by MAPK and Nf-κB.
- Another decisive event in this regard seems to be the inhibition of HNF-1α and HNF-4α by MAPK.
- Furthermore, the model suggests that the downregulation of a few genes is due to the inhibition of the glucocorticoid receptor (GR) by MAPK and the inhibition of AHR by NF-κB, respectively.

For the described events the necessary logic gates are contained in nearly all of the 100 optimized models. This shows the importance of the events for explaining the data.

4.8.3 Validation of the role of RXRα

An important role of the complexes between RXRα and nuclear receptors was suggested by the optimized fuzzy logic model. Knock-down of the RXRα gene and following real-time qPCR (see 4.4.3) was conducted in order to confirm this result. Primary human hepatocytes from one donor (Table 4.1, donor 6) were used here. 48 hours after the beginning of the knock-down the RXRα protein was not detectable on a western blot (Figure 4.8, A), which demonstrates its proper knock-down. Figure 4.8 B shows the gene expression changes in the primary human hepatocytes upon IL-6 stimulation and RXRα knock-down. The heavy downregulation (more than 90%) of RXRA mRNA upon RXRα knock-down again verifies the proper knock-down. After IL-6 stimulation, the acute phase response genes were strongly upregulated and most DMET genes were

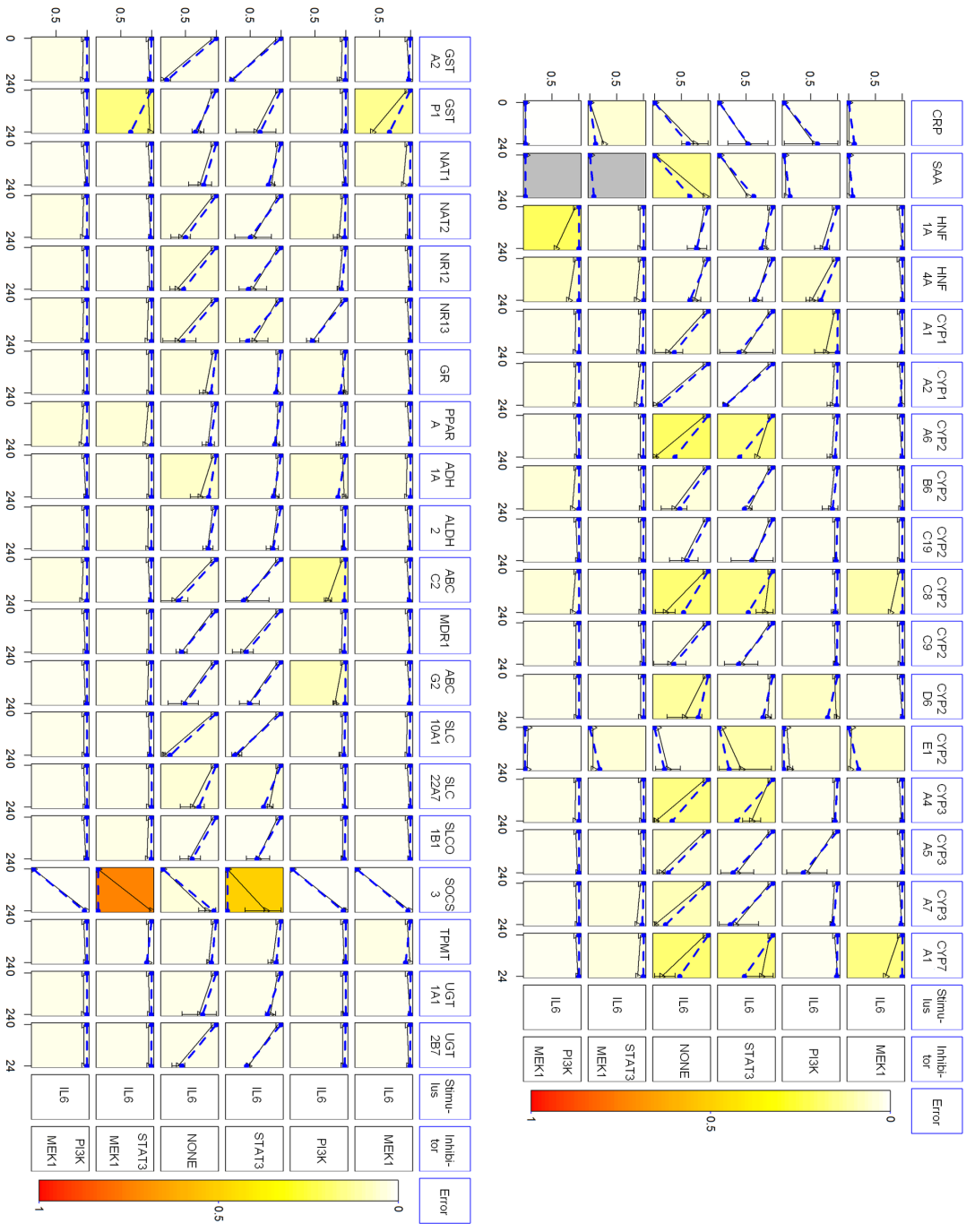

Figure 4.6: (Caption next page.)

Figure 4.6: (*Previous page.*) Comparison of the predictions of the optimized PKN with the respective data points from the gene expression data. The figure shows the normalized data points (in black) compared to the points predicted by the model family (in blue) for all genes (represented in the columns) and treatments (represented in the rows). Table 4.1 also contains the respective treatments (donors 1-5). Vertical bars display the standard deviations of the normalized data points throughout the different donors around the mean value. As gene expression values for time point 0 were not given in the data, we here assume that the value at time point 0 corresponds to the value for the control treatment at time point 24. The figure was created with an adapted method of CNORfuzzy (Terfve *et al.*, 2012).

downregulated, which was similar to the expression data for the other donors (see 4.6). Knock-down of RXRα showed comparable effects for the gene expression of the CYPs. Only CYP1A1 expression was not decreased by the knock-down. The patterns of downregulation for the transporter genes ABCC2, ABCG2, SLC22A7, and SLCO1B1 were comparable for RXRα knock-down and for IL-6 stimulation, whereas some phase II metabolism genes (UGT1A1, UGT2B7, GSTP1) were not downregulated by the knock-down. For GSTP1 this is in agreement with the modeling results, as the model suggests its IL-6 induced downregulation via AHR (Figure 4.7). The genes AHR, HNF4A, NR1I2/PXR, and NR1I3/CAR, which encode transcription factors with regulatory influences on DMET genes, were clearly inhibited after the knock-down of RXRα as well as upon IL-6 stimulation. However, for the NR1I3 and HNF4A genes an influence of RXRα was not suggested by the model. RXRα knock-down induced expression of acute phase genes (CRP, SAA) and SOCS3 to a clearly lesser extent than IL-6 stimulation, which is in agreement with the model.

The Pearson correlation coefficient can be determined to investigate whether two variables x and y are correlated. It is defined as follows:

$$r(x,y) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{4.10}$$

The possible values of r are from the interval $[-1,1]$, where 1 represents full positive correlation, -1 full negative correlation, and 0 stands for no correlation. In this case we set x to the fold change values of the genes upon IL-6 stimulation and y to the respective values after RXRα knockout. This produces a value of 0.33 for r. Now it remains to be determined whether this value represents a significant correlation here. It is known that under the null hypothesis of no correlation the following value is t-distributed with $n-2$ degrees of freedom (Lowry, 2015):

$$t = r\sqrt{\frac{n-2}{1-r^2}} \tag{4.11}$$

The corresponding p-value for a specific value of t shows whether the respective corre-

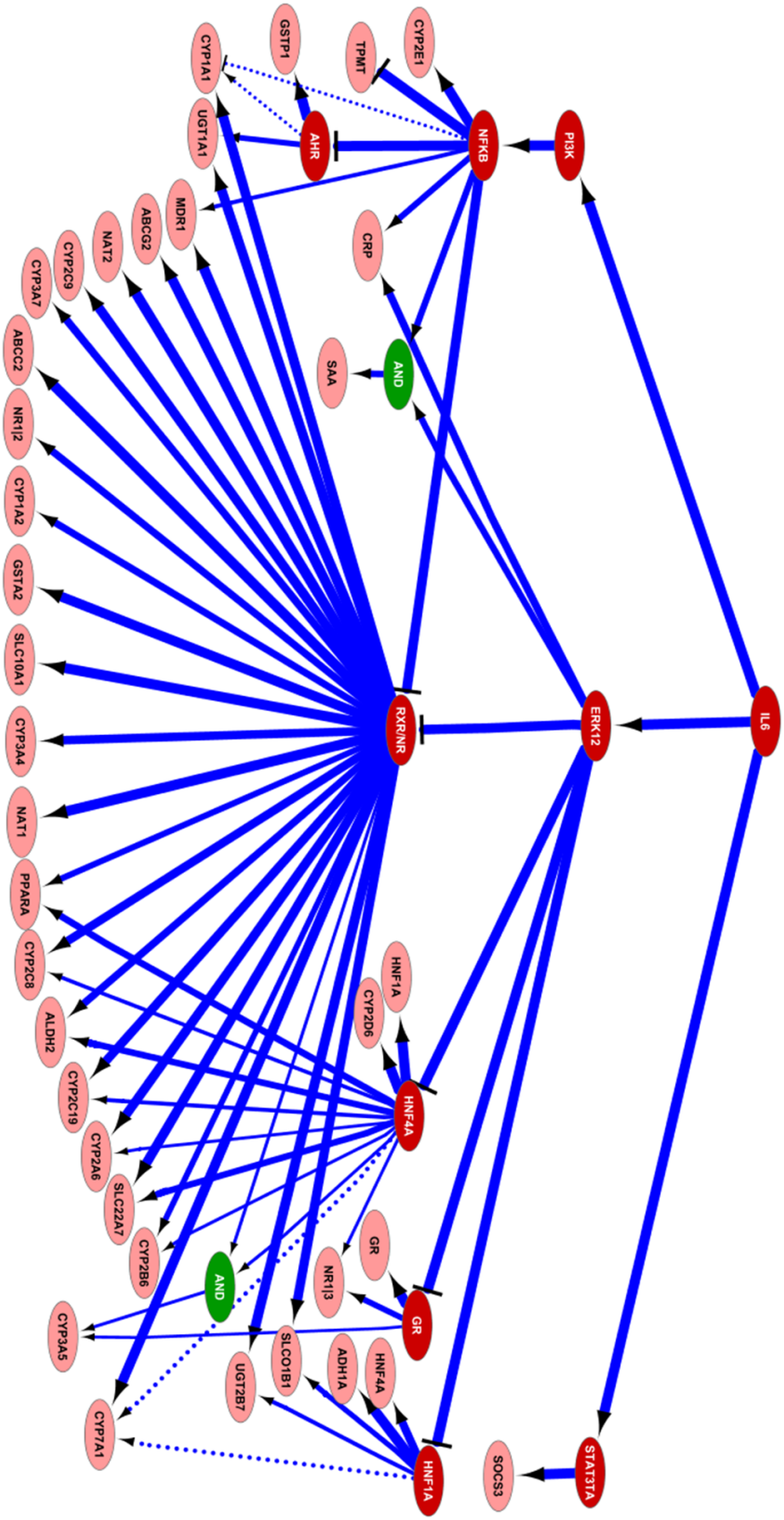

Figure 4.7: (Caption next page.)

Figure 4.7: (*Previous page.*) Model family obtained from optimizing the PKN with respect to the gene expression data. The figure shows the resulting model family and includes only logic gates that are contained in at least 30% of the 100 optimized models. By default multiple inputs of a node are OR connected. In contrast to that, AND gates are specified with additional AND nodes. Signaling molecules in the model are colored in red, genes in lighter red, AND nodes in green. Some important transcription factors compressed by CNORfuzzy before optimization (HNF-1α, HNF-4α , NF-κB, AhR) were added again here for better clarity. If this caused non-uniqueness for some regulatory events, the respective logic gates are dotted (e.g., CYP7A1 is regulated either by HNF-1α or by HNF-4α or by both factors, but the regulatory event cannot be concluded from the compressed models). The line width of a gate corresponds to the percentage of the 100 optimization runs, in which the gate was retained in the model. The figure was created with Cytoscape (Shannon *et al.*, 2003).

lation is significant. In our case a p-value smaller than 0.01 was obtained, which is why the correlation between IL-6 stimulation and RXRα knock-down is highly significant.

Overall a coordinating role of RXRα in the downregulation of DMET genes is backed by the knock-down experiments. However, a few genes for which the model suggests a regulation by RXRα are not affected by RXRα knock-down, and vice versa. In order to investigate whether this is specific to the donor used in the knock-down experiments or whether another transcription factor plays a decisive regulatory role for those genes, further experiments beyond the scope of this work are necessary.

4.9 Comparison of main hypothesis to previous knowledge

As the major event for the IL-6 induced downregulation of most DMET genes the model suggested the inhibition of the complexes of RXRα and nuclear receptors by MAPK and Nf-κB. The function of several nuclear receptors including CAR, FXR, LXR, PPAR, and PXR is known to depend on dimerization with RXRα (Lefebvre *et al.*, 2010). A role of MAPK and Nf-κB in the inhibition of these complexes was suggested previously (Burgermeister *et al.*, 2003; Zordoky and El-Kadi, 2009). A decisive impact of RXRα in the downregulation of DMET genes was previously proposed (Ghose *et al.*, 2004), but the study involved only few mouse genes and the hypothesis has to our knowledge not been tested for humans so far. In the study a loss of RXRα in the cell nucleus during acute phase response induced by endotoxin was shown, while the gene expression of RXRα was unchanged. This unaffected gene expression of RXRα corresponds to the unchanged RXRα expression upon IL-6 shown in Figure 4.8. The molecular events causing RXRα inhibition need to be investigated in more detail. Phosphorylation of nuclear receptors could play a key role in this regard (Ghose *et al.*, 2004).

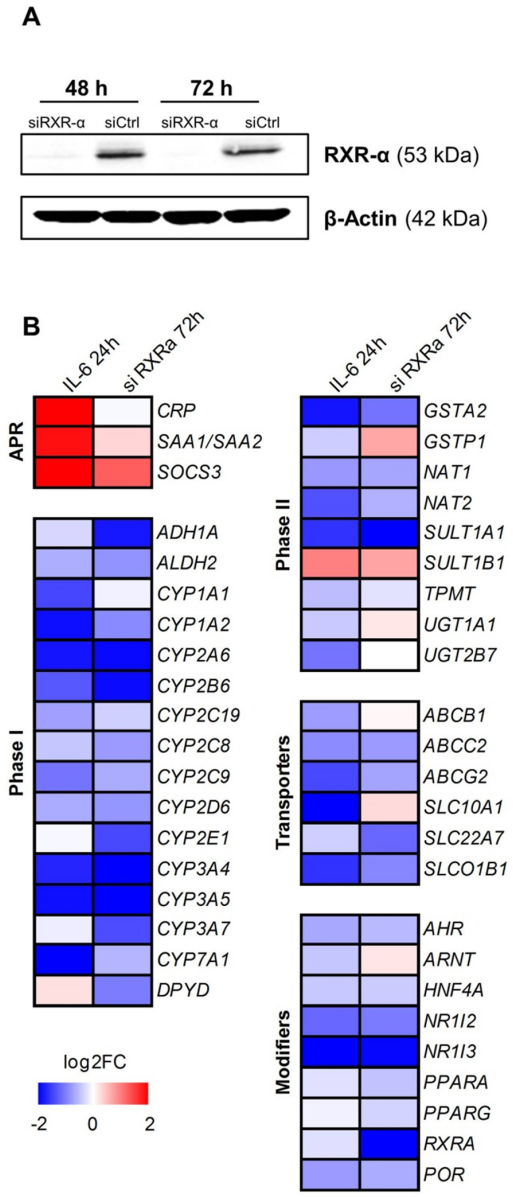

Figure 4.8: Validating the role of RXRα in the downregulation of DMET genes by knock-down experiments. (A) Western blot analysis of RXRα after knock-down in primary human hepatocytes. The western blots of RXRα based on the proteins in primary human hepatocytes 48 h and 72 h after application of siCtrl (control) and siRXRα are displayed here. β-Actin was used as a positive control. (B) Heat map comprising relative changes in gene expression (log2FC) for acute phase response (APR) and DMET genes after IL 6 stimulation (IL-6 vs. control, 24 h) or RXRα knock-down (siRXRα vs. siControl, 72 h). Upregulation is represented in red, downregulation in blue color. Gene expression was normalized to GAPDH (compare 4.6.1). Note that a few genes measured here are not contained in the model in Figure 4.7 (e.g., SULT1A1, SULT2B7).

4.10 Summary and conclusions

A fuzzy logic model was developed to identify regulatory events responsible for the IL-6 induced regulation of DMET gene expression. Adaptation of an optimization routine based on experimental data and a prior knowlege network (Morris *et al.*, 2011) enabled us to calibrate the model. The resulting model suggested that inhibition of the complexes between RXRα and nuclear receptors is the main regulatory event leading to downregulation of DMET genes. This hypothesis was supported by knock-down experiments.

Fuzzy logic modeling avoids the estimation of various kinetic parameters like in ODE modeling, which is often not adequate for larger biological networks (Melas *et al.*, 2011). Furthermore, the species states in a fuzzy logic model do not need to show simple on-off characteristics like in Boolean models. Therefore, the application of fuzzy logic modeling is an approach between ODE modeling and Boolean modeling that can be adequate for elucidating regulatory events behind a biological phenomenon.

The normalization of the gene expression data and the determination of the logic gate values both involve Hill functions. One motivation for the application of Hill functions in CNORfuzzy was that they are adequate for describing protein-protein interactions (Morris *et al.*, 2011). But Hill functions can also describe gene regulations (Santillán, 2008), which are included in our model. Whereas the parameters of the "gate equations" are estimated by CNORfuzzy, the parameters of the Hill functions for data normalizations need to be set in advance. The choice of those parameters is somewhat arbitrary and was dependent on the respective gene. While the chosen modeling approach proved effective for identifying important regulatory events, the biological meaning of the estimated parameters is questionable. Therefore, their values were not taken into account and instead the main hypothesis suggested by the model was tested with further experiments.

The quality of the resulting model depends on the prior knowledge network. Here the optimization produced a satisfactory fit between predictions of the optimized model and experimental data. In contrast, important interactions could also be missing in the PKN resulting in a poor fit. From analysis of the fitting results the modeller might get an idea which interactions need to be added to the PKN. However, the missing interactions could not be obvious, which is why an automatic method for identifying such interactions would be a helpful alternative. For Boolean models such a method has been developed (Eduati *et al.*, 2012), which could be adapted for fuzzy logic models.

Several data sets were combined for model training and thus a "mean" model over these data sets was constructed. A large variability in the gene expression data over different donors could be a problem in this respect. However, in the experimental data donor variability was low and only observed for few genes. The influence of IL-6 on DMET gene expression was mostly similar for all liver donors.

The mechanisms behind inhibition of the RXRα/NR complexes upon IL-6 stimulation remain to be investigated. Further experimentation targeting this question can lead to a refinement of the model and thus to a refined hypothesis for explaining DMET downregulation after IL-6 stimulation.

Chapter 5

The Systems Biology Simulation Core Algorithm

For storing ODE models SBML (see 2.2.4) is the most important format. In order to simulate SBML models computationally, a dedicated algorithm which involves interpretation of SBML as well as following numerical simulation is necessary. In this chapter, which is mainly based on Keller *et al.* (2013), we describe the Systems Biology Simulation Core Algorithm and its implementation in the Systems Biology Simulation Core Library.

5.1 Motivation

Models in SBML and other formats like CellML (Cuellar *et al.*, 2006) can be interpreted differently. If kinetic rate laws are given for the reactions of a model, this model is often interpreted as an ordinary differential equation (ODE) system. In contrast to a conventional ODE system, SBML and CellML models can contain additional structures like events, algebraic rules, and assignment rules in the case of SBML. In order to analyze and simulate such models, a numerical solver library that can be integrated into larger software applications is of great help. The SBML (Hucka *et al.*, 2003; Finney and Hucka, 2003; Finney *et al.*, 2006; Hucka *et al.*, 2008, 2010) and CellML (Cuellar *et al.*, 2006) language specifications describe the interpretation of models in great detail. However, the implementation of a simulation algorithm for such models is far from straightforward.

We focus on a simulation algorithm for models in the SBML format because of two reasons:

1. The SBML community provides a large number of benchmark tests (Keating *et al.*, 2013), which can be easily used to evaluate the performance of a simulation algorithm. The reason for the creation of this SBML Test Suite were the different results of simulation tools for similar SBML models (Bergmann and Sauro, 2008). A similar test suite of this size is not available for CellML.

2. The number of supporting software tools is much larger for SBML than for CellML.

With the Systems Biology Simulation Core Algorithm (SBSCA), we provide a method for precisely interpreting and simulating ODE models given in SBML. We show how to adapt existing numerical integration routines in order to enable simulation of SBML models.

The SBSCA is implemented in the Systems Biology Simulation Core Library (SBSCL), a platform-independent open-source library in Java. With this reference implementation we demonstrate the correctness and usefulness of the algorithm. The SBSCL does not comprise a graphical user interface and is especially dedicated to integration into third-party programs.

The library consists of several ODE solvers as well as an SBML interpreter and is the first simulation library based on JSBML (see 2.2.5). All existing levels and versions of SBML are supported by the SBSCL. Furthermore, the library contains classes for exporting simulation configurations to the SED-ML format (Waltemath *et al.*, 2011b). SED-ML (Simulation Experiment Description Markup Language) is an XML-derived format for description of simulation experiments. It enables to reference the model that was simulated as well as, e.g., the simulation time and the integration routine used for simulation. This facilitates reproduction of simulation experiments.

In order to support other systems biology community formats in the future, the interpretation of an SBML model is strictly separated from the numerical method for simulation of the model. Thus development of an interpretation algorithm for a specific format would be sufficient to enable simulation of models given in that format. The architecture of the SBSCL has been especially designed for easy integration of a CellML module.

5.2 A formal representation of models in systems biology in SBML

Prior to the description of the algorithm for the interpretation of SBML models, we here define the mathematical equations implied by the SBML format. This general description will provide the basis for explaining all steps of the algorithm precisely later.

When we simulate a model, we are interested in the changes of the amounts or concentrations of the species. Whether the following notation of the species' values stands for their amounts or their concentrations, is dependent on the units of the species and can therefore be specified by the model creator.

The structure of a reaction network can be described mathematically by the stoichiometric matrix $\mathbf{N}$ with its rows representing the reacting species $\vec{S}$ and its columns standing for the reactions $\vec{R}$ (see 2.2.1). How the changes of the species' values are determined based on the reaction velocities $\vec{v}$ and the stoichiometric matrix in general has already been shown (Equation (2.1)). We now assume that the velocities of the reactions are dependent on the species $\vec{S}$, the time t, the stoichiometries in the stoichiometric matrix $\mathbf{N}$, a

modulation matrix **W**, and a parameter vector $\vec{p}$. Then we derive the following formula:

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{S} = \mathbf{N}\vec{v}(\vec{S},t,\mathbf{N},\mathbf{W},\vec{p})\,. \tag{5.1}$$

The parameters in $\vec{p}$ can be rate constants, but also other quantities with an influence on the reaction velocities. The modulation matrix in **W** has been defined previously (Liebermeister and Klipp, 2006; Liebermeister *et al.*, 2010). It is of size $|\vec{R}| \times |\vec{S}|$ and stands for the regulatory influences of species on the reactions, e.g., inhibition or stimulation. Stimulation of a reaction by a species is represented by 1, inhibition correspondingly by -1.

In order to derive the predicted value of a species for a certain time point t_T, the ordinary differential equation system (5.1) needs to be integrated within the interval $[t_0, t_T]$:

$$\vec{S} = \int_{t_0}^{t_T} \mathbf{N}\vec{v}(\vec{S},t,\mathbf{N},\mathbf{W},\vec{p})\mathrm{d}t\,, \tag{5.2}$$

For this equation we assume $t_0, t_T \in \mathbb{R}^+$ and $t_0 < t_T$.

This simple case allows to solve Equation (5.2) in a straightforward way by using numerical differential equation solvers (see 2.2.3). The main difficulty when simulating such a differential equation system are nonlinearities in the kinetic equations in the vector function $\vec{v}$, which might lead to stiff differential equation systems. As nonlinear kinetic equations are common in biological reactions, it is usually recommendable to apply a numerical integration routine with step size adaptation to an ODE system in systems biology.

In general the mathematical description of biological network dynamics is more complex than Equation (5.2). As SBML models can contain elements like events and rules (see 2.2.4), an extended formula is necessary for describing these models:

$$\vec{Q} = \int_{t_0}^{t_T} \mathbf{N}\vec{v}(\vec{Q},t,\mathbf{N},\mathbf{W},\vec{p}) + \vec{g}(\vec{Q},t)\mathrm{d}t + \vec{f}_E(\vec{Q},t) + \vec{r}(\vec{Q},t)\,, \tag{5.3}$$

Besides amounts (or concentrations) of reacting species, the vector $\vec{Q}$ of quantities also comprises the sizes of the compartments $\vec{C}$ and the values of all global model parameters $\vec{P}$. Furthermore, SBML models can contain local parameters $\vec{p}$, which influence velocities of reactions. But these local parameters are constant over time and therefore not included in the global parameter vector $\vec{P}$ and in $\vec{Q}$.

All vector function terms might contain a delay function, which is an expression of the form $\mathrm{delay}(e,\tau)$ with $\tau > 0$. This enables to include values of e, which can be simply a variable as well as a complex mathematical expression computed at an ealier time point $t - \tau$. With such delay functions Equation (5.3) is transformed into a delay differential equation (DDE).

In general the changes of some species' values are not given by the product $\mathbf{N}\vec{v}$ in Equation (5.3), but by rate rules (function $\vec{g}(\vec{Q},t)$). These rate rules also enable to define

the change rates of quantities other than species. The species whose change rates are stated in rate rules are not allowed to participate in any reaction, which is why their entries in the stoichiometric matrix **N** are zero for all reactions. Correspondingly, the rate rule function $\vec{g}(\vec{Q},t)$ returns the rates of change for the quantities defined in rate rules and 0 for all other quantities.

Important elements of SBML models are events $\vec{f}_E(\vec{Q},t)$ and assignment rules $\vec{r}(\vec{Q},t)$. Events enable to model sudden changes of quantities. Once a trigger condition is valid, an event can directly change the value of one or more quantities, e.g., set some parameter to a specific value. Assignment rules also define values of quantities directly, but are valid at all times in contrast to events.

Algebraic rules are an alternative to assignment rules in SBML and allow to express conservation relations or other complex interrelations conveniently. They have to evaluate to zero at any time during model simulation. Based on bipartite matching (Hopcroft and Karp, 1973) algebraic rules can often be transformed into assignment rules, which enables their inclusion into the term $\vec{r}(\vec{Q},t)$. The transformation algorithm is explained below and involves solving the algebraic rules by quantities whose values are not set by other constructs (e.g., by assignment rules or reactions).

Biological systems can comprise processes at different time scales, i.e., fast and slow subsystems. In order to solve such systems, separation of the fast and the slow subsystems is necessary. Then differential algebraic equations (DEA), i.e., ODE systems coupled with additional constraints, are created.

5.3 The algorithm for simulation of SBML models

It was demonstrated in the previous section that determining a solution of Equation (5.3) is significantly more complicated than just solving the simple Equation (5.2). In this section we describe the necessary steps to solve the systems defined in Equation (5.3) in detail. For an efficient computation of this solution multiple preprocessing steps are required involving, e.g., the conversion of algebraic rules into assignment rules. Furthermore, repeated computation of intermediate results should be avoided, if possible.

5.3.1 Initialization

The first step of the simulation is setting the values of species, parameters, and compartments to the initial values defined in the model. In order to avoid excessive reevaluation of mathematical expressions, all rate laws of the reactions, assignment rules, transformed algebraic rules (see the respective transformation algorithm below), initial assignments, event assignments, rate rules, and function definitions are integrated into a single directed acyclic syntax graph. In this graph the abstract syntax trees representing all those elements are merged such that similar elements are only contained once. Therefore, evaluation of this acyclic syntax graph is much faster than evaluating each syntax graph

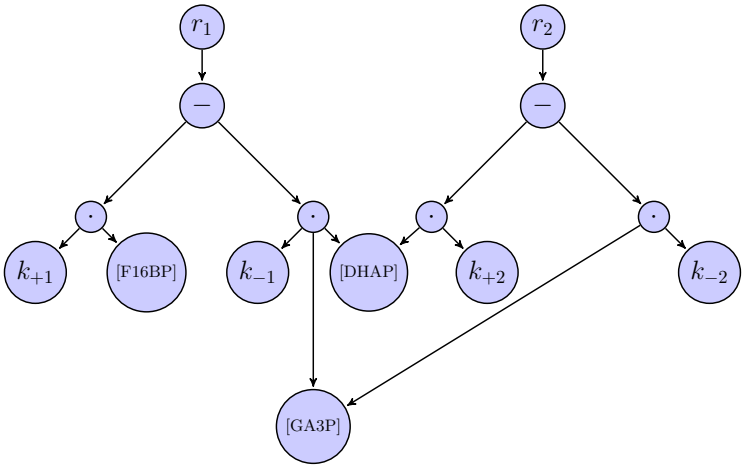

Figure 5.1: Creation of an abstract syntax graph for a small model. This figure shows the abstract syntax graph of kinetic equations from an example model with the following reactions:

R_1 : F1,6BP $\rightleftharpoons$ DHAP + GA3P
R_2 : DHAP $\rightleftharpoons$ GA3P

These reactions belong to the glycolysis. The involved molecules are fructose 1,6-bisphosphate (F1,6BP), dihydroxyacetone phosphate (DHAP), and glyceraldehyde 3-phosphate (GA3P). With SBMLsqueezer (Dräger *et al.*, 2008) the following mass action kinetics were created:

$v_{R_1} = k_{+1} \cdot [\text{F1,6BP}] - k_{-1} \cdot [\text{DHAP}] \cdot [\text{GA3P}]$
$v_{R_2} = k_{+2} \cdot [\text{DHAP}] - k_{-2} \cdot [\text{GA3P}]$

It is obvious from the figure that the nodes for [DHAP] and [GA3P] are only contained once in the syntax graph, but connected to more than one multiplication node. The syntax graph shown here is thus not a tree.

individually. As the evaluation of mathematical syntax graphs is necessary at each simulation step, using this single syntax graph significantly decreases simulation time. In Figure 5.1 an example for such a syntax graph is given.

In the following initialization step the initial assignments and the assignment rules (including transformed algebraic rules) are processed. In contrast to assignment rules, initial assignments are only relevant for time point 0. At this specific time point both constructs are equivalent and processing of them yields initial values for the respective variables.

5.3.2 Solving algebraic rules

In order to facilitate straightforward processing of algebraic rules, the algorithm transforms them to assignment rules. After that, all algebraic rules are exactly treated like assignment rules.

Every algebraic rule should contain one or more variables whose values are not defined elsewhere in the model (e.g., by assignment rules). For each algebraic rule such a variable must be determined in order to enable conversion of the rule to an assignment rule. The first step of the transformation algorithm involves creating a bipartite graph as described in the SBML specifications (Finney *et al.*, 2006; Hucka *et al.*, 2008, 2010). One set of vertices in this graph comprises the variables of the model (variable vertices), whereas the other set of vertices (equation vertices) consists of the equations (i.e., assignment rules, rate rules, and algebraic rules) as well as of species occuring in a kinetic law of at least one reaction. The edges of the graph now connect

- equation vertices representing species to variable vertices representing the same species,
- vertices representing rate rules or assignment rules to vertices representing the variables defined by the rules, and
- vertices representing algebraic rules to all variable vertices representing the variables contained in the respective rule.

With the algorithm by Hopcroft and Karp (Hopcroft and Karp, 1973) a maximal matching between the variable vertices and the equation vertices is now determined. In this maximal matching each vertex is connected to not more than one other vertex and the number of connections (i.e., edges used for the matching) is maximal.

We begin with an initial greedy matching: Starting from an arbitrary equation vertex this involves creating paths which only contain edges to vertices that have not been included in any path so far. Once an edge has been added to a path, the respective target vertex cannot be visited any more. When a path cannot be continued via edges to unvisited vertices, the greedy algorithm tries to start from another equation vertex that has not been visited. If such a vertex does not exist any more, the algorithm stops. In all the paths the edges can now be alternatingly seen as matching edges (i.e., edges representing a matching between the connected vertices) or non-matching edges.

The algorithm by Hopcroft and Karp now tries to increase this initial matching, which is usually not maximal, by trying to augment paths. An augmenting path is a path extended by previously unmatched start and end nodes. The resulting path contains one more matching edge than the original path. When no more augmenting paths can be found, the algorithm stops. The result is by definition a maximal matching.

If some equation vertex remains unmatched after the algorithm by Hopcroft and Karp, the model is overdetermined and not considered a valid SBML model (as declared in the SBML specifications (Finney *et al.*, 2006; Hucka *et al.*, 2008, 2010)). We now assume

the model is a valid SBML model. Then the derived maximal matching gives a unique variable vertex for each algebraic rule. Every algebraic rule is now solved by the respective variable leading to an assignment rule. The described algorithm is summarized in Figure 5.2 as a flow chart.

5.3.3 Event handling

An SBML event is a list of assignments that is executed once a trigger condition switches from *false* to *true*. SBML also enables the definition of a delay, which can lead to execution of the event assignments at a later time point. In SBML Level 3 Version 1 the processing of events is even more complex than for previous levels and versions: In earlier versions only the event trigger and its delay needed to be defined. In contrast to that, Level 3 Version 1 contains a few new language elements, which have a strong influence on event handling: Whereas the order in which different events at a certain time point are processed could be chosen by the programmer in SBML Level 2, in Level 3 Version 1 this order is stated by the event's priority element. Therefore, the correct sequence of simultaneous events is now crucial in event handling. Furthermore, an event can now be canceled within the time interval from trigger activation to the actual event execution. Events for which such a cancellation is possible are called *nonpersistent*.

The events to be executed at some time step can be divided into two subsets:

- events with triggers activated at the current time and without delay, and
- events with delay triggered at some previous time point.

For every element of the resulting set of events its priority needs to be evaluated. Then one of the events with highest priority is randomly chosen for execution. The execution of an event can change the priorities or the trigger conditions of the events that are still to be executed. This means that for nonpersistent events their triggers have to be reevaluated. Furthermore, the priorities of all events waiting for execution are to be recomputed. Then the event with highest priority is processed next. Execution and following reprocessing of other events is repeated until no further event is left for execution. The slightly simplified algorithm for event processing at a certain time point is shown in Figure 5.3.

5.3.4 Time step adaptation considering events and the calculation of derivatives

The time when events are triggered should be calculated precisely in order to obtain exact results in the numerical integration process. An event could, e.g., be triggered at time t_τ, which is between the integration time points $t_{\tau-1}$ and $t_{\tau+1}$. If events are only processed at time points $t_{\tau-1}$ and $t_{\tau+1}$, the trigger condition can possibly not be evaluated

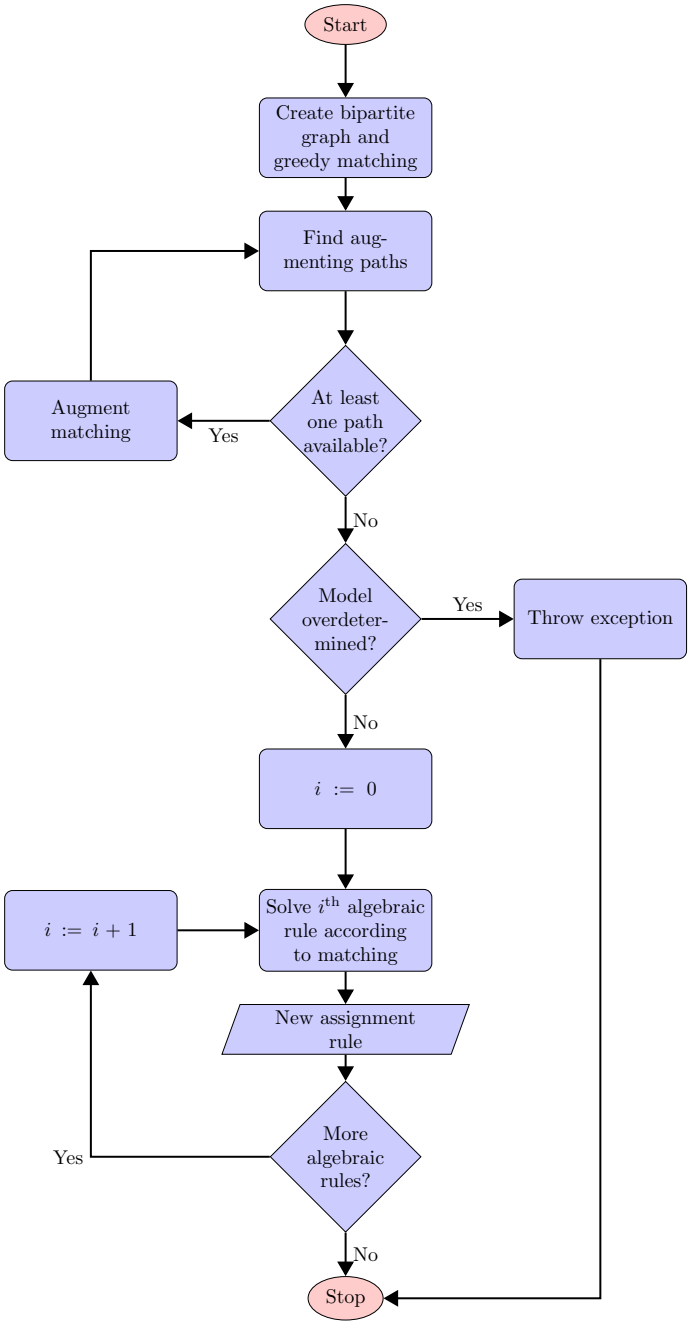

Figure 5.2: Algorithm for transforming algebraic rules to assignment rules. In the first major step a maximal matching between the equations and the variables of a model is determined. This involves constructing a bipartite graph, computing an initial greedy matching, and afterwards constructing a maximal matching. A maximal matching is derived by augmenting paths and thus extending the matching. The matching is maximal, if no more augmenting paths can be found. All equation vertices in the maximal matching must be matched to a variable vertex. Otherwise, the model is overdetermined and an exception is thrown by the algorithm. In the case of a valid model an assignment rule is generated for each algebraic rule. The left-hand side of such a transformed rule comprises the variable the respective algebraic rule has been matched to.

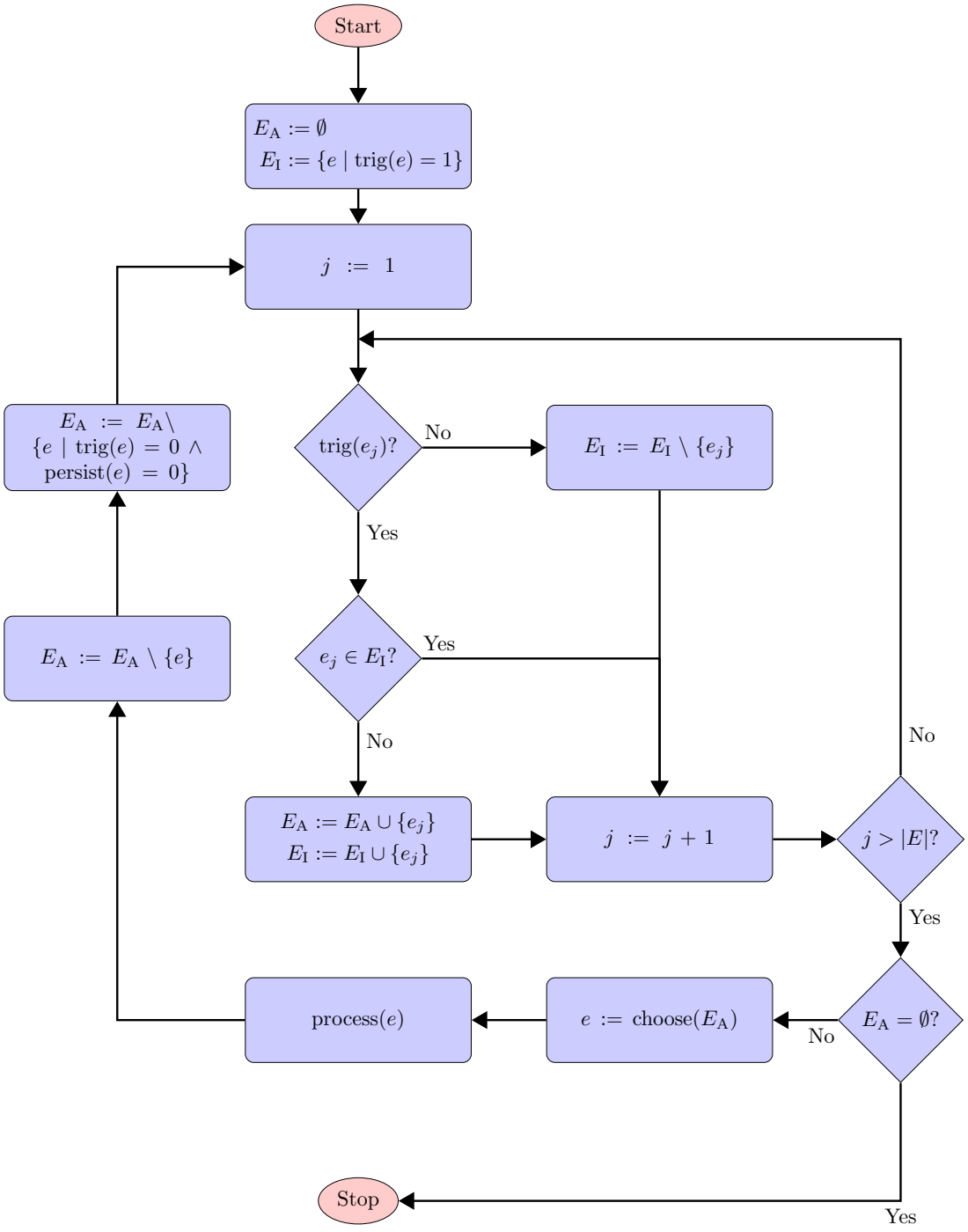

Figure 5.3: (Caption next page.)

Figure 5.3: (*Previous page.*) Processing of events: simplified algorithm (handling of delayed events omitted). We define E as the set of all events in a model and E_I as the set of events whose trigger conditions have already been evaluated to *true* in previous time steps. Events within E_I are inactive. Their triggers need to switch from *true* to *false* before they can become active again. E_A is defined as the subset of E containing events with triggers switching from *false* to *true* at the current time point t. At the beginning E_A is empty. A persistent event is removed from E_A once all of its assignments have been evaluated. In contrast to that, a nonpersistent event is also removed from E_A when its trigger condition changes to *false* when processing other events. The function trig(e) returns 1, if the trigger condition of event $e \in E$ is satisfied, and 0 otherwise. Correspondingly, the function persist(e) returns 1, if event e is persistent, or 0, if e is nonpersistent. The trigger conditions of active events $e_\mathrm{a} \in E_\mathrm{A}$ that are not persistent are recomputed in each iteration of the algorithm. If this leads to a change of the trigger from *true* (1) to *false* (0), the event is removed from E_A. In the next step the triggers of all events are evaluated. If a trigger changes from *false* to *true*, an event becomes active and is added to E_A. An event with its trigger changed from *true* to *false* is not inactive any more and removed from E_I. After the processing of all event triggers, one event e with highest priority from E_A is chosen for execution by the function choose(E_A). This choice is random, if there are several events with the highest priority. Then the assignments of the chosen event are evaluated. After that, the triggers of all events in E are evaluated again. The algorithm stops once the set E_A is empty.

to *true* at any of these time points. Therefore, a numerical integration method with step size adaptation is necessary in order to ensure event execution at the correct time points. If events occur, Rosenbrock's method (Press *et al.*, 1993) can adapt its step size h (see Figure 5.4). Given the current vector $\vec{Q}$ and a certain time interval $[t_{\tau-1}, t_\tau]$, Rosenbrock's method determines the new value of $\vec{Q}$ at a time point $t_{\tau-1}+h$, with $h > 0$. If the error tolerance is not satisfactory, h is reduced and the new value of $\vec{Q}$ is computed at the changed time point $t_{\tau-1}+h$.

After a successful step, the events as well as the assignment rules are processed at time point $t_{\tau-1}+h$, which can lead to a change in $\vec{Q}$. In this case, the adaptive step size is reduced by setting h to $h/10$ and $\vec{Q}$ is calculated again at time point $t_{\tau-1}+h$. The step size adaptation is repeated until either the minimum step size is reached or the event and assignment rule processing does not change $\vec{Q}$ anymore. This algorithm leads to a precise determination of the correct time for event execution.

Given the values $\vec{Q}$ at time point t, the current vector of derivatives $\dot{\vec{Q}}$ is determined as follows (also see Figure 5.5): In the first step the rate rules are processed: $\dot{\vec{Q}} = \vec{g}(\vec{Q}, t)$. The values of vector elements with no rate rule defined are 0 here. Next the velocity v_i of each reaction R_i is determined based on the abstract syntax graph (see 5.3.1). The derivatives of all species participating in R_i need to be updated after the computation of v_i.

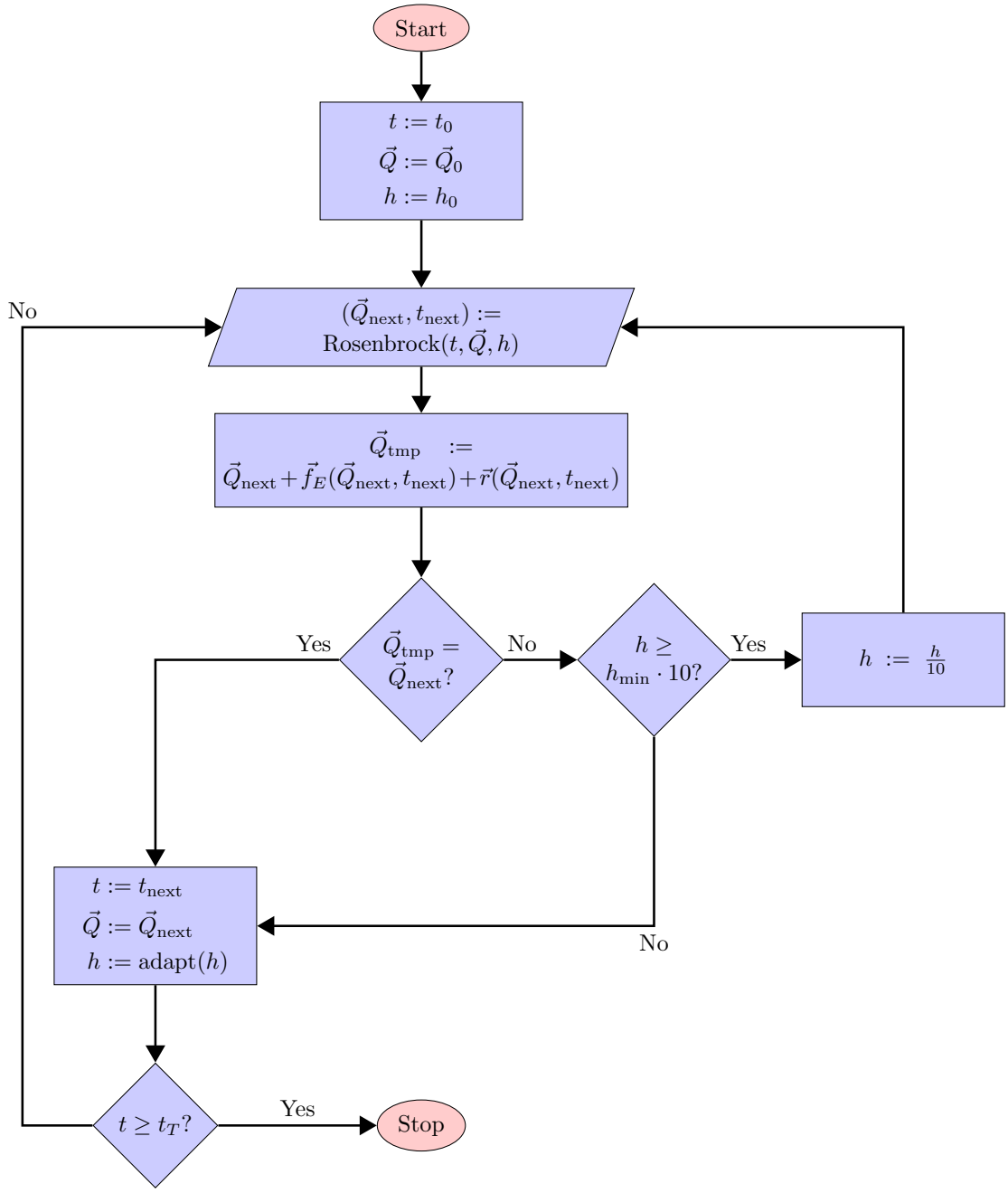

Figure 5.4: Refined step size adaptation for events. When simulating from time t_0 to t_T the Rosenbrock solver continuously tries to increase time t by the current adaptive step size h and calculates a new vector of quantities $\vec{Q}_{\text{next}}$ for time point $t+h$. The step size is adapted by the routine, if the error tolerance is not satisfactory. After a step with an acceptable error tolerance, the events and rules of the model are processed. If this changes $\vec{Q}$, h is decreased and the Rosenbrock solver calculates another $\vec{Q}_{\text{next}}$ for the new time point $t+h$. This process is repeated until either the minimum step size h_{min} has been reached or $\vec{Q}_{\text{next}}$ is not changed by event and assignment rule processing any more. h_{min} thus determines the precision of the event processing. The adapt function is defined by Rosenbrock's method (Press *et al.*, 1993) and not explained in detail here.

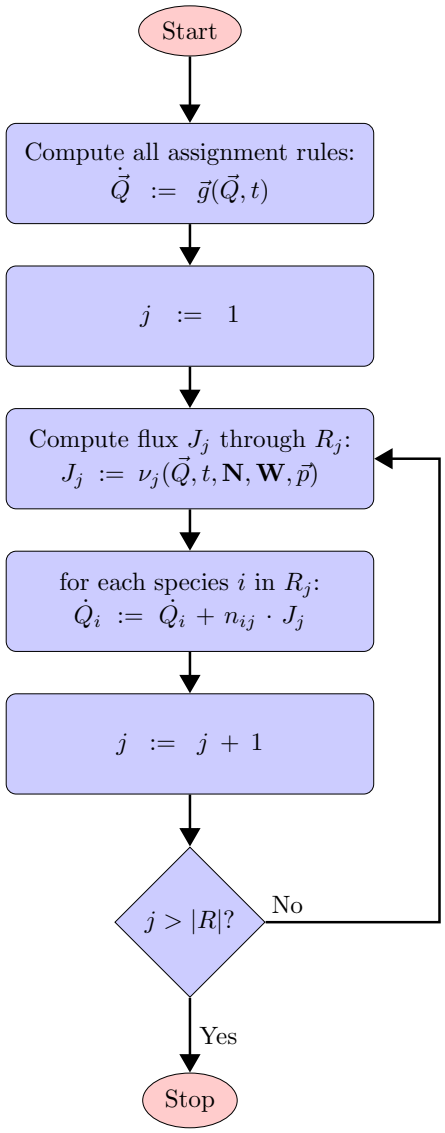

Figure 5.5: Calculation of the derivatives at a specific point in time. The derivatives of all quantities are stored in the vector $\dot{\vec{Q}}$, which is set to the null vector $\vec{0}$ at the beginning. After that, the rate rules of the model are processed by applying the function $\vec{g}(\vec{Q},t)$, which can change some elements of $\dot{\vec{Q}}$. The next step involves for every reaction R_j the computation of its velocity J_j. For each species (with index i) which participates in R_j its derivative is updated by adding the product of the stoichiometry n_{ij} and J_j. For the sake of simplicity, the stoichiometries n_{ij} in the matrix $\mathbf{N}$ are assumed to be constant here. However, these values can also be variable. SBML provided `StoichiometryMath` elements for a direct computation of the stoichiometries before Level 3. In Level 3, stoichiometries can be changed by assignment rules. If stoichiometries are variable, the values for n_{ij} have to be recomputed in each simulation step.

5.3.5 Processing models with fast and slow subsystems

If the model to simulate comprises fast and slow subsystems, the SBSCA processes both of these systems separately. For the system containing the fast reactions we assume that it always reaches a steady state (i.e., a state in which the amounts of the species do not significantly change when continuing simulation) within an extremely small time frame. This quasi-steady state assumption was suggested in the SBML specification (Hucka *et al.*, 2010). The algorithm first computes a steady state for the fast reactions and considers the reactions within the slow subsystem as inactive at this point. This steady state computation involves simulation until all species values do not significantly change any more. The resulting values of the species are then the simulated values for the initial time point. Afterwards the SBSCA takes a simulation step with the fast reactions being inactive and the slow reactions being active. Then another steady state computation based on the fast reactions follows, which represents the adaptation of the fast subsystem to changes within a very small time frame. In this way simulation of the slow subsystem and steady state computation with the fast subsystem are alternatingly conducted until the end time is reached.

For the Rosenbrock solver the processing of fast reactions is similar to that of events and thus more accurate than for the other solvers. After each successful simulation step involving the slow reactions, a steady state is computed. If this steady state computation causes a change greater than a certain threshold in the amount of some species and the step size h is not below a certain value, the step size is reduced and simulation is repeated (compare Figure 5.4). Both the threshold and the value for the minimum step size for considering changes by the fast reactions were fixed to 10^{-3}. In principle this could lead to inaccurate simulation results. However, the benchmark tests, which will be covered later in this chapter, were all passed with the chosen settings.

5.4 Implementation of the algorithm in the Systems Biology Simulation Core Library

The Systems Biology Simulation Core Library written in Java comprises an implementation of the the Systems Biology Simulation Core Algorithm. The software architecture of this library is shown in Figure 5.6. It provides an extensible numerical backend, which can be integrated into programs for systems biology research. The SBML interpretation algorithm uses data structures provided by JSBML (Dräger *et al.*, 2008).

As the library contains interfaces for differential equation systems, which are not specific for a particular type of model (e.g., SBML), support for other community standards, such as CellML (Lloyd *et al.*, 2004), can be easily implemented. To this end, an interpreter for such a model format is sufficient. An instance of this interpreter can then be passed to any available solver.

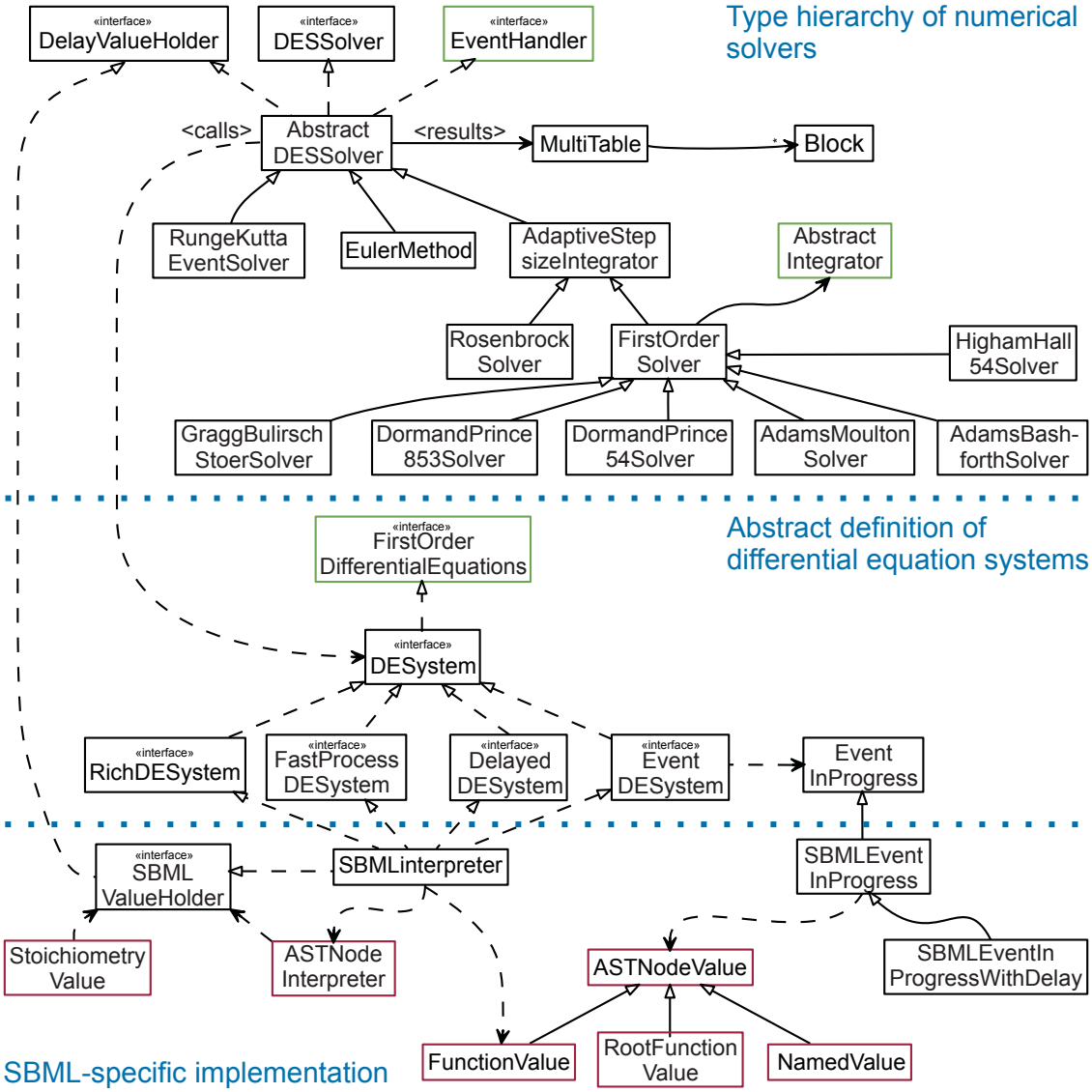

Figure 5.6: Architecture of the Systems Biology Simulation Core Library (simplified). In the library numerical methods are strictly separated from differential equation systems. The upper part of the figure shows the type hierarchy of all currently included numerical integration routines. In the middle part the interfaces defining several special types of the differential equations to be solved are displayed. All these interfaces are implemented by the class `SBMLinterpreter` (bottom part), which is the main class for the interpretation of an SBML model. In order to support other data formats, a similar interpreter needs to be implemented.

5.4.1 Solver classes

The classes for all solvers are derived from the abstract class `AbstractDESSolver` (Figure 5.6). With the help of wrapper classes we included several solvers of the Apache Commons Math library (version 3.0, Apache Software Foundation (2013)). In addition, the library provides an implementation of Euler's method (Press *et al.*, 1993), the explicit fourth order Runge-Kutta method (Press *et al.*, 1993) as well as of Rosenbrock's method. The implementation of Rosenbrock's method is a modification of a previously developed class (Kotcon *et al.*, 2011). This implementation was adjusted to enable a very precise timing of the events (see 5.3.4). The results of a simulation with any of the numerical integration methods are stored in objects of the class `MultiTable`, which contains `Block` data structures. In one of those blocks the quantity values are stored, whereas in another block the velocities of the reactions are saved.

5.4.2 SBML interpretation

Interpretation of SBML models can be divided into evaluation of events and rules, computation of stoichiometric information, and computation of the current values for all model components (e.g., species and compartments). The class `SBMLinterpreter` returns the current set of time-derivatives of the variables for a given state of the ODE system (i.e., the current values of its variables). `SBMLinterpreter` is connected to a MathML interpreter for the mathematical expressions contained in kinetic laws, rules and events (`ASTNodeInterpreter`). The respective nodes are all contained in the abstract syntax graph created at the beginning of the simulation (see 5.3.1). As the values of the nodes depend on the current state of the ODE system, they have to be recalculated, if the state of the system has changed.

Representation of rules

Rules can be seen as events occuring at every time point. Therefore, they are processed similar like events. Transforming algebraic rules to assignment rules involves for every object of type `AlgebraicRule` the creation of a new `AssignmentRule` object with the help of bipartite matching (see 5.3.2). These `AssignmentRule` objects are only created for the simulation purpose and do not influence the content of the SBML model.

Representation of events

Events are stored in the `SBMLinterpreter` via an array containing one object of `EventInProgress` for every event of the model. These events can be divided into events with and without delays (see 5.3.3). For events without delay the event assignments are usually executed at the time point when the event has been triggered. The only exception to this is the cancellation of an event by other events. Events with delay can produce

multiple additional assignments within the time interval from trigger time to execution time. The class `SBMLEventInProgressWithDelay`, which is a subclass of the general class `SBMLEventInProgress`, saves the time points at which a delayed event is to be executed.

When processing events with priority, the events having the highest priority are stored in a list until one of them is selected for execution. For the organization of such priority queues a binary max heap data structure could be the method of choice. The largest value in the heap is always contained in the root. Once this value is removed, the heap is reorganized such that the next largest value is stored in the root. However, the execution of an event can have an effect on the priority of the remaining events (see 5.3.3). If several priorities change simultaneously, the standard method for storing the heap is not applicable any more. Therefore, we did not use complex data structures for the processing of events with priority.

Representation of function definitions

SBML Level 2 and Level 3 enable to define functions, which can be used in kinetic laws of reactions. Such function definitions comprise a lambda expression optionally containing a list of arguments and the mathematical expression of the function. Function definitions are included into the abstract syntax graph during the initialization of the algorithm and are represented by objects of the class `FunctionValue`. A syntax graph node with a function definition is evaluated in several steps:

- The arguments are evaluated.
- The values of the arguments are given to the corresponding argument nodes in the graph.
- The mathematical expression of the function is evaluated based on the current arguments.

Different function definitions can have the same identifiers for arguments, as the identifiers of arguments are only valid within the corresponding function definition. Therefore, naming conflicts in the abstract syntax graph have to be prevented.

The stoichiometry math construct

The `StoichiometryMath` construct enables to change the reaction's stoichiometry during simulation. However, it should be noted that in common biochemical reaction systems the stoichiometry is constant. In SBML Level 3 Version 1 the stoichiometry of a reaction can be set directly by addressing the identifier of a `SpeciesReference` within rules or events. A `SpeciesReference` object comprises the species taking part in a reaction and its stoichiometry. `SBMLinterpreter` stores possibly changing stoichiometries (i.e., stoichiometries with a `StoichiometryMath` construct or occuring in events

or rules) during initialization. The corresponding abstract syntax nodes are reevaluated once the stoichiometries have to be calculated.

Constraints

Constraints in an SBML model introduce assumptions about model behavior. The abstract syntax graph of each constraint is evaluated at every time step. If a constraint is violated, `SBMLinterpreter` generates an instance of `ConstraintEvent`, which is afterwards processed by the `ConstraintListener` class. The user is informed about the constraint violation including the simulation time point and the constraint message via the Java `Logger`.

5.4.3 SED-ML support

The standard MIASE (Minimum Information About a Simulation Experiment) (Waltemath *et al.*, 2011a) involves minimum information that should be given to adequately describe a simulation of a model. In order to support this standard, the SBSCL comprises an interpreter of SED-ML files (Waltemath *et al.*, 2011b). The user can thus store the details about a specific simulation, which facilitates reproducing the simulation results (compare 5.1). Furthermore, a simulation can be directly started by loading a SED-ML file and passing it to the SED-ML interpreter. The solver for simulation is specified in the SED-ML by its KiSAO (Kinetic Simulation Algorithm Ontology) term (Courtot *et al.*, 2011). This eases execution of SED-ML files.

5.4.4 Points of Control

The default settings for simulation of SBML models comprise the Rosenbrock solver with an absolute error tolerance of 10^{-12} and a relative error tolerance of 10^{-6}. With this setup we were able to simulate most models used for testing the SBSCL. The Rosenbrock solver with its adaptive step size is the numerical routine in the library dedicated for simulating stiff ODE systems as well as for precisely timing events in a model (see 5.3.4). If the user chooses another solver for integration, this can lead to a lower simulation time due to, e.g., the lacking of a step size adaptation. However, it is then possible that the model cannot be simulated accurately any more. The SBML specifications state that model simulation is always started at time point 0.0. As the SBSCL is not limited to SBML, other start times of the simulation are also accepted. The end time of the simulation can also be specified by the user. If the relative and absolute error tolerance is modified, this can influence the accuracy of the simulation and the computation time. A higher accuracy usually comes with a higher computation time, and vice versa.

5.5 Benchmark and application to published models

An Intel Core i5 CPU with 3.33 GHz and 4 GB RAM was used with Microsoft Windows 7 (Version 6.1.7600) as operating system and Java Virtual Machine version 1.6.0_25 for testing the running time of the Systems Biology Simulation Core Library.

Furthermore, the SBSCL was successfully tested under Linux (Ubuntu version 10.4) and Mac OS X (versions 10.6.8 and 10.8.2).

5.5.1 Application to the models of the SBML Test Suite

All of the models from SBML Test Suite version 2.3.2 (Keating *et al.*, 2013) were first simulated using the following settings:

- Rosenbrock solver as integration routine,
- 10^{-6} as relative error tolerance, and
- 10^{-12} as absolute error tolerance.

The relative tolerance had to be set to 10^{-8} for six models (numbers 863, 882, 893, 994, 1109, and 1121) in order to simulate them accurately. Precise simulation for three other models (numbers 872, 987, 1052) was only possible when setting the relative tolerance to 10^{-12} and the absolute tolerance to 10^{-14}.

Table 5.1 shows the total running times for simulation of the SBML Test Suite models. The simulation of all models together is possible within seconds. Therefore, the simulation of one SBML model takes only milliseconds on average with regular desktop computers.

For the models in SBML Level 3 Version 1 the total simulation time is considerably higher than for the models in other SBML levels and versions. This is due to the fact that the test suite comprises some models of this version whose simulation requires the processing of a large number of events. The simulation of model number 966 of the SBML Test Suite, which is only available in SBML Level 3 Version 1, takes 20 s, because 23 events need to be processed. The triggers of two events change from *false* to *true* every 10^{-2} time units, while the end time for simulation is 1,000 time units. The two events thus need to be evaluated 100,000 times during a simulation, which also involves their precise timing. The simulation of model number 966 consumes over 50 % of the total simulation time for the models in SBML Level 3 Version 1.

5.5.2 Application to the models of the BioModels Database

The Systems Biology Simulation Core Library was successfully validated with the SBML Test Suite. However, the models contained there are not explaining biological behavior,

Level	Version	Models	Correct simulations	Total running time (in s)
1	2	252	252	2.9
2	1	885	885	6.8
2	2	1,041	1,041	6.8
2	3	1,041	1,041	6.8
2	4	1,043	1,043	6.8
3	1	1,077	1,077	38.5

Table 5.1: Simulation of the models from the SBML Test Suite using the Rosenbrock solver. This table shows for all SBML levels and versions the number of tested models and the total running times for the simulation of all models. The time for reading the SBML file is not included in the running time. Therefore, the measured running time only comprises the CPU time needed for model simulation.

but were created for testing purposes. Therefore, testing the SBSCL with realistic biochemical models is necessary in order to show its usefulness. The BioModels Database (Le Novère *et al.*, 2006; Li *et al.*, 2010) comprises a collection of published and curated SBML models (see 3.4.3). For these models it currently contains neither reference data nor any settings for the numerical computation (such as, e.g., the step size and the end time). But for most of the models pre-computed plots of time courses are provided. While the BioModels Database is not adequate for benchmark tests, it can still be used to check whether the SBSCL is able to simulate published models containing diverse features.

The 424 curated models from the BioModels Database (release 23, October 2012) were simulated with identical settings, which was previously suggested (Bergmann and Sauro, 2008). Our settings were:

- time interval $[0, 10]$,
- Rosenbrock solver as integration routine,
- 10^{-6} as relative error tolerance,
- 10^{-12} as absolute error tolerance, and
- a step size of 0.01 time units.

The absolute tolerance had to be changed to 10^{-10} in order to allow an accurate simulation for the models number 234 (Tham *et al.*, 2008) and number 339 (Wajima *et al.*, 2009) from the BioModels Database.

Then the SBSCL was able to solve all curated models without any errors. This shows the reliability of the library when simulating biochemical models given in SBML.

5.5.3 Test with two specific models

Next we select two models with diverse features from the BioModels Database in order to show the capabilities of the SBSCL: model number 206 (Wolf *et al.*, 2000) and model number 390 (Arnold and Nikoloski, 2011).

The model by Wolf *et al.* describes glycolytic oscillations observed in yeast cells. It contains eleven reactions and nine reactive species. The simulated time courses for the concentrations of 3-phosphogylcerate, ATP, glucose, glyceraldhyde 3-phosphate, and NAD^+ are shown in Figure 5.7. After approximately 15 s of only small concentration changes, the concentrations of all metabolites start to oscillate rhythmically. The dynamics of selected reaction velocities over time are displayed in Figure 5.7B.

Arnold and Nikoloski compared several models describing the Calvin-Benson cycle and created a consensus model from them (Arnold and Nikoloski, 2011). This model contains eleven species, six reactions, and one assignment rule. All kinetic equations in the model comprise the calling of function definitions. The simulation results for the species ribulose-1,5-bisphosphate, ATP, and ADP are shown in Figure 5.8. The SBSCL can again reproduce the figures provided by the BioModels Database.

5.6 Comparison to existing simulation implementations for SBML

Several other tools enable simulation of SBML models and are listed in the SBML software matrix (Bergmann *et al.*, 2012). From those programs we chose those fulfilling the following criteria for comparison with our library:

- The last update of the tool was released after the final release of the SBML Level 3 Version 1 Core specification, i.e., after October 6th 2010.
- The program supports SBML Level 3.
- The software is open-source.
- The program is not dependent on commercial products not freely available (such as MATLAB or Mathematica).

The following tools were selected:

- BioUML (Kolpakov *et al.*, 2011),
- COPASI (Hoops *et al.*, 2006),
- iBioSim (Myers *et al.*, 2009),
- JSim (Raymond *et al.*, 2003),

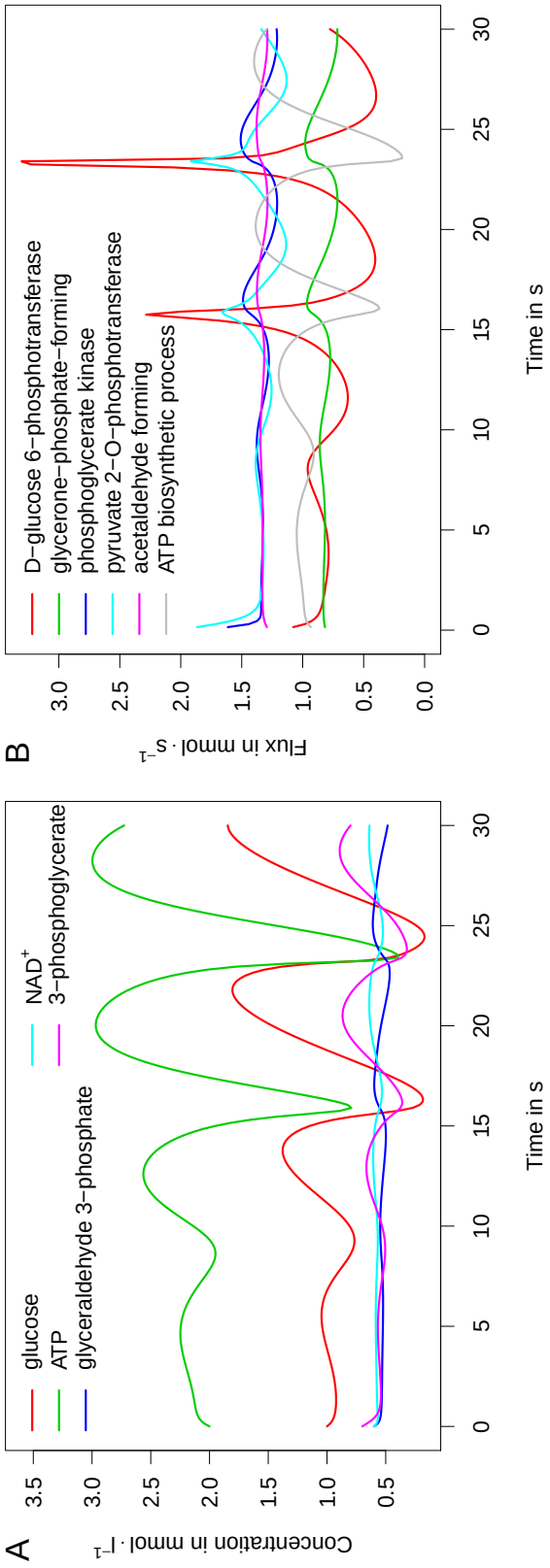

Figure 5.7: Simulation of glycolytic oscillations. The Systems Biology Simulation Core Library was used for simulation of model number 206 from the BioModels Database (Le Novère *et al.*, 2006; Li *et al.*, 2010; Wolf *et al.*, 2000), which describes oscillations during glycolysis in yeast. Part (A) of the figure shows the concentration changes of the intracellular metabolites 3-phosphoglycerate, ATP, glucose, glyceraldehyde 3-phosphate (GA3P), and NAD^+ in the time interval $[0, 30]$ seconds. In part (B) the changes of the velocities (fluxes) over time are displayed for the reactions D-glucose 6-phosphotransferase, glycerone-phosphate-forming, phosphoglycerate kinase, pyruvate 2-O-phosphotransferase, acetaldehyde forming, and ATP biosynthetic process. Simulation was conducted using the Adams-Moulton solver (Hairer *et al.*, 2000) (KiSAO term 280) with 200 integration steps, an absolute error tolerance of 10^{-10} and a relative error tolerance of 10^{-5}. For an accurate simulation a numerical integration method with step size adaptation is necessary. The Adams-Moulton solver, which is not appropriate for some stiff differential equation systems, is able to produce simulation results that are similar to those provided by the BioModels Database.

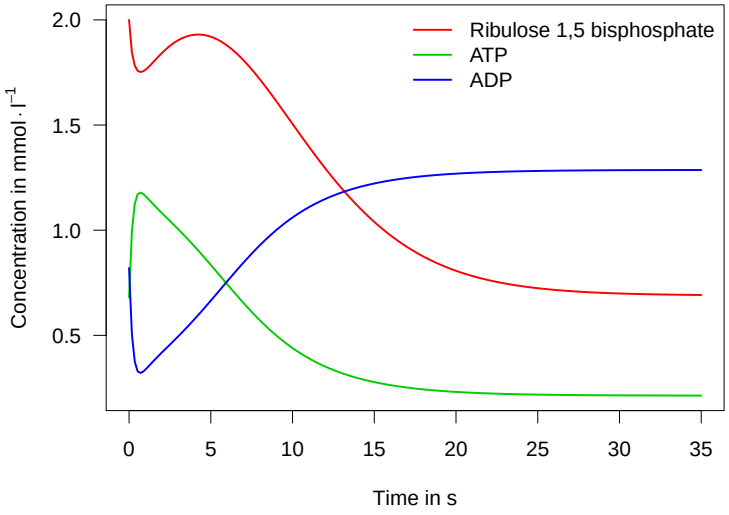

Figure 5.8: Simulation of the Calvin-Benson cycle. The Systems Biology Simulation Core Library was used to solve model number 390 from the BioModels Database (Le Novère *et al.*, 2006; Li *et al.*, 2010; Arnold and Nikoloski, 2011). The concentration dynamics of ribulose 1,5-bisphosphate, which is a key metabolite for CO_2 fixation in the reaction catalyzed by ribulose-1,5-bisphosphate carboxylase oxygenase (RuBisCO), as well as those of ATP and ADP during the first 35 s of the photosynthesis are shown in the figure. Simulation was possible using Euler's method (KiSAO term 30, Press *et al.* (1993)) with 200 integration steps.

- LibSBMLSim (Takizawa *et al.*, 2013), and
- VCell (Moraru *et al.*, 2008; Resasco *et al.*, 2012).

The comparison of these programs with the most recent versions is given in Table 5.2. Benchmark tests with respect to the SBML Test Suite (Keating *et al.*, 2013) are also summarized for most of the compared SBML simulation tools in the continuously updated SBML Test Suite Database (Bergmann, 2013).

Not all the compared programs have their main focus on solving ODE systems. iBioSim, e.g., is especially dedicated to the stochastic analysis of SBML (Madsen *et al.*, 2012). For some tools, such as VCell or COPASI, SBML is not the native format.

Direct access to their application programming interfaces (API) is provided by most programs. COPASI, LibSBMLSim, and the Systems Biology Simulation Core Library are specially designed for the use as a solver backend. iBioSim can be executed via a script, e.g., for batch processing of several models (illustrated by the checkmark in brackets in the table).

As of January 2015 only two other tools pass the entire test suite for all SBML levels and versions: BioUML, which is a program for modeling, simulation, and parameter fitting, and iBioSim. The simulation library LibSBMLSim, which is written in C, only supports models given in SBML Level 2 Version 4 and SBML Level 3 (illustrated by the

checkmark in brackets in the table). COPASI is widely used due to its low running time. However, algebraic rules and fast reactions are not supported by COPASI, whereas events are generally supported, but this support does not involve all of the current constructs (also illustrated by the checkmark in brackets).

Therefore, the Systems Biology Simulation Core Library is the only API simulation library supporting all SBML elements.

5.7 Limitations of the algorithm and the library

The precise timing of events during model simulation is one key part of the derived algorithm. This was achieved by modifying the Rosenbrock solver. However, the precise event timing comes with a significant increase in running time when events are triggered repeatedly within small time intervals, e.g., every 10^{-3} time units. This behavior can be observed in the BioModels Database model number 408 (Hettling and van Beek, 2011), which comprises three events. A solver other than Rosenbrock can be chosen when the precise timing of events is not important and the ODE system is not stiff.

Rosenbrock's method was specially designed for solving stiff ODE systems and is therefore the method of choice in the Systems Biology Simulation Core Library for solving such systems. However, our experiments indicate that the Rosenbrock solver is sometimes inefficient for solving non-stiff ODE systems compared to other solvers. For large models this can cause an increased simulation time. An example for this phenomenon is the simulation of model number 235 of the BioModels Database, which comprises 622 species, that participate in 778 reactions and are distributed across three compartments (Kühn *et al.*, 2009). Changing the relative and absolute tolerance is sometimes helpful, but for some systems the running time of Rosenbrock's method is still limited. A similar behavior can be seen for some other integration methods: The Runge-Kutta-Fehlberg method (Fehlberg, 1970) (KiSAO term 86) included in iBioSim also shows a high running time concerning the BioModels Database model number 235.

Some ODE solvers are more advanced than those by Runge-Kutta-Fehlberg and Rosenbrock. CVODE from SUNDIALS (Hindmarsh *et al.*, 2005) can, e.g., adapt to both non-stiff and stiff ODE systems. Therefore, the SUNDIALS library, which is incorporated into BioUML, can simulate complicated ODE systems better than the Rosenbrock solver. However, this library is not available under the LGPL and no open-source version of this solver has been implemented in Java. Therefore, we did not use the CVODE solver in our library.

The processing of algebraic rules is a major problem when simulating SBML models. Several tools do not support this element (see Table 5.2). The variable by which to solve an algebraic rule can be determined via bipartite matching (Hopcroft and Karp, 1973). However, the transformation of such a rule into an assignment rule (i.e., solving the rule by the respective variable) involves symbolic computation and is in some cases not possible. While our algorithm supports all algebraic rules occuring in the models of

Program	Version	Difficult SBML elements			Fully SBML Test Suite compliant	SED-ML	Programming language	GUI	API access	Platform	Comments
		Fast reactions	Algebraic rules	Events							
BioUML	0.9.8	✓	✓	✓	✓	✓	Java	✓	Java-Script	independent	
COPASI	4.14	–	–	(✓)	–	✓	C++ (with multiple bindings)	✓	✓	Windows, Mac OS X, Linux, Solaris	
iBioSim	3.0	✓	✓	✓	✓	✓	Java, C	✓	(✓)	Windows, Mac OS X, Linux (Fedora 17)	
JSim	2.15	–	✓	–	–	–	Java	✓	✓	Windows, Mac OS X, Linux	
LibSBMLSim	1.1.0	✓	✓	✓	(✓)	–	C (with multiple bindings)	–	✓	Windows, Mac OS X, Linux, Free BSD	
Simulation Core Library	1.4	✓	✓	✓	✓	✓	Java	–	✓	independent	
VCell	5.2	✓	–	✓	–	–	Java frontend, C/C++ server backend	✓	–	independent	Internet access required

Table 5.2: Comparison of SBML-capable simulators. The table contains the most characteristic features of SBML-capable simulation tools (January 28th 2015). It shows which programs support fast reactions, algebraic rules, and events in SBML and solve all models of the most recent SBML Test Suite (Keating *et al.*, 2013) correctly. A dash means that not all possible cases for the SBML elements are supported or not all models of the SBML Test Suite can be solved correctly, respectively. Additionally, the table provides information about SED-ML support, the underlying programming language, the existence of a GUI and of an API access, as well as the platforms supported by the software.

the SBML Test Suite and the BioModels Database, it is therefore possible to construct SBML models with algebraic rules that are not simulated correctly with the algorithm. If an equation cannot be solved by the free variable, an alternative idea is the identification of the value of the variable with nested intervals. But this would lead to a significantly higher running time, as recomputation of the nested intervals would be required at every time point during simulation. In contrast to that, the transformation approach processes each algebraic rule only once during the initialization step of the algorithm. The computation of the value of the respective assignment rule done at every time point is then not very time-consuming.

If a model contains small and fast subsystems, our algorithm processes both subsystems alternatingly. For the fast subsystem a steady state is computed after each simulation step in the slow subsystem. As this steady state determination can involve an arbitrary long simulation, the simulation times of such models are often long. In our approach we found settings that led to an adequate compromise between running time and simulation accuracy for the tested models. However, this is not necessarily applicable to all models involving different time scales.

5.8 Summary and conclusions

In the work described in this chapter we first explained SBML models mathematically and afterwards derived an algorithm for an efficient simulation of these models. An important design feature of the algorithm is the separation of SBML interpretation and numerical integration, which is why combination of the algorithm with additional numerical integration methods is easy. The Rosenbrock solver is the universal simulation method that can deal with stiff differential equation systems and precisely solve models containing diverse SBML elements.

We implemented the algorithm in the Systems Biology Simulation Core Library, which is an efficient Java API for the simulation of differential equation systems used in systems biology. Integration of the library into larger applications is straightforward. Since version 4.2 it is, e.g., integrated into CellDesigner (Funahashi *et al.*, 2003), which is an editor for biochemical models. SBMLsimulator (Dörr *et al.*, 2014), which is described in the following chapter, comprises a convenient graphical user interface for the simulation and optimization of SBML models and uses the SBSCL as a computational backend. In order to support further model formats like, e.g., CellML, it suffices to implement a suitable interpreter class. The support of SED-ML by the SBSCL facilitates exchange and reproduction of simulation experiments conducted with this library.

SBML enables adding additional model features by specifying extension packages since Level 3. Those extension packages comprise the graphical representation (Gauges *et al.*, 2006), the description of qualitative networks (Chaouiya *et al.*, 2013a), and many more. The algorithm shown here is dedicated to processing the core elements of SBML. For the different SBML packages separate interpretation algorithms are necessary.

Therefore, further work on the Systems Biology Simulation Core Library can include implementing interpreters for SBML extension packages. As previously discussed, the support for CellML and the inclusion of further numerical integration routines are other possibilities for improving the capabilities of the library.

Chapter 6

Simulation and parameter estimation of SBML models with SBMLsimulator

In the previous chapter the Systems Biology Simulation Core Library was introduced. While it supports simulation of SBML models, the results cannot be displayed graphically. Such a graphical presentation of results often helps to understand the properties of a biological system very quickly. Another important task in systems biology is the estimation of unknown model parameters (see 3.4.4). In order to enable simulation with a graphical display of the simulation results as well as parameter estimation for SBML models, SBMLsimulator has been developed. It contains the SBSCL for simulation as well as EvA2 (Kronfeld *et al.*, 2010; Becker and Kronfeld, 2014) for parameter estimation (see 3.1.2). This chapter, which is mainly based on Dörr *et al.* (2014), describes SBMLsimulator.

6.1 Important features of SBMLsimulator compared to other tools

For simulation and parameter estimation of biochemical models many programs are available including AMIGO (Balsa-Canto and Banga, 2011), SBToolbox2, (Schmidt and Jirstrand, 2006), COPASI (Hoops *et al.*, 2006), and Potters-Wheel (Maiwald and Timmer, 2008). SBMLsimulator is an easily usable parameter estimation tool that as one of very few tools fully supports SBML for all levels and versions (due to the integration of the Systems Biology Simulation Core Library). Software tools that cannot interpret all SBML elements can have a smaller running time for some models. However, they do not guarantee that all SBML models are simulated correctly.

The important features of SBMLsimulator are:

- It is platform-independent.
- It is open-source.
- It is independent of commercial software.

- It provides full support of SBML.
- Its graphical user interface was specially designed for easy usability.

6.2 Implementation

SBMLsimulator contains two libraries within one tool. Therefore, it comprises all the optimization algorithms provided by EvA2 (Kronfeld *et al.*, 2010; Becker and Kronfeld, 2014) and the modeling languages (currently only SBML, but possibly more languages in the future) and ODE solvers incorporated into the SBSCL. The modular design of SBMLsimulator enables easy exchange of both libraries. SBMLsimulator thus profits from improvements of these libraries. While the SBSCL already supports SBML completely and contains several ODE solvers, interpreters for more modeling formats and additional solvers may be added in the future (compare 5.8). Many nature-inspired heuristic optimization algorithms are contained in EvA2 (see 3.1.2) and this selection can also be extended in further versions of the toolbox.

6.2.1 Integration of SBSCL and EvA2

Simulation of a model in SBMLsimulator is conducted with the help of the SBSCL (Keller *et al.*, 2013). The SBSCL outputs the simulation results with a `MultiTable` object (see 5.4.1), which is read by SBMLsimulator in order to plot the time course of the model's variables.

For the heuristic optimization routines provided by EvA2 a fitness function is necessary (see 3.4.4). Given a set of parameter values and experimental data, this fitness function returns a value to EvA2. This value represents the distance between the simulation output with the current set of parameters and the experimental data. The fitness function is thus a measure for how well a given set of parameters reproduces the experimental data. Both the simulation results and the distance of simulated to experimental data are computed by the SBSCL. The parameters of the model are optimized by EvA2 with respect to the fitness function. This means that EvA2 tries to find a parameter set leading to the smallest possible deviation of simulation results and experimental data. Optimization targets and search intervals need to be defined by the user prior to a parameter estimation.

6.3 Program details

SBMLsimulator can be used on every platform for which a Java Virtual Machine is available. The graphical user interface (GUI) enables presentation of the simulation curves for model elements, which the user can choose. During a parameter estimation with EvA2, the current best simulation results are displayed in the GUI. Besides running

a GUI, SBMLsimulator can also be started in command-line mode, which is especially dedicated to facilitate running large optimization tasks on a cluster.

6.3.1 The graphical user interface of SBMLsimulator

The graphical user interface (see Figure 6.1) contains several sub-windows for the display of simulation results and for settings which the user can change: SBMLsimulator enables choosing the numerical solver as well as setting the start point, end point, and the step size of a simulation at the lower part of the GUI. Some integration routines (e.g., Rosenbrock's solver) are based on a fixed error tolerance, which can be specified under Edit/Preferences. Different quality functions for computing the distance between simulated and experimental data are provided by SBMLsimulator (also in the lower part of the GUI). In the upper left part of the window the model quantities to be plotted can be chosen. SBMLsimulator allows the user to change initial values of species as well as parameter and compartment values in the middle left part of the window. Once the simulation button is clicked, simulation is started with the chosen settings.

Import of experimental data is facilitated by a dedicated dialog. The most intuitive mapping of columns in the data to model quantities is suggested by SBMLsimulator. The user can approve this suggestion or change the mapping for certain columns. After uploading the experimental data, they are plotted together with the simulation results (see right part of Figure 6.1). This enables a straightforward comparison of simulated and experimental data. The distance of loaded experimental data to the simulated data is computed after a simulation and displayed at the bottom of the GUI.

The main window of SBMLsimulator (with the name "Simulation") is dedicated to the display of the simulation results and the modification of simulation settings. In addition to this window, there are windows for displaying the simulation data ("Computed data") as well as the experimental data ("Experimental data") in table form. Another window enables the user to see the structure of the model ("Model").

Experimental data are not only used for comparison to simulation results. Additionally, a parameter estimation with respect to the data can be started once experimental data have been imported into SBMLsimulator. The parameters to be estimated and their ranges need to be set by the user in a dedicated window or via importing a text file which contains the required information. Afterwards, the type of evolutionary algorithm is chosen, which can also involve changing specific settings of the method in EvA2.

When SBMLsimulator has finished processing a generation of parameter sets during optimization, the current best simulation curves and the experimental data are plotted. Such an intermediate result is displayed in Figure 6.1. The user can thus follow the decrease of the distance between simulated and experimental data.

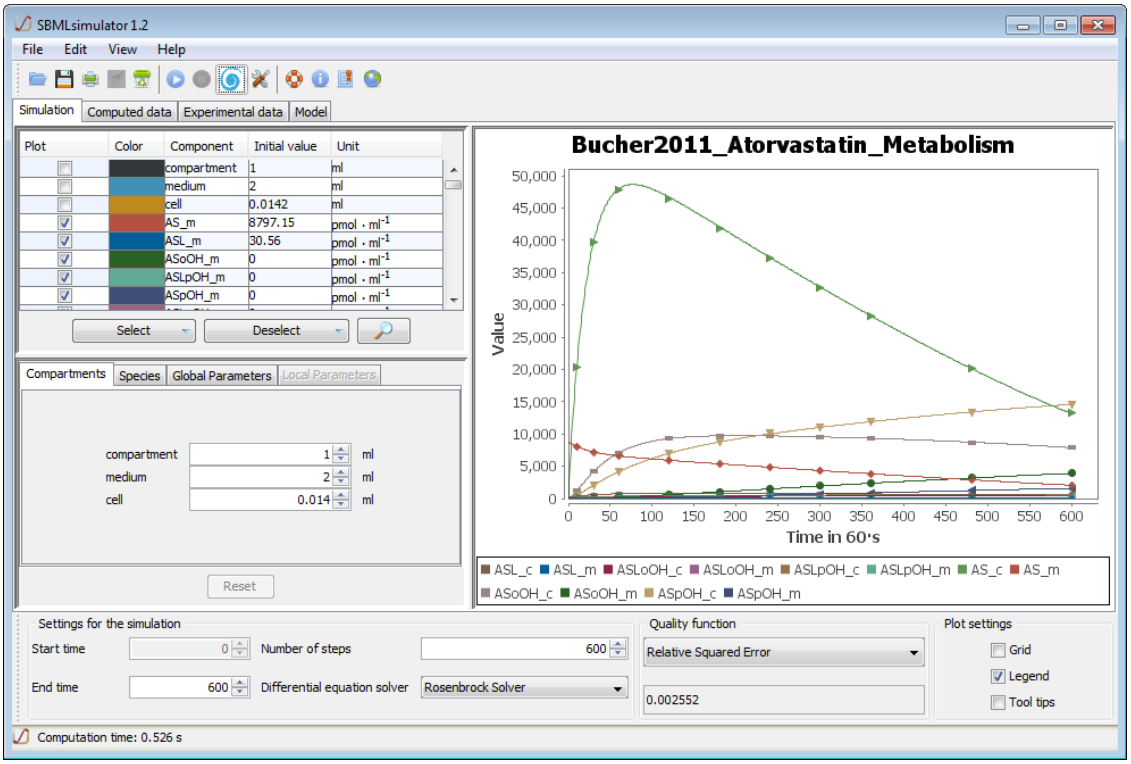

Figure 6.1: The graphical user interface (GUI) of SBMLsimulator. The main window of SBMLsimulator after importing a model describing atorvastatin biotransformation (Bucher *et al.*, 2011) is displayed in this figure. The user can change initial quantities (middle left part of window) and decide which quantities are to be plotted (upper left). In addition, settings for simulation, such as the integration routine, the simulation start and end time, the simulation step size, and the quality function for comparing the simulated data to experimental data, can be defined at the bottom of the window. The window contains several buttons below the menu bar, which can be, e.g., used to start a simulation or a parameter estimation with EvA2. In the right part of the figure an intermediate solution of such a parameter estimation is shown. The experimental values (in this case articifial values) are shown as shapes and the simulated values with the current set of parameters are presented as curves. Here EvA2 already found a parameter set leading to a small deviation between experimental and simulated data.

6.3.2 The command-line mode for running time-consuming jobs on a cluster

For each fitness evaluation during parameter estimation a simulation is necessary, which is why the whole optimization process in EvA2 can take several hours or days for some models. The running time of a parameter estimation is thus heavily influenced by the time needed for conducting a single simulation. Parameter estimations with long running times will often be performed in the command-line mode instead of using the GUI-mode in which SBMLsimulator is started by default. The command-line mode facilitates running time-consuming parameter estimation tasks on a computer cluster. Such tasks are usually conducted multiple times, which is another reason for transferring them to a cluster. The repetition of a parameter estimation is advisable, as an optimization run can get stuck in a local optimum of the fitness function. Furthermore, with multiple runs of a parameter estimation one can assess whether the estimated quantities are identifiable (Schilling *et al.*, 2009). This has already been explained at the beginning of this thesis (see 3.4.5).

6.4 Example for using SBMLsimulator as a proof of concept

We estimated the parameters of a model explaining the biotransformation of atorvastatin (Bucher *et al.*, 2011) in order to demonstrate the usefulness of SBMLsimulator. The model is stored in the BioModels Database (Chelliah *et al.*, 2013) under the accession number BIOMD0000000328. First, an artificial data set was created by simulating the model with the start and end time described in the publication. The simulated data were saved and the values at time points similar to those used in the publication were extracted. After the created data set had been read by SBMLsimulator, optimization with EvA2 was started with respect to this data set using differential evolution (Storn and Price, 1997). This optimization involved estimating the same parameters as in the publication, with the intervals given in Table 6.1.

SBMLsimulator enables to define separate parameter intervals for their random initialization (minimum/maximum initial value) and for the following optimization procedure (minimum/maximum value). Here the same intervals were chosen for both cases. Simulation was performed with the Rosenbrock solver (Press *et al.*, 1993) and an absolute error tolerance of 10^{-12} as well as a relative error tolerance of 10^{-6}, because this method can solve stiff differential equation systems. The relative squared error (RSE), which has been suggested previously (Dräger *et al.*, 2009; Dräger, 2011), was used as quality function. As the exact definition of the RSE will be more relevant in the next chapter, it will be provided there.

The parameters of the atorvastatin biotransformation model were estimated 100 times on a computer cluster. In order to exclude parameter estimations stuck in a local op-

Table 6.1: Estimated parameters with units, their initial intervals and their intervals during parameter estimation for the atorvastatin biotransformation model (Bucher *et al.*, 2011).

Parameter	Unit	Minimum initial value	Maximum initial value	Minimum value	Maximum value
`Import_ASLpOH_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`Import_ASLoOH_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`Import_ASpOH_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`Export_ASLpOH_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`Export_ASLoOH_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`Export_ASoOH_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`Export_AS_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`Export_ASL_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`Import_AS_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`Import_ASoOH_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`Export_ASpOH_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`k_PON_OH_c`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`k_PON_ASL_c`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	0.1	10^{-6}	0.1
`Import_ASL_k`	$\mathrm{ml \cdot min^{-1}}$	10^{-6}	1	10^{-6}	1
`fu_AS`	dimensionless	10^{-6}	1	10^{-6}	1
`fu_ASL`	dimensionless	10^{-6}	1	10^{-6}	1
`CYP3A4_ASoOH_Vmax`	$\mathrm{pmol \cdot min^{-1}}$	10^{-6}	100	10^{-6}	100
`CYP3A4_ASLpOH_Vmax`	$\mathrm{pmol \cdot min^{-1}}$	10^{-6}	100	10^{-6}	100
`CYP3A4_ASLoOH_Vmax`	$\mathrm{pmol \cdot min^{-1}}$	10^{-6}	100	10^{-6}	100
`CYP3A4_ASpOH_Vmax`	$\mathrm{pmol \cdot min^{-1}}$	10^{-6}	100	10^{-6}	100
`UGT1A3_AS_Vmax`	$\mathrm{pmol \cdot min^{-1}}$	10^{-6}	100	10^{-6}	100

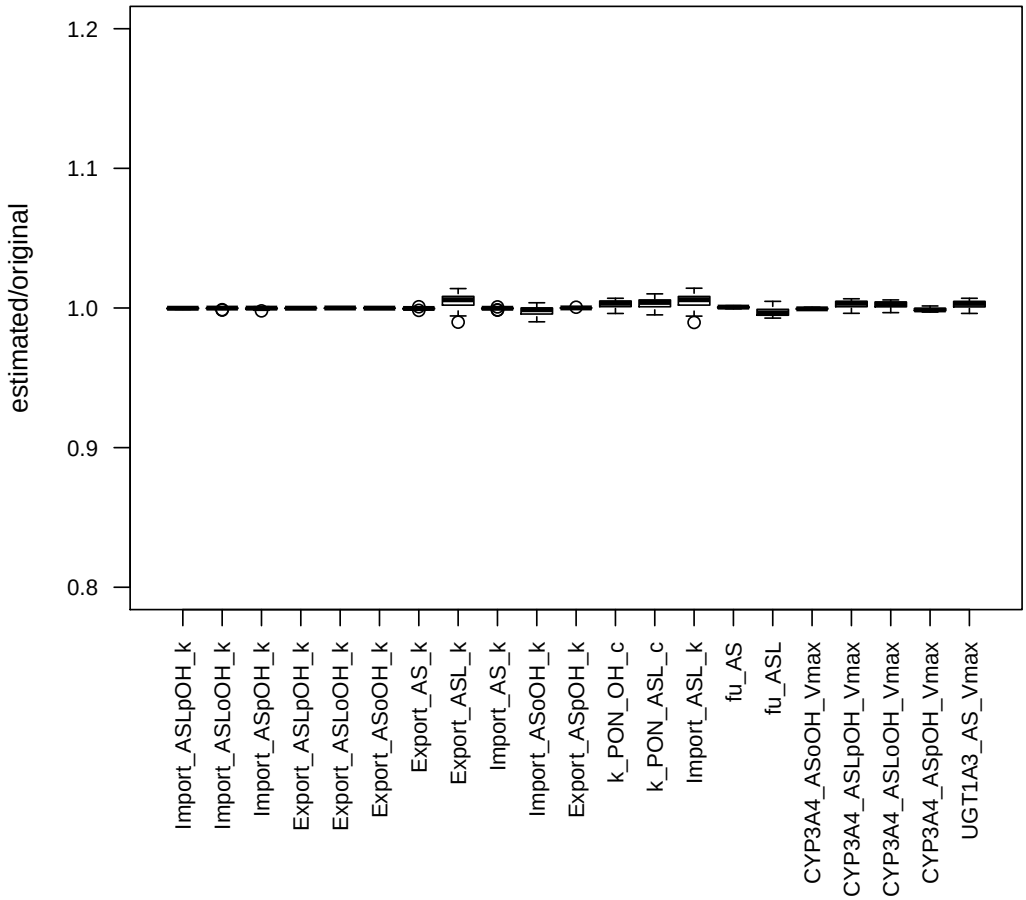

Figure 6.2: Distribution of parameters estimated with SBMLsimulator. SBMLsimulator was used to run 100 parameter estimations for the atorvastatin biotransformation model (Bucher *et al.*, 2011) on a computer cluster. Shown here is the distribution of the 50 estimations with the best fitness values. The estimated values were divided by the original parameter values prior to plotting for the sake of their standardization. The plot demonstrates that all parameters were estimated closely around their original values. The figure has been created with the R software package (R Development Core Team, 2011).

timum of the fitness function, the 50 % of the estimated parameter sets with the best fits were extracted from all resulting parameter sets. The distributions of the parameters were then plotted based on these extracted parameter sets (see Figure 6.2). A high standard deviation for the estimates of some parameter would have suggested that this parameter could not be identified. However, for all estimated parameter values the standard deviation was low, which demonstrates that SBMLsimulator was able to identify all parameters. These results demonstrate that the linkage of parameter estimation and simulation in SBMLsimulator works well.

6.5 Summary and conclusions

SBMLsimulator is a platform-independent program for simulation and parameter estimation of biochemical models, which completely supports the SBML standard. Two powerful toolboxes are combined in one graphical interface: the SBSCL for simulation and the optimization framework EvA2 for estimation of parameter values. SBMLsimulator can be used for manual exploration of parameter space (by changing parameter values in the GUI) and for automatic calibration of large biochemical models with optimization routines. By modifying individual parameters manually the user possibly gets new insight into the behavior of a model. With its command-line mode SBMLsimulator enables transferring time-consuming optimizations to a computer cluster, which often involves running a parameter estimation multiple times in parallel. A small parameter estimation study with repeated optimizations was done based on a published model. The capability of SBMLsimulator to identify parameters was thus demonstrated.

Future versions of SBMLsimulator could include the option to perform a parameter estimation in log-scale. This can be advantageous, because the parameters are often of different magnitudes (Raue *et al.*, 2013). The following chapter describes a parameter estimation study involving a beta version of SBMLsimulator which already enables such a log-scale parameter estimation.

SBMLsimulator should also be able to simulate models of a format different from SBML, such as CellML (Cooling, 2010). This has already been discussed for the SBSCL (see 5.8). Once the SBSCL is extended with CellML support, after some small modifications SBMLsimulator will be able to simulate models in CellML like the CellMLSimulator (Nickerson *et al.*, 2008).

Chapter 7

Investigating the influence of experimental noise on parameter estimation

In the last chapter parameter estimation with SBMLsimulator was successfully tested based on time-resolved artificial data. These data, which were extracted from model simulation results and therefore perfect, were adequate for demonstrating the capabilities of SBMLsimulator. However, such perfect data is far from realistic and the quality of experimental data can differ a lot. Besides the measured time points, this quality depends on the magnitude of experimental noise in the data, which might hamper estimation of unknown parameters. This leads to the question what variance in the data is tolerable for available biochemical models, such that an acceptable parameter estimation is still possible. This chapter therefore describes a study investigating the influence of noise on parameter estimation with a beta version of SBMLsimulator supporting parameter estimation in log-scale.

7.1 Uncertainty analysis and noise addition to data

When estimating parameters with noisy data, one is not only interested in the concrete parameter values, but also in the reliability of the estimated parameters. Some parameters may not be identifiable due to the model structure or the data available for parameter estimation. Methods for investigating this parameter identifiability have already been described at the beginning of this thesis (see 3.4.5). Related to identifiability testing is the uncertainty analysis, which involves the determination of confidence intervals for parameters. Different methods have been suggested for obtaining confidence intervals (Rodriguez-Fernandez *et al.*, 2006; Joshi *et al.*, 2006; Kirk and Stumpf, 2009). The approach by Rodriguez-Fernandez *et al.* is based on the computation of the Fisher information matrix. In contrast to that, the other two methods involve bootstrapping of the experimental data and repeated parameter estimation with the bootstrapped data sets. The method by Kirk *et al.* also comprises Gaussian regression of the experimental data.

Repeated addition of noise to experimental data before parameter estimation is a key principle of both latter approaches.

Furthermore, in systems biology studies noise is often added to artificial data (e.g., Guillén-Gosálbez *et al.* (2013)). In a small study with noisy artificial data different fitness functions were compared for a small toy model (Raue *et al.*, 2013). Here we conducted a large study involving parameter estimation based on data which contained different magnitudes of noise. Three already published models of different sizes from the BioModels Database (Le Novère *et al.*, 2006; Li *et al.*, 2010) were included in this work.

7.2 Data creation for the parameter estimation study

Perfect data were created by simulating the models with the known parameters. From the obtained simulation data we extracted the respective data points which were also contained in the original experimental data used by the model creators. The included quantities and time points of our artificial data are thus similar to those in the data which the model creators applied for parameter estimation. We added noise to the perfect data in order to create data sets resembling real experimental data.

For the study an assumption was that the experimental noise was normally distributed, which was also assumed in similar studies (Kirk and Stumpf, 2009):

$$x_m(q,t) = x(q,t) + \varepsilon(q,t), \qquad \varepsilon(q,t) \sim N(0, \sigma^2(q,t)) \tag{7.1}$$

From a perfect data point $x(q,t)$ with the value of a quantity q determined at time point t we thus created a noisy data point $x_m(q,t)$ by adding a value drawn from a normal distribution with mean 0 and variance $\sigma^2(q,t)$. Kirk and Stumpf assumed that the variance was independent of x (Kirk and Stumpf, 2009). In contrast to that, our study addresses the more general case, in which the variance can be dependent on x. Therefore, we assumed that the standard deviation (i.e., the square root of the variance) for a specific data point was a certain proportion of the value of that point. It could then be derived by multiplying the value with a specific factor $f(q)$:

$$\sigma(q,t) = f(q) \cdot x(q,t) \tag{7.2}$$

The noise added to a perfect data point is thus relative noise.

For setting of $f(q)$ of a quantity q we drew a value from $[0.5p_{mean}, 1.5p_{mean}]$, where p_{mean} was a fixed mean factor (e.g., 5%). We thus obtained random factors for each quantity around the fixed mean factor. With those different factors we took into account that the variability of certain quantities is often greater than that of others. Experimental replicates, which usually exist in experimental data, were created by repeating the procedure of adding noise to perfect data with the same values $f(q)$ for each quantity. The creation of the noisy data sets involved random steps (i.e., the setting of $f(q)$ and the

drawing from a normal distribution), which is why this step was repeated 20 times. A higher number of repetitions would have been preferable, but was infeasible due to the long running times of the parameter estimations.

7.3 Models used for analysis

Parameter estimations were conducted with three different models. The same quantities as in the publications were estimated. Some quantity ranges were not given or had to be adapted for our study (see below).

As already mentioned, the time points for the artificial data sets were chosen similar to those in the respective publications. A special case is time point 0, because values from this time point can be extracted from the data and taken as initial species values for simulation of the model. In the atorvastatin biotransformation model this was done for two of the species by the model creators. The other initial species values in the three models were not extracted from the experimental data. They were instead based on specific assumptions (e.g., the value being 0 at the beginning) or estimated together with the other quantities. Values of 0 in the simulated data arising from such an assumption are difficult to handle for our noise addition procedure, as it is based on multiplication of a data point with a factor $f(q)$ (see Equation (7.2)). For simplicity, we thus did not include time point 0 in the data for the ERK signaling and the Epo receptor model. When processing the atorvastatin biotransformation model, the initial values of the two mentioned species were estimated around the respective values in the data at time point 0 (see below). The data for time point 0 were then excluded from the computation of the fitness values.

7.3.1 Epo receptor model

The model describes information processing at the erythropoetin (Epo) receptor (Becker *et al.*, 2010). In the BioModels Database it is stored under the identifier BIOMD0000000271. 8 reactions, 6 species, and 8 kinetic parameters are contained in the model.

The data used for parameter estimation comprised measurements of three quantities, two of which were sums of specific species values (`Epo_medium`, `Epo_cell`) and one was a direct species value (`Epo_EpoR`). 10, 30, 60, 120, 180, 240, and 300 minutes were the time points included in the data.

Prior to estimation the parameter `Bmax` was fixed by the model creators. The parameter `koff` depends on `kon`, which is why it was not estimated, but calculated based on the estimated value for `kon`. Some other parameter (`kt`) was determined by the model creators using an auxiliary model. In our study we therefore considered this parameter as fixed. 6 quantities (5 parameters and the initial value of the species `Epo`) to estimate remained and their ranges could be taken from the publication (see Table 7.1).

Table 7.1: The table contains the estimated quantities and their ranges for the Epo receptor model (Becker *et al.*, 2010).

Quantity	Minimum value	Maximum value
kon	10^{-7}	1000
ke	10^{-7}	1000
kex	10^{-7}	1000
kdi	10^{-7}	1000
kde	10^{-7}	1000
Epo (initial value)	1890	2310

7.3.2 Atorvastatin biotransformation model

The biotransformation of the drug atorvastatin in human hepatocytes is described by this model (Bucher *et al.*, 2011), which was already used in the previous chapter (see 6.4). In the BioModels Database the model is stored with the identifier BIOMD0000000328. This model contains some changes compared to the model described in the publication, such as renaming of parameters. Here the model available at the BioModels Database, which comprises 29 reactions, 18 species, and 30 parameters, was used.

Measurements for 12 of the model species (AS_m, ASL_m, ASoOH_m, ASLpOH_m, ASpOH_m, ASLoOH_m, AS_c, ASL_c, ASpOH_c, ASoOH_c, ASLpOH_c, ASLoOH_c) were contained in the data. An initial value of 0 was assumed by the model creators for all but two of the species (AS_m, ASL_m). For AS_m and ASL_m we extracted in each estimation run the values at time point 0 from the corresponding data (only one value in the case of non-replicative data). The initial values of the two species were then estimated between 90% of the minimum extracted value and 110% of the respective maximum value (compare discussion at the beginning of this section). The time points in the data were: 0 (only for AS_m and ASL_m), 10, 30, 60, 120, 180, 240, 300, 360, 480, and 600 minutes. Calculation of the fitness function for a solution during the estimation procedure was done based on the data of all 12 measured species at the time points different from 0. In contrast to that, the data points at time point 0 for AS_m and ASL_m were used for deriving the estimation ranges of their initial values (see above).

Several quantities were fixed by the model creators before estimation. The estimation ranges of the remaining 21 quantities (see Table 7.2), all of them kinetic parameters, were not given in the publication. We therefore set these ranges such that the estimation procedure was reliably able to identify the global optimum: For all quantities a range of $[10^{-4}, 100]$ was applied during estimation. The chosen ranges were thus different from those in the previous chapter (compare Table 6.1), as a logarithmic parameter space could be used in this study.

Table 7.2: The table comprises the estimated quantities for the atorvastatin biotransformation model (Bucher *et al.*, 2011) and short identifiers for these quantities.

Quantity	Short identifier
Import_ASLpOH_k	P1
Import_ASLoOH_k	P2
Import_ASL_k	P3
Import_ASpOH_k	P4
Export_ASLpOH_k	P5
Export_ASLoOH_k	P6
Export_ASoOH_k	P7
Export_AS_k	P8
Export_ASL_k	P9
Import_AS_k	P10
Import_ASoOH_k	P11
Export_ASpOH_k	P12
CYP3A4_ASoOH_Vmax	P13
CYP3A4_ASLpOH_Vmax	P14
CYP3A4_ASLoOH_Vmax	P15
CYP3A4_ASpOH_Vmax	P16
UGT1A3_AS_Vmax	P17
k_PON_OH_c	P18
k_PON_ASL_c	P19
fu_AS	P20
fu_ASL	P21

7.3.3 ERK signaling model

This model describes the Epo induced ERK (extracellular signal-regulated kinase) signaling cascade (Schilling *et al.*, 2009). It is available in the BioModels Database (model identifier BIOMD0000000270) and contains 42 reactions, 32 species, 24 parameters, 5 scaling factors, and 4 constraints.

Measurements of 8 quantities (`rescaled_pEpoR`, `rescaled_ppMEK1`, `rescaled_ppMEK2`, `rescaled_ppERK1`, `rescaled_ppERK2`, `rescaled_pJAK2`, `rescaled_pSOS`, `rescaled_mSOS_SOS`) were contained in the data. All of those quantities are derived in the model by multiplying a scaling factor with a certain species value. The following time points were included in the data: 0.5, 1, 1.5, 2, 3, 4, 5, 6, 7.5, 9, 10.5, 12, 13.5, 15, 16.5, 18, 20, 22, 25, 28, 31, 35, 40, 45, 50, 60, 65, and 70 minutes.

Parameter estimation and following identifiability tests were conducted repeatedly by the creators of the model. Dependent quantities were fixed to values producing the best fit after each round of such tests. Of the remaining 21 quantities to estimate, 17 were parameters, 3 were scaling factors and one was the initial concentration of species `SOS`. We took the ranges of the scaling factors and of the initial concentration from the publication. Estimation with the perfect data got frequently stuck in a local optimum, which is why the parameter ranges were narrowed compared to those in the publication (see Table 7.3). The global optimum could then be reliably reached by the estimation procedure without raising the number of estimation repetitions or the number of fitness evaluations, which both would have consumed much more running time.

7.4 Settings for optimization

We used a beta version of SBMLsimulator (Dörr *et al.*, 2014) for simulation and parameter estimation. In addition to the version of SBMLsimulator described in the previous chapter, a logarithmic parameter space can be selected in this beta version. This logarithmic parameter space was previously suggested (Raue *et al.*, 2013) and proved effective for our study. Rosenbrock's solver (Press *et al.*, 1993) with absolute tolerance 10^{-12} and relative tolerance 10^{-6} was chosen as integration routine.

The performance of several heuristic optimization methods when estimating the parameters of a metabolic network was compared by Dräger *et al.* (2009). One of the best performing methods was differential evolution (Storn, 1996; Storn and Price, 1997), which has been explained at the beginning of the thesis (see 3.1.3). This method was used with its standard settings in EvA2 ($F = 0.8$, $CR = 0.6$, $\lambda = 0.6$, population size of 50, DE type: DE/current-to-best) for the whole study.

There is no guarantee that evolutionary algorithms always find the globally optimal solution. They get instead sometimes stuck in a local optimum. Estimation for each data set (or combination of replicative data sets, respectively) was thus repeated 10 times when optimizing the Epo receptor and the ERK signaling model and 20 times when optimizing

Table 7.3: The table displays the estimated quantities, their ranges, and short quantity identifiers for the ERK signaling model (Schilling *et al.*, 2009).

Quantity	Short identifier	Minimum value	Maximum value
JAK2_phosphorylation_by_Epo	P1	10^{-3}	1
SHP1_activation_by_pEpoR	P2	10^{-3}	1
actSHP1_deactivation	P3	10^{-3}	1
pJAK2_dephosphorylation_by_actSHP1	P4	10^{-3}	1
SOS_recruitment_by_pEpoR	P5	10^{-3}	1
mSOS_induced_Raf_phosphorylation	P6	10^{-3}	1
pRaf_dephosphorylation	P7	10^{-3}	1
First_MEK_dephosphorylation	P8	10^{-3}	1
pSOS_dephosphorylation	P9	10^{-3}	1
pEpoR_dephosphorylation_by_actSHP1	P10	10^{-3}	10
First_MEK2_phosphorylation_by_pRaf	P11	10^{-3}	10
First_ERK2_phosphorylation_by_ppMEK	P12	10^{-3}	10
Second_ERK_dephosphorylation	P13	10^{-3}	10
mSOS_release_from_membrane	P14	10^{-3}	100
Second_ERK1_phosphorylation_by_ppMEK	P15	10^{-3}	100
First_ERK_dephosphorylation	P16	10^{-3}	100
Second_MEK1_phosphorylation_by_pRaf	P17	10^{-3}	1000
SOS (initial value)	P18	0.1	100
scale_pEpoR	P19	0.01	50
scale_pJAK2	P20	0.01	50
scale_ppERK	P21	0.01	50

the atorvastatin biotransformation model. The higher number of repetitions for the last model is due to the fact that finding the global optimum appeared more difficult here than for the other two models. For further analysis we extracted the solution yielding the best fitness values from all 10 (or 20) quantity sets resulting from optimization.

In addition to the described settings, the number of fitness evaluations for an estimation run was fixed to 150,000 for the Epo receptor and the ERK signaling model and 200,000 for the atorvastatin biotransformation model. Then differential evolution was always able to identify the global optimum (i.e., a solution with a low deviation to the known real quantity values) with respect to perfect data.

7.5 Fitness functions and estimation constraints

Parameter estimation was run with different fitness functions depending on whether replicative or non-replicative data sets were used. All applied fitness functions are presented in this section. For the ERK signaling model constraints were given, which is why we first explain how to incorporate constraints into the fitness function.

Quantity estimation for the ERK signaling model involved constraints given in the following form:

$$\frac{\max(q_1)}{\max(q_2)} = v \tag{7.3}$$

This means that the fraction between the maximum values of two model quantities q_1 and q_2 should have the value v. Similar to the approach of the model creators, we added the following term to the fitness function:

$$\left(\frac{\max(q_1)}{\max(q_2)} - v\right)^2 \tag{7.4}$$

For non-replicative data the relative squared error (RSE) was used as fitness function, which was suggested, e.g., by Dräger *et al.* (2009). The definition of the RSE is as follows:

$$E_{RSE} = \sum_{q\in Q}\sum_{t\in T}\left(\frac{x_{\text{pred}}(q,t) - x_m(q,t)}{x_m(q,t)}\right)^2 \tag{7.5}$$

$x_{\text{pred}}(q,t)$ represents the simulated value for quantity q at time point t, while $x_m(q,t)$ is the respective measured value.

In the previous formula we assumed that the data only contained one replicate of each data point. The RSE for data comprising n replicates can be calculated by summing up the RSE for each replicative data set and dividing by the number of replicates afterwards:

$$E_{RSE,n} = \frac{\sum_{i=1}^{n}\sum_{q\in Q}\sum_{t\in T}\left(\frac{x_{\text{pred}}(q,t) - x_m(q,t,i)}{x_m(q,t,i)}\right)^2}{n} \tag{7.6}$$

$x_m(q,t,i)$ is the measured value for quantity q at time point t in the replicate i. The division by the number of replicates n is included in the formula in order to have a similar influence of constraints for different values of n. In our study n was always 3 for the replicative data, which is common in experimental measurements.

Instead of just computing the distance of the simulated data to each replicative data set, one can also include the standard deviation of the data into the fitness function, which was suggested, e.g., by Raue *et al.* (2013) and Vanlier *et al.* (2013). Then a data point with a high standard deviation contributes less to the fitness. In our study the variance depends on the data, which is why the formula is as follows:

$$E_{SD,n} = \frac{\sum_{i=1}^{n} \sum_{q\in Q} \sum_{t\in T} \left(\frac{x_{\text{pred}}(q,t) - x_m(q,t,i)}{\sigma(q,t)} \right)^2}{n} \tag{7.7}$$

This equation only differs from Equation (7.6) in the division by the standard deviation σ of a data point instead of its value. A division by the number of replicates n is also included here because of a possible addition of terms for constraints. We multiply those added constraint terms with the factor 20 in order to give them a weight more similar to that for the other fitness functions which do not involve a division by the standard deviation.

Instead of computing the variances (or the standard deviations) from the data, they can also be estimated, as suggested by Raue *et al.* (2013) and Vanlier *et al.* (2013). Here we calculated the standard deviations based on just three data points (i.e., the number of replicates). Therefore, the value of the fitness function is possibly very sensitive to noise in the data. An estimation of the factors $f(q)$ contained in Equation (7.2) together with the estimated quantities could circumvent this problem.

Similar to Vanlier *et al.* we determine the probability density for observing the measurement data given the estimated quantities. To this end, a key assumption is independent additive Gaussian noise in the data. In contrast to Vanlier *et al.*, the variance in our study is specific for each metabolite q and time point t. With those assumptions the probability density of the measured data x_m comprising n replicates given the parameters θ is computed as:

$$p(x_m|\theta) = \prod_{i=1}^{n} \prod_{q\in Q} \prod_{t\in T} p(x_m(q,t,i),\theta) \tag{7.8}$$

$$= \frac{1}{\prod_{q\in Q} \prod_{t\in T} \left(\sqrt{2\pi}\sigma(q,t)\right)^n} e^{-\sum_{i=1}^{n} \sum_{q\in Q} \sum_{t\in T} \left(\frac{x_{\text{pred}}(q,t) - x_m(q,t,i)}{\sqrt{2}\sigma(q,t)} \right)^2} \tag{7.9}$$

Maximizing this likelihood is equivalent to minimizing the negative logarithm of the

likelihood. This leads to the following formula:

$$-2\ln\left(p(x_m|\theta)\right) = \sum_{q\in Q}\sum_{t\in T} n\cdot\ln\left(2\pi(\sigma(q,t))^2\right) + \sum_{i=1}^{n}\sum_{q\in Q}\sum_{t\in T}\left(\frac{x_{\text{pred}}(q,t)-x_m(q,t,i)}{\sigma(q,t)}\right)^2 \tag{7.10}$$

With the assumption for the standard deviations (Equation (7.2)) and including the division by the number of replicates n, we obtain this equation:

$$E_{ML,n} = \frac{\sum_{q\in Q}\sum_{t\in T} n\cdot\ln\left(2\pi\left(f(q)\cdot x(q,t)\right)^2\right) + \sum_{i=1}^{n}\sum_{q\in Q}\sum_{t\in T}\left(\frac{x_{\text{pred}}(q,t)-x_m(q,t,i)}{f(q)\cdot x_{\text{pred}}(q,t)}\right)^2}{n} \tag{7.11}$$

Added constraint terms are again multiplied by 20, because the formula includes a division by the (estimated) standard deviation.

If Equation (7.11) was used as fitness function, the estimation procedure involved additional estimation of the factors $f(q)$. In this case these factors were estimated between 10^{-3} and 0.5.

7.6 Results of optimization with respect to single artificial data sets

Applying the procedure described in Section 7.2, we created 20 data sets for each of the three models and for different magnitudes of relative noise (no noise, 5%, 10%, 15%, and 20% noise).

In the absence of experimental noise, the estimated values of all unknown quantities of the Epo receptor model were very accurate (see Figure 7.1). With a mean noise of 5% the quantities were estimated within ± 30% of the real values, which is acceptable. When increasing the noise to 10%, with one exception (for parameter `kex`) the estimation results were still within ± 50% of the real values. The occurence of such an outlier could also mean that the global optimum of the fitness function was not found for the respective data set. When the noise was further increased to 15%, `kex` became clearly unidentifiable (i.e., the respective interval of the estimation results spread to ± 100% of the real value). One more quantity (`kdi`) could not be identified with 20% noise.

Regarding the atorvastatin biotransformation model, with perfect data all quantities could be estimated close to their real values (see Figure 7.2), but increasing the noise to 5% noise already caused a spread of the estimation intervals beyond ± 50% of the real values for three quantities. Such large intervals were present for two more quantities with 10% noise. A further increase of the noise gave rise to more than half of the quantity intervals spreading beyond ± 50% of the real values with most of these quantities being clearly unidentifiable.

The estimation results for the unknown quantities of the ERK signaling model were

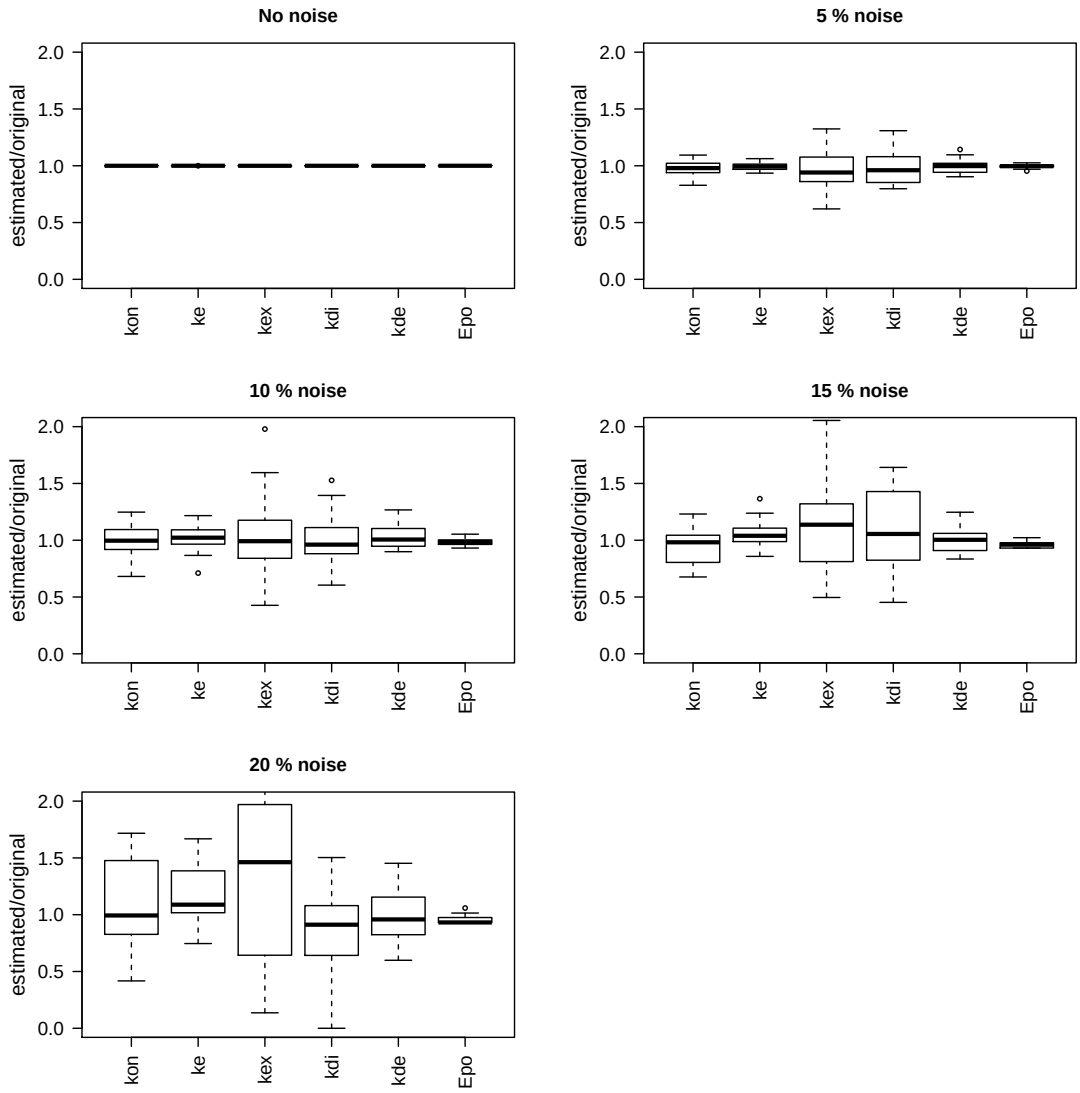

Figure 7.1: Distributions of the estimated quantities of the Epo receptor model with different magnitudes of noise. The figure shows the distributions of the quantity estimates of the Epo receptor model for 20 data sets and with different magnitudes of noise. The estimated values were divided by the original quantity values before plotting. Like all the following figures in this chapter and in Appendix B the figure has been created with the R software package (R Development Core Team, 2011).

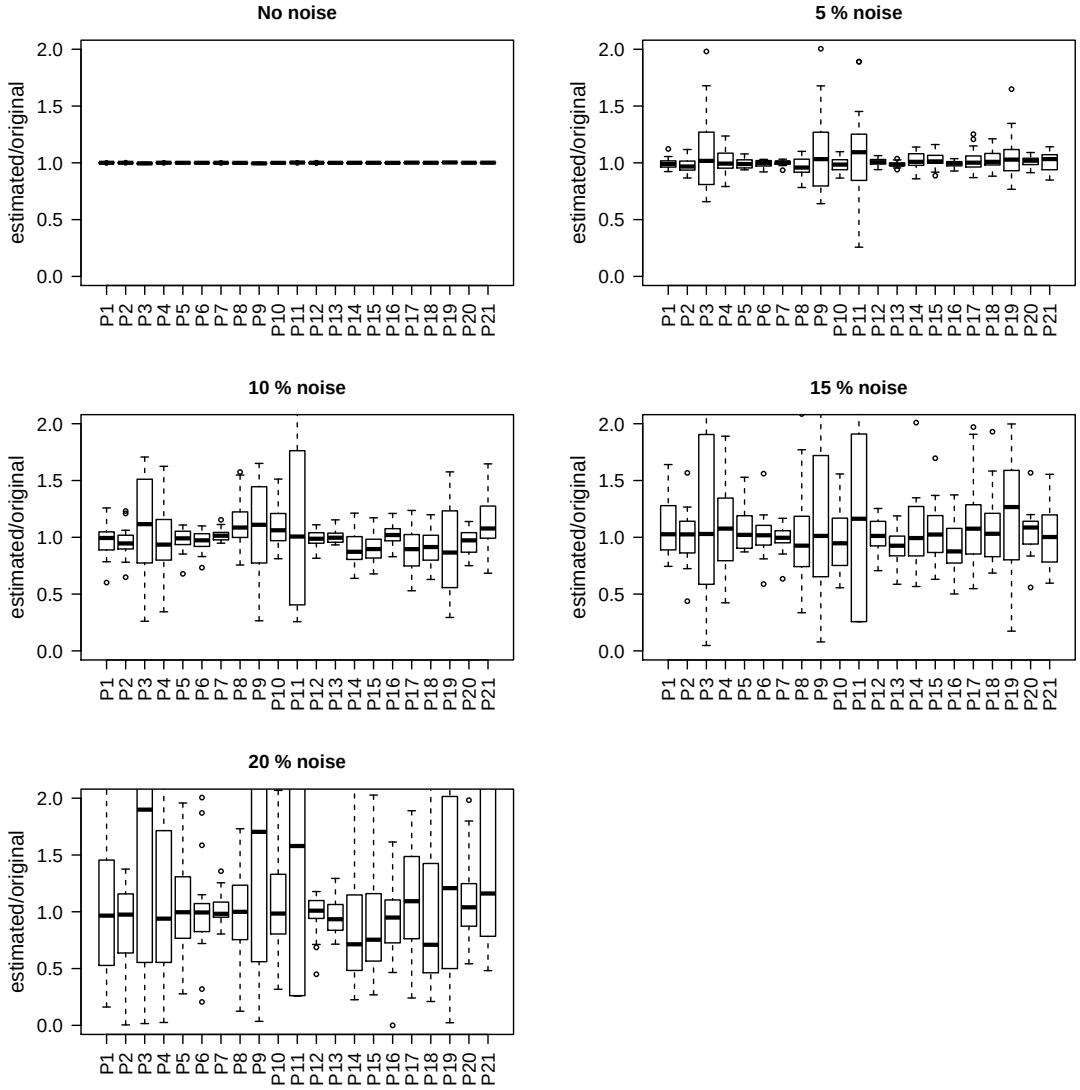

Figure 7.2: Distributions of the estimated quantities of the atorvastatin biotransformation model with different magnitudes of noise. The distributions of the quantity estimates of the atorvastatin biotransformation model for 20 data sets with different magnitudes of noise are plotted. Short identifiers are used for the quantities (see Table 7.2). Before plotting the estimated values were divided by the original quantity values.

within 15% of the original values in the absence of noise (see Figure 7.3). This deviation, which is greater than for the other two models, is due to the fact that during estimation we applied the relative squared error as fitness function for all three models. In contrast to that, the authors of the model used a χ^2 function. The values estimated by us are, however, still acceptable. With a noise of 5% all quantity intervals were within $\pm$ 50% of the original values. Clear unidentifiability of any quantity was still not present when the noise was increased to 10%, although up to two outliers occurred for some quantities with deviations from the real values greater than $\pm$ 50%. A further increase of the noise to 20% led to a larger spread of the quantity estimates. However, compared to the atorvastatin biotransformation model, which comprises a similar number of unknown quantities, the intervals of the estimation results were much smaller on average.

7.7 Results of optimization with respect to replicative data sets

Artificial replicative data sets were created as described in Section 7.2, because biological data usually consist of experimental replicates (frequently $n = 3$). We produced 20 noisy data sets containing three replicates each for the same magnitudes of relative noise as before (no noise, 5%, 10%, 15% and 20% noise). Estimation with respect to those data sets was then conducted using the different fitness functions defined in Section 7.5: the mean relative squared error (Equation (7.6)), the function involving the standard deviation calculated from the data (Equation (7.7)), and the function comprising the estimated standard deviations (Equation (7.11)). The obtained results were compared to the estimation results with single data sets (see previous section).

Figures 7.4 and 7.5 contain the distributions of the estimated quantities of the Epo receptor model using the different fitness functions for a noise of 10% and 20%, respectively. The distributions for 5% and 15% noise are given in Appendix B (Figures B.1 and B.2). Estimation with respect to replicative data is clearly advantageous, especially for higher magnitudes of noise. Compared to the other fitness functions, estimation of the standard deviations together with the unknown quantities led to intervals of estimated quantities narrowest around the real values. Using this approach, with 20% noise all but one of the quantities were estimated within $\pm$ 50% of the real values (with the exception of one outlier). In contrast to that, the estimation results when computing the standard deviations from the data prior to optimization were not clearly better than the results when applying the mean relative squared error as fitness function.

Using the atorvastatin biotransformation model, estimation of the standard deviations together with the parameters yielded clearly better estimation results for 10% and 20% noise than using the other fitness functions (see Figures 7.6 and 7.7). In contrast to the results when applying the other fitness functions, the intervals of all but one of the estimated parameters were within approximately $\pm$ 50% of the real values with 10%

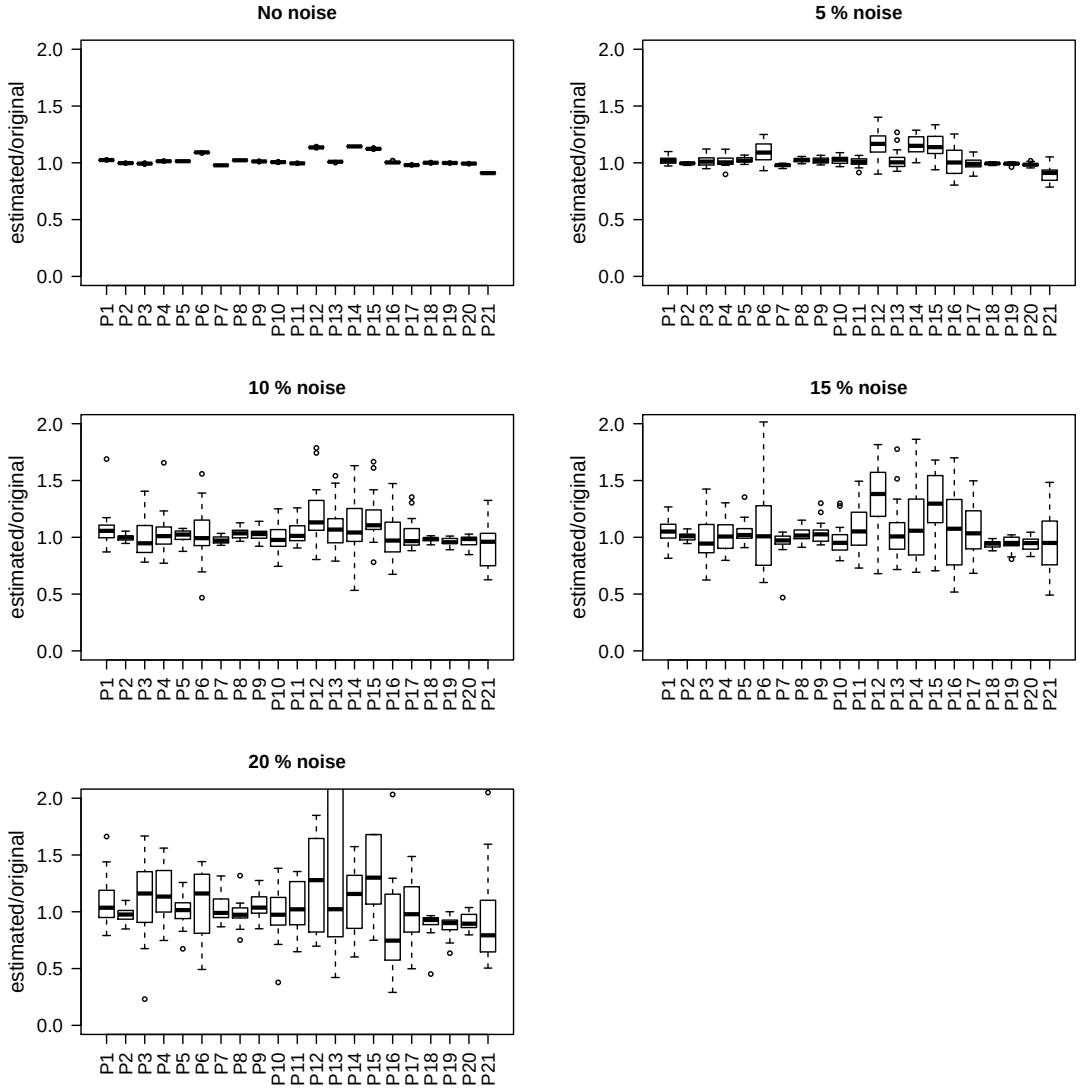

Figure 7.3: Distributions of the estimated quantities of the ERK signaling model with different magnitudes of noise. The figure shows the distributions of the quantity estimates of the ERK signaling model for 20 data sets with different magnitudes of noise. Short identifiers are used for the quantities (see Table 7.3). Prior to plotting the estimated values were divided by the original quantity values.

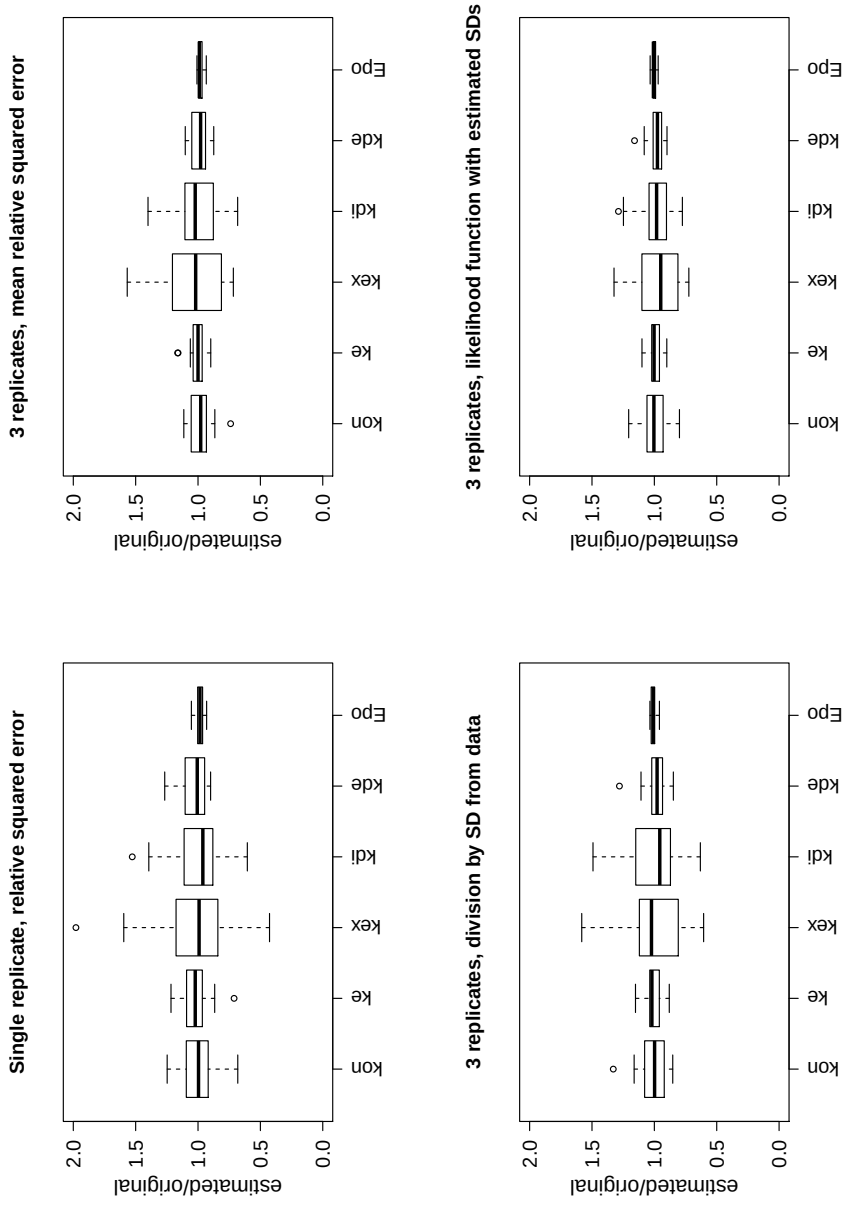

Figure 7.4: Distributions of the estimated quantities of the Epo receptor model with 10% measurement noise using different fitness functions. The figure shows the distributions of the quantity estimates of the Epo receptor model for 20 noisy data sets using different fitness functions for parameter estimation: relative squared error for single replicates and triplicates (Equation (7.5), Equation (7.6)), division by the standard deviation of each data point in the fitness function (Equation (7.7)), fitness function comprising estimated standard deviations (Equation (7.11)). The estimated values were divided by the original quantity values before plotting.

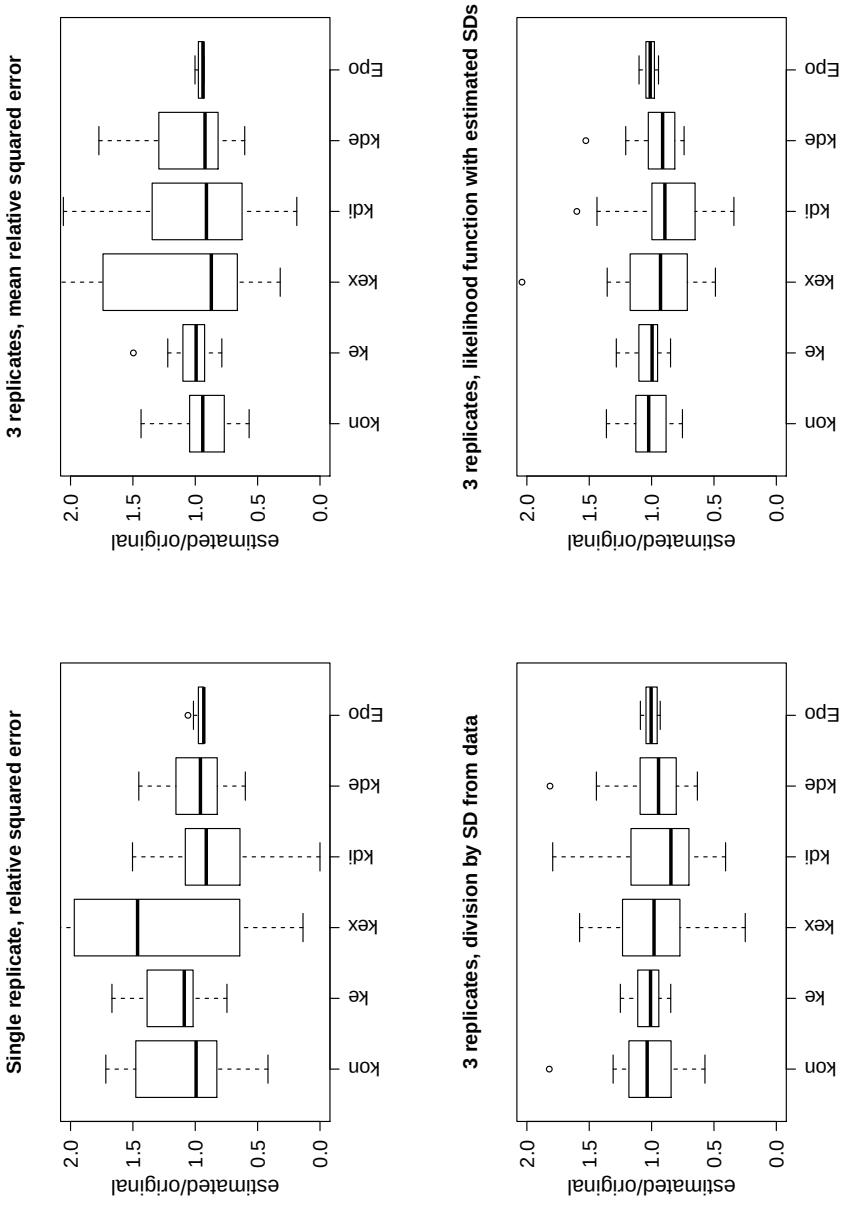

Figure 7.5: Distributions of the estimated quantities of the Epo receptor model with 20% measurement noise using different fitness functions. This figure shows the estimation results with 20 % noise and is otherwise similar to Figure 7.4.

noise (except one outlier) when estimating the standard deviations. The same also applies for estimation with 15% noise (see Figure B.4 in Appendix B). This approach also caused the ranges of 5 quantities to spread markedly beyond ± 50% of the real values with 20% noise, whereas at least two more quantities could clearly not be identified when using the other fitness functions. Calculating the standard deviations from the data prior to estimation often gave rise to results worse than those obtained when applying the mean relative squared error as fitness function.

Several outliers occured for two of the fitness functions with 5% noise (see Figure B.3 in Appendix B). This could be due to all of the optimization runs missing the global optimum for some data sets. With 5% noise the distribution of only one estimated quantity spread clearly beyond ± 50% when using the mean relative squared error as fitness function for the replicative data sets. Compared to that, the estimation of the standard deviations together with the parameters even caused a larger spread here.

For the ERK signaling model estimation of the standard deviations was not distinctly better than using the mean relative squared error as fitness function with 10% noise (see Figure 7.8). This was also not the case for 5% and 15% noise (see Figures B.5 and B.6 in Appendix B). The results when estimating the standard deviations with 5% noise were even slightly worse in comparison to applying the mean relative squared error. In contrast to that, with 20% noise the first approach yielded clearly better estimation results than the latter approach. The results when calculating the standard deviations prior to estimation were worse than those when using the other approaches for all magnitudes of noise.

With 20% noise estimating the standard deviations together with the quantities caused two quantities to spread clearly beyond ± 50% of the real values. The same approach led to just one quantity being clearly unidentifiable with 15% noise (see Figure B.6).

7.8 Summary of the estimation results

In the study with 5% noise non-replicative data sets were often sufficient for deriving good estimates of model parameters. However, the use of replicative data was necessary in order to obtain good estimation results when the data contained higher noise magnitudes. Calculating the standard deviations of the data points before estimation was not better than applying the mean relative squared error as fitness function. This can be explained by the poor precision using a small number of replicates. With a higher number of experimental replicates the situation would probably be improved. A number of replicates much greater than three is, however, often not feasible due to the limited number of data points that can be measured in many experimental setups.

For data with a mean relative noise of at least 10% estimating the standard deviations together with the unknown quantities proved advisable. With 5% noise the results applying that approach were sometimes worse than the results when using the mean relative squared error. The additional quantities to estimate (i.e., the relative noise of each measured quantity) presumably complicate the estimation procedure. New local or global

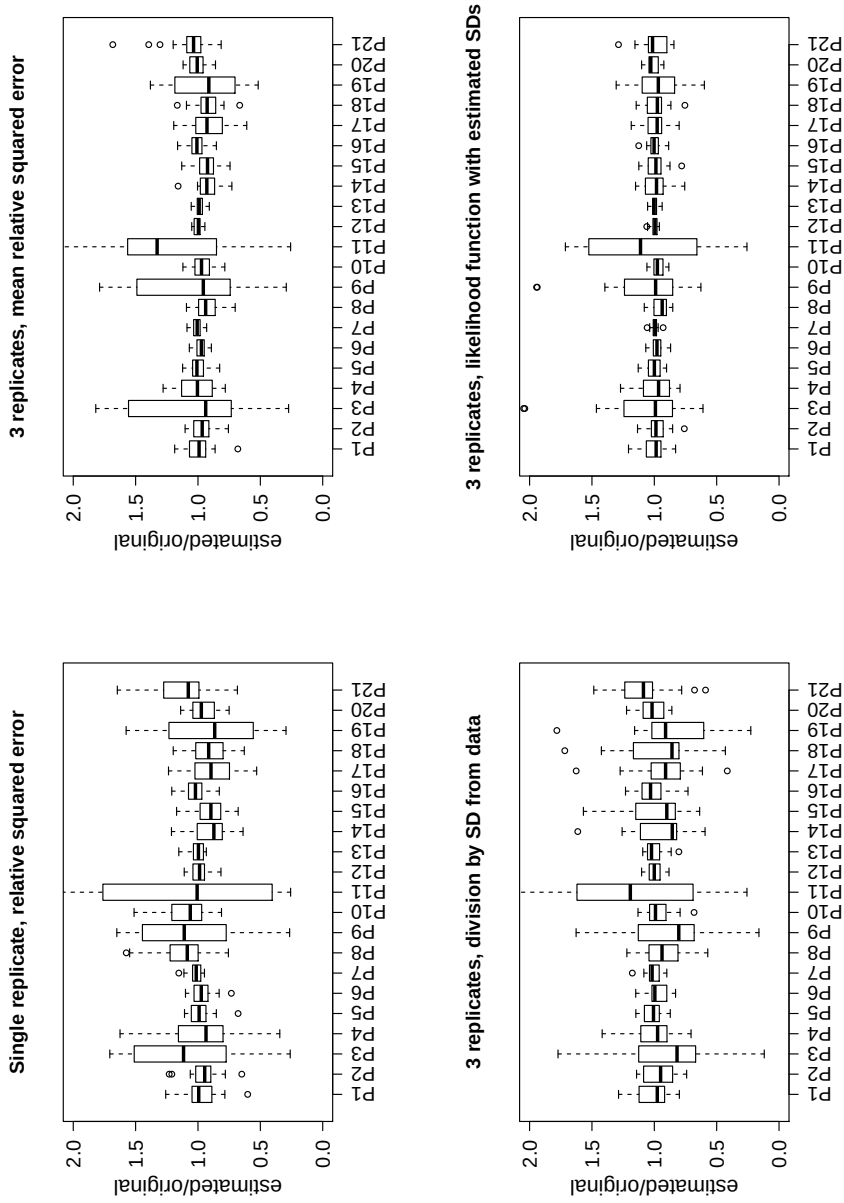

Figure 7.6: Distributions of the estimated quantities of the atorvastatin biotransformation model with 10% measurement noise using different fitness functions. The figure contains the distributions of the quantity estimates of the atorvastatin biotransformation model for 20 noisy data sets using different fitness functions for parameter estimation: relative squared error for single replicates and triplicates (Equation (7.5), Equation (7.6)), division by the standard deviation of each data point in the fitness function (Equation (7.7)), fitness function comprising estimated standard deviations (Equation (7.11)). Short identifiers are applied for the quantities (see Table 7.2). The estimated values were divided by the original quantity values prior to plotting.

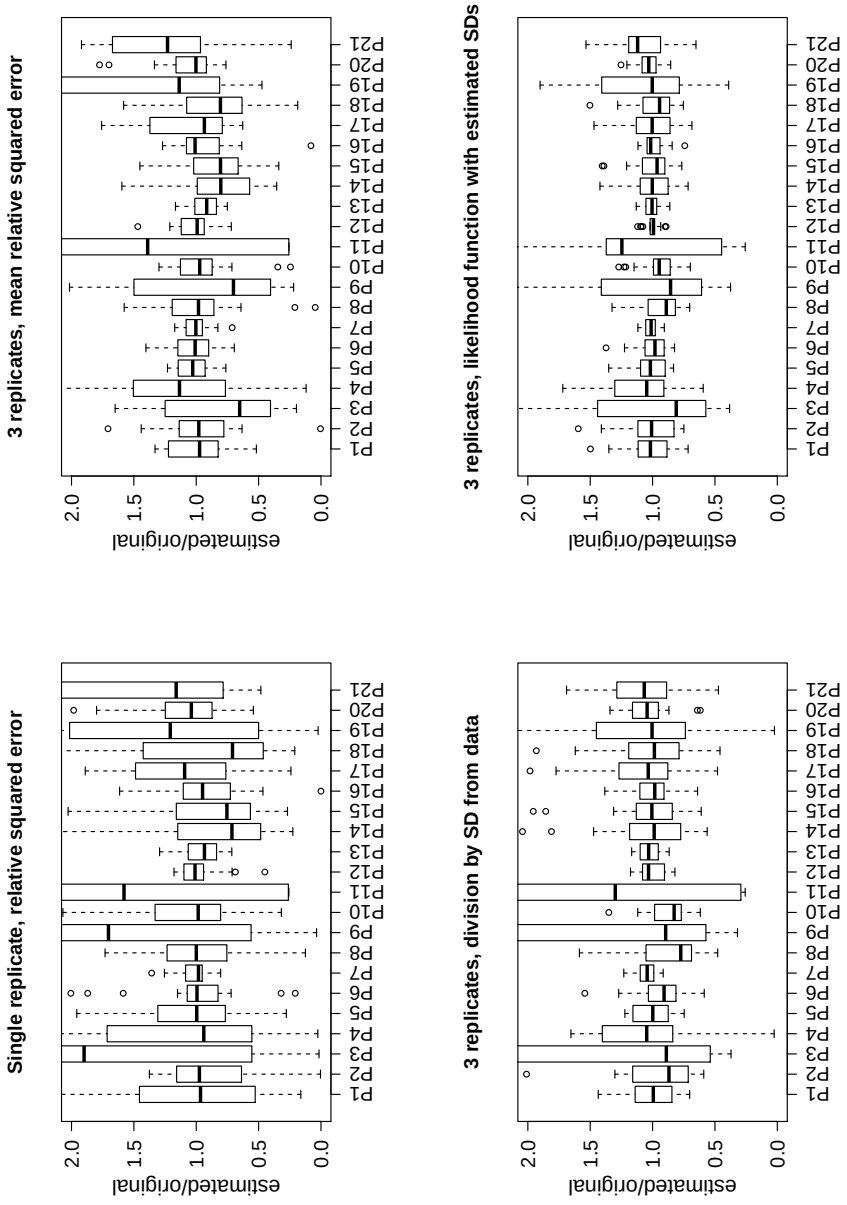

Figure 7.7: Distributions of the estimated quantities of the atorvastatin biotransformation model with 20% measurement noise using different fitness functions. This figure shows the estimation results with 20 % noise and is otherwise similar to Figure 7.6.

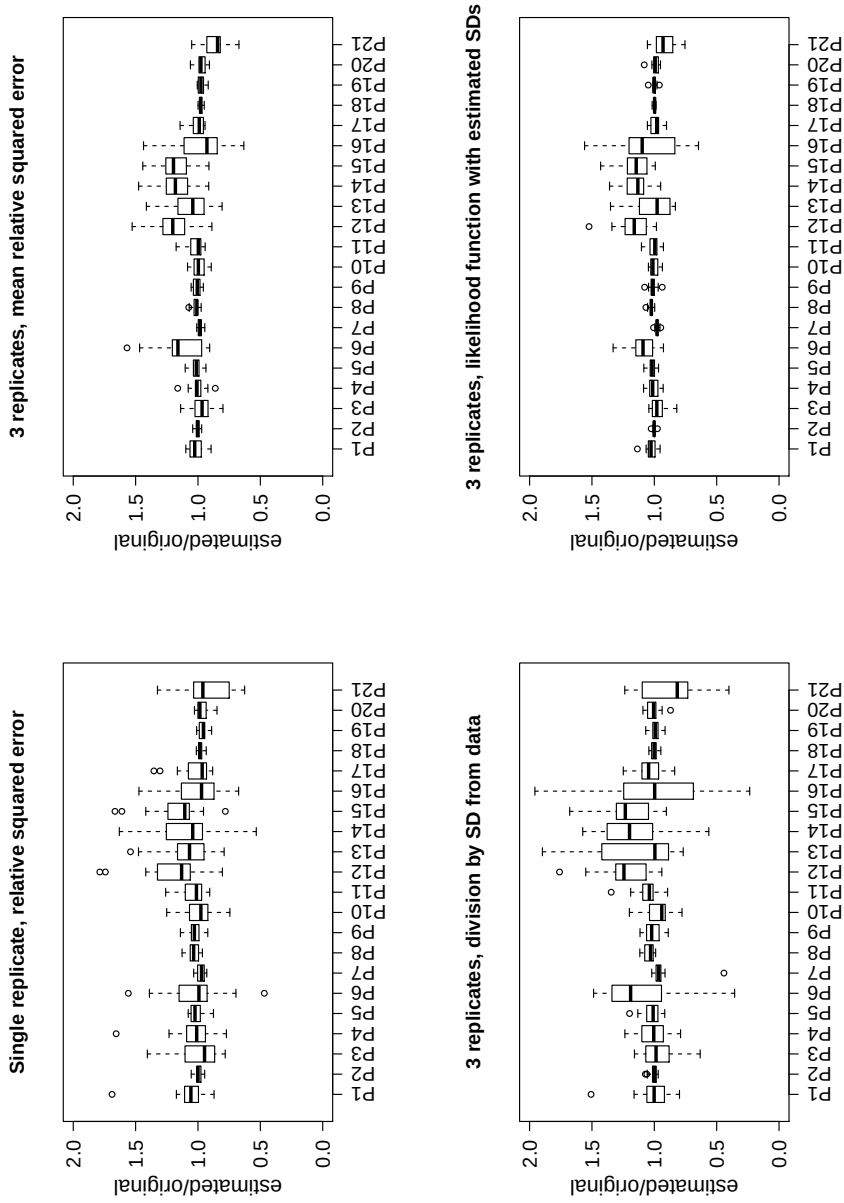

Figure 7.8: Distributions of the estimated quantities of the ERK signaling model with 10% measurement noise using different fitness functions. The figure shows the distributions of the quantity estimates of the ERK signaling model for 20 noisy data sets using different fitness functions for parameter estimation: relative squared error for single replicates and triplicates (Equation (7.5), Equation (7.6)), division by the standard deviation of each data point in the fitness function (Equation (7.7)), fitness function comprising estimated standard deviations (Equation (7.11)). Short identifiers are applied for the quantities (see Table 7.3). The estimated values were divided by the original quantity values before plotting.

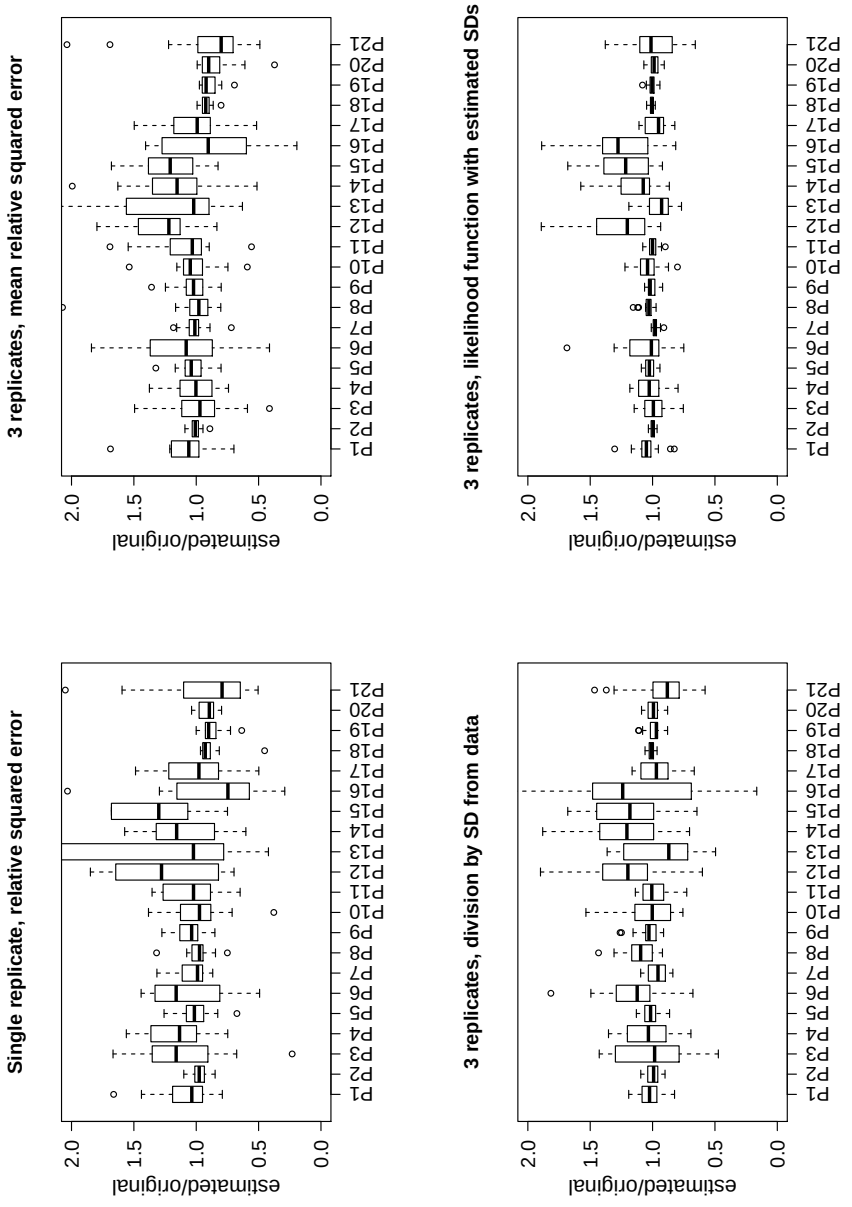

Figure 7.9: Distributions of the estimated quantities of the ERK signaling model with 20% measurement noise using different fitness functions. This figure shows the estimation results with 20 % noise and is otherwise similar to Figure 7.8.

optima might occur in the fitness landscape with a low amount of experimental noise. This effect obviously vanishes with higher noise values.

A quantity contained in one of the models was hard to identify even for 5% noise. This demonstrates that a small amout of noise can already cause unidentifiability of some quantities. The estimation results for such quantities need to be treated carefully.

Except for the mentioned quantity and few outliers, estimation of the quantities in the three models worked well up to 10% noise when using replicative data and estimation of the standard deviations. Applying that approach with 15% noise one quantity was unidentifiable in most models. Increasing the noise level to 20% caused unidentifiability of more quantities. However, the estimation results were still acceptable for most of the quantities.

7.9 Limitations of the study

An assumption of our study was that the data contained only relative noise (i.e., the standard deviation of a data point could be obtained by multiplying a quantity-specific factor by the value of the data point). Some absolute noise independent of the value of the data point is often also present in experimental data, which is why our assumption may be too simplifying in some cases. Investigating the influence of absolute and relative noise in combination (or just absolute noise) on parameter estimation was, however, beyond the scope of this study and could be the target of further research.

Furthermore, we assumed that the measurement noise followed a normal distribution, a frequent assumption for biological data. In contrast to that, a log-normal noise distribution was also assumed previously (Raue *et al.*, 2013) and could lead to results different from those in our study.

The long running times of parameter estimations is one key problem of studies like ours. A parameter estimation run took more than a day for the ERK signaling model with the chosen settings. For the other two models, however, estimation was considerably faster. In order to enable a parameter estimation study within an acceptable time frame, one has to run many parameter estimations in parallel. Therefore, multiple cluster nodes are needed. We only created 20 data sets for each model and magnitude of noise and conducted few estimation runs for each data set, as the running times for estimation were long and the capacity of cluster nodes was limited. The present data sets and estimation results already permitted us to compare the different fitness functions as well as to draw conclusions about the effect of measurement noise on parameter estimation. With more capacity for parallelization available, the study can be repeated increasing the number of data sets and estimation runs.

7.10 Summary and conclusions

The aim of this study was to investigate the effect of different magnitudes of measurement noise on estimation of model quantities (i.e., parameters or initial values of model elements) for three published models. Furthermore, this study aimed at examining in larger scale than previously (Raue *et al.*, 2013), if including the standard deviations of measurement points in the fitness function (by calculating them prior to estimation or by estimating them together with the model quantities) leads to better estimation results.

The study suggests that the computation of the standard deviations before estimation is not advisable for three experimental replicates, which is a common number in biological experiments. However, estimation of the standard deviations (i.e., the relative noise values for each measured quantity) together with the unknown quantities proved to be effective in the presence of a higher magnitude of experimental noise (beginning from 10% noise). This approach enabled the estimation of nearly all unknown quantities with respect to experimental data consisting of three replicates up to a mean relative noise of 15%. With respect to non-replicative data a lower magnitude of noise was tolerated (5 to 10%, depending on the model).

Additionally, this study shows an alternative approach for uncertainty analysis (see 7.1). Noise is added multiple times to data obtained from model simulation using the estimated parameters. In contrast to bootstrapping noisy experimental data, we thus bootstrap "perfect" data. After parameter estimation for each noisy data set, confidence intervals for each parameter can be derived from the estimated parameter values.

Chapter 8

Discussion and concluding remarks

Biochemical network models describe biological behavior and provide hypotheses testable by experiments. This thesis covered methods for the simulation and optimization of such models and their application. The types of models considered were logical models and kinetic ordinary differential equation (ODE) models. While logical modeling is especially applied for describing large biological signaling networks qualitatively, kinetic ODE models represent dynamic network behavior. Kinetic ODE models are usually smaller than logical models and require at least some prior knowledge about mechanistic details.

Logical models are often Boolean models in which the states of the model species can only be 0 or 1. While this restriction is adequate for many signaling pathways, in some cases it prevents a satisfactory description of biological reality. An example for this is the modeling of IL-6 induced hepatic gene regulation conducted in this thesis. In order to circumvent the shortcomings of Boolean models, we used fuzzy logic modeling which allows the species states to be in a continuous interval. A method which was previously employed for the calibration of a network with respect to prior literature knowledge and proteomic data (Morris *et al.*, 2011) was adapted to also enable optimization based on gene expression data. The optimized model suggested the downregulation of RXRα as the main event responsible for the downregulation of many genes encoding drug metabolizing enzymes and transporters (Keller *et al.*, 2016). This hypothesis was supported by further experiments. The described modeling approach is an example how important regulatory events can be obtained from prior knowledge and data. However, the method applied does not produce models describing dynamic behavior.

For the description of the dynamics of biological networks kinetic ODE models are frequently used. In order to exchange and store such models, dedicated formats were developed. SBML (Hucka *et al.*, 2004) is the most important format in this respect. Its main elements are species, compartments, and reactions whose assigned rate laws represent the differential equations. Simulation of SBML models is a difficult task, as these models can also contain rules as well as events definining sudden changes of certain values. Furthermore, different time scales are possible for reactions, which means that the model can contain a fast and a slow subsystem. The SBML Test Suite (Bergmann and Sauro, 2008) comprises numerous benchmark tests which need to be passed by a

simulation tool in order to fully support the SBML standard. Very few tools support the whole standard and their algorithms have not been published. Therefore, we developed the Systems Biology Simulation Core Algorithm and implemented it in the Systems Biology Simulation Core Library (SBSCL) (Keller *et al.*, 2013). The SBSCL comprises several solvers of ODE systems connected with an interpreter of the SBML format. Because of the strict separation between SBML interpretation and numerical simulation, other solvers as well as other model formats can be easily integrated into the library. The SBSCL enables the simulation of SBML models containing all elements defined in the standard. While all models of the SBML Test Suite can be simulated correctly, for simulation of some models our library is not as fast as some other tools, such as COPASI which is implemented in C++. This is mainly due to the more efficient numerical integration routines available in C. However, COPASI does not support SBML completely like the SBSCL. Once more efficient integration routines are implemented in Java, they can be integrated in the SBSCL, which will presumably lead to a smaller running time for several models.

Kinetic ODE models contain parameters whose values need to be set. Values for parameters can be obtained from literature or databases, such as SABIO-RK (Wittig *et al.*, 2012). But those values are often unknown and then need to be estimated with respect to experimental data. This estimation involves repeated simulation with different combinations of parameter values. Due to the large parameter space heuristic optimization routines like evolutionary algorithms are usually applied here. During an estimation the fitness function representing the distance between simulated and experimental data is continuously improved. EvA2 (Kronfeld *et al.*, 2010; Becker and Kronfeld, 2014) is an optimization toolbox that comprises several heuristic optimization routines. As the SBSCL has been specifically designed for a straightforward integration into other software tools, we connected this simulation library with EvA2 into the program SBMLsimulator (Dörr *et al.*, 2014). SBMLsimulator comprises a graphical user interface for the display of the results obtained by simulation with the SBSCL. For large parameter estimation tasks it can also be run in command-line mode, which enables to start long model optimizations on a cluster computer. The correctness of parameter estimation with SBMLsimulator was tested on such a cluster computer using a published model and artificial data.

How well model parameters can be estimated (also referred to as parameter identifiability), depends on the specific model structure, but also on the quality of the experimental data used for model optimization. This quality involves the time points for which measurements are available. Experimental data are always noisy and the magnitude of experimental noise is also likely to have a large influence on parameter estimation. In this thesis we used SBMLsimulator in a large simulation study testing robustness of the parameter estimation against different magnitudes of noise for three published models. Several fitness functions were compared one of them involving estimation of the experimental noise together with the parameters. This approach, which was previously tested for a small toy model (Raue *et al.*, 2013), led for all three models to better estimation

results with data comprising larger magnitudes of noise. A mean relative noise of 20% caused unidentifiability of several quantities for each of the three models. While conclusions for other models are difficult to draw, this study shows that a realistic magnitude of experimental noise can greatly hamper parameter estimation. As absolute noise is usually also present in experimental data, but was for the sake of simplicity not included in this study, the robustness of parameter estimation against experimental noise is expected to be even worse.

In this thesis logical models and ODE models were treated as separate modeling approaches. Logical models, which were not dynamic, were calibrated with respect to data comprising measurements for only a single time point. However, CellNOptR (Terfve *et al.*, 2012) also allows gates to be activate at different discrete time points. Gates active at a later time point could, e.g., represent feedback loops. Such models are then "discrete dynamic", because species states can be computed at several discrete time points. Calibration of these models is possible with respect to time-resolved data. The method for fuzzy logic modeling does currently not support model calibration with respect to time-resolved experimental data, but could be similarly extended. A Boolean model can even be transformed into a fully dynamic ODE model with the Odefy toolbox (Krumsiek *et al.*, 2010). This transformation involves the introduction of parameters whose values need to be known. In order to estimate the parameters, detailed time-resolved experimental data are needed. CellNOptR comprises an implementation of the Odefy method and enables such a parameter estimation.

While the logical modeling covered in this thesis is mainly used for describing signaling or gene regulatory networks, other techniques should be applied for modeling large metabolic pathways. Kinetic ODE modeling of such large networks would require a large amount of data for the estimation of unknown parameters, which limits use of ODE modeling for the investigation of large metabolic pathways (Töpfer *et al.*, 2015). A widely used alternative are structural models, which comprise species and reactions, but no kinetic rate laws. Such models can also be stored in the SBML format. One important example is HepatoNet1, which qualitatively describes large parts of the metabolism of the human liver (Gille *et al.*, 2010). In order to examine the dynamics of structural models, dynamic flux balance analysis (DFBA) is an adequate method. DFBA (Mahadevan *et al.*, 2002) is related the original flux balance analysis (FBA) (Orth *et al.*, 2010), which finds a distribution of fluxes (i.e., velocities) of reactions under the assumption that the amounts of the species in the network are unchanged (the so-called steady state assumption). In contrast to FBA, DFBA does not assume a steady state and predicts species concentrations and flux distributions over time. Thus the dynamics of a metabolic network can be described without estimation of a large number of kinetic parameters like in ODE modeling. Currently SBMLsimulator is extended to support dynamic flux balance analysis.

The Systems Biology Simulation Core Library and SBMLsimulator can be the basis for further research. This might involve the extension of the capabilities of both the library and the simulation software. Newer levels and versions of SBML could necessitate

an update of the SBSCL in order to maintain full support of the standard. The developed simulation and optimization methods can be applied in ODE modeling, for which the parameter estimation study described here was an example. Another obvious application of SBMLsimulator is its use during the optimization of newly constructed kinetic ODE models.

The application of model simulation and optimization methods covered in this thesis could be similar for comparable biological problems. Fuzzy logic modeling has been demonstrated to be useful for the elucidation of regulatory events responsible for down-regulation of genes. Utilization of the respective method for similar questions is advisable. The conducted parameter estimation study shows a way to test robustness of a given ODE modeling approach. Furthermore, it can be concluded from the study that it is recommendable to take experimental noise into account during parameter estimation.

Appendix A

References for the prior knowledge network in Chapter 4

- Agrawal *et al.* (2001)
- Bolotin (2010)
- de Boussac *et al.* (2010)
- Burgermeister *et al.* (2003)
- Castellano and Downward (2011)
- Chang *et al.* (2003)
- Eulenfeld *et al.* (2012)
- Fessing *et al.* (1998)
- Ghose *et al.* (2004)
- Hagihara *et al.* (2005)
- Hayden and Ghosh (2004)
- Hennessy *et al.* (2005)
- Higgins and Hayes (2011)
- Hoesel and Schmid (2013)
- Hoque *et al.* (2012)
- Hösel *et al.* (2009)
- Israel (2010)
- Jover *et al.* (2002)
- Jover *et al.* (2009)
- Kast *et al.* (2002)
- Lee *et al.* (2000)
- Li *et al.* (2002)
- Migita *et al.* (2005)
- PubMed (www.ncbi.nlm.nih.gov/pubmed)
- Petrovic *et al.* (2007)
- Runge-Morris and Kocarek (2009)
- Ryll *et al.* (2011)
- Teng and Piquette-Miller (2005)
- Thomas *et al.* (2013)
- Tolson and Wang (2010)
- TRANSFAC (Matys *et al.*, 2006)
- Le Vee *et al.* (2013)
- Zanger and Schwab (2013)
- Zhao *et al.* (2013)
- Zordoky and El-Kadi (2009)

Appendix B

Additional figures for the parameter estimation study in Chapter 7

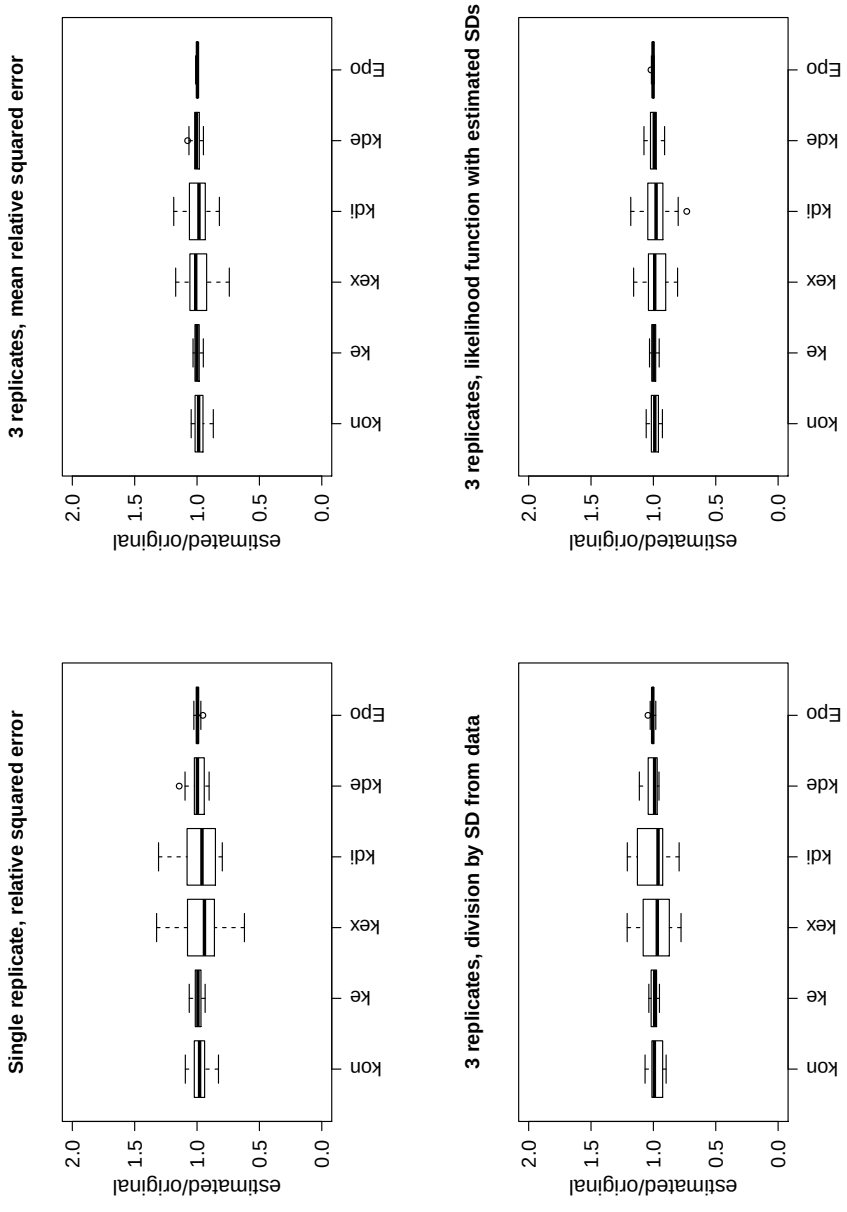

Figure B.1: Distributions of the estimated quantities of the Epo receptor model with 5% measurement noise using different fitness functions. The figure shows the distributions of the quantity estimates of the Epo receptor model for 20 noisy data sets using different fitness functions for parameter estimation: relative squared error for single replicates and triplicates (Equation (7.5), Equation (7.6)), division by the standard deviation of each data point in the fitness function (Equation (7.7)), fitness function comprising estimated standard deviations (Equation (7.11)). The estimated values were divided by the original quantity values before plotting.

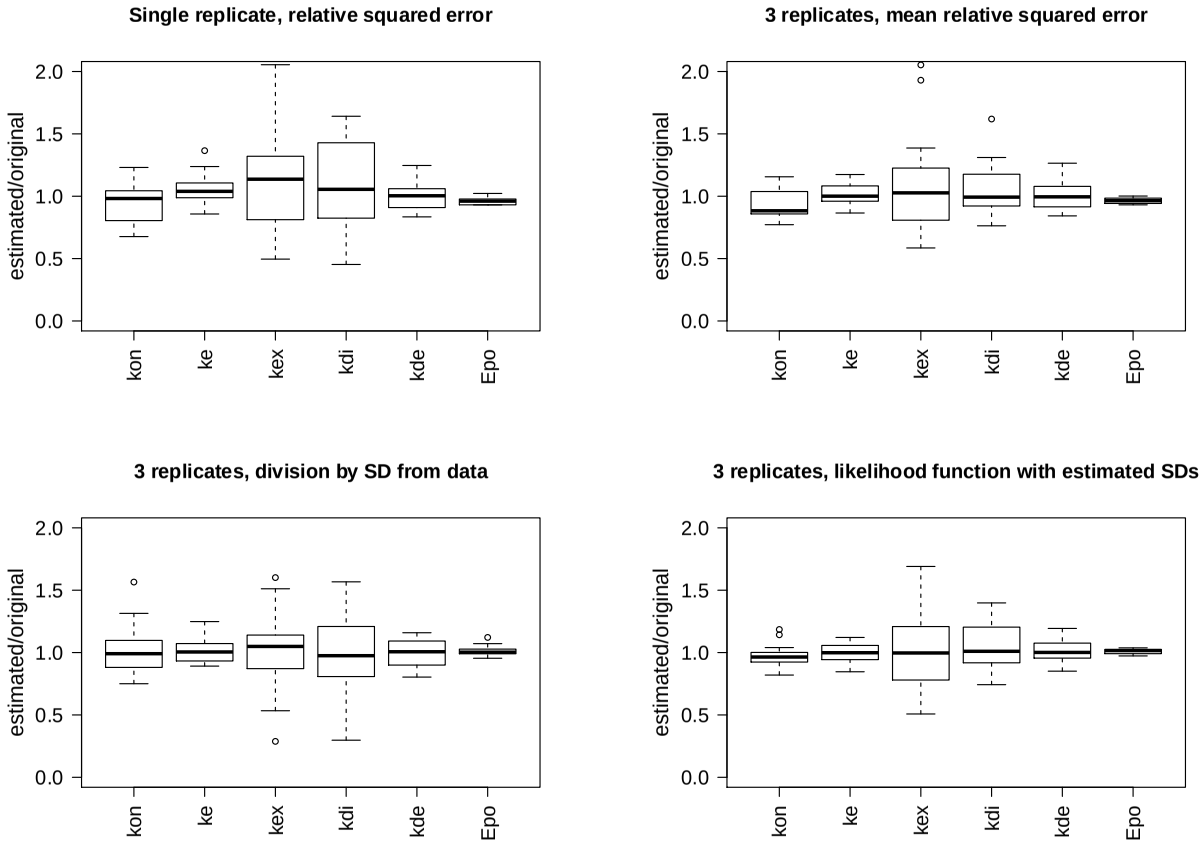

Figure B.2: Distributions of the estimated quantities of the Epo receptor model with 15% measurement noise using different fitness functions. This figure shows the estimation results with 15 % noise and is otherwise similar to Figure B.1.

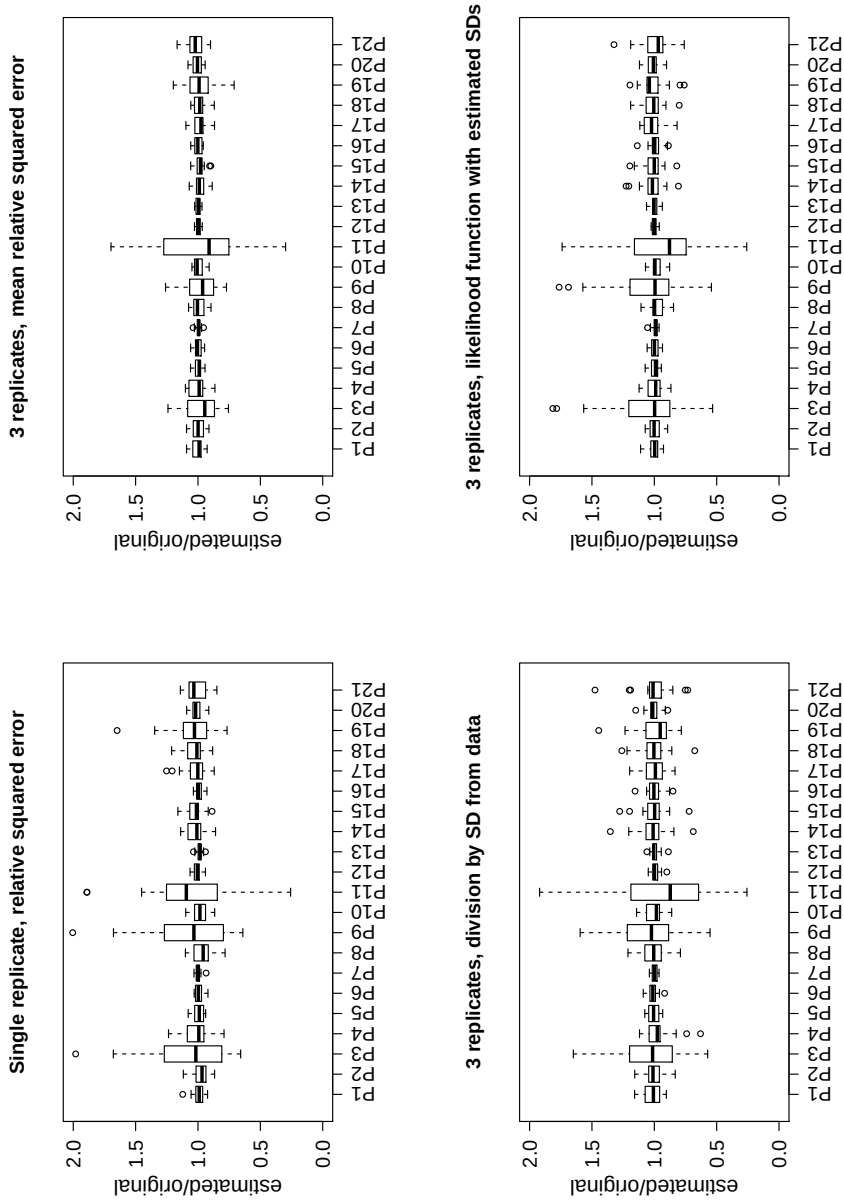

Figure B.3: Distributions of the estimated quantities of the atorvastatin biotransformation model with 5% measurement noise using different fitness functions. The figure contains the distributions of the quantity estimates of the atorvastatin biotransformation model for 20 noisy data sets using different fitness functions for parameter estimation: relative squared error for single replicates and triplicates (Equation (7.5), Equation (7.6)), division by the standard deviation of each data point in the fitness function (Equation (7.7)), fitness function comprising estimated standard deviations (Equation (7.11)). Short identifiers are applied for the quantities (see Table 7.2). The estimated values were divided by the original quantity values prior to plotting.

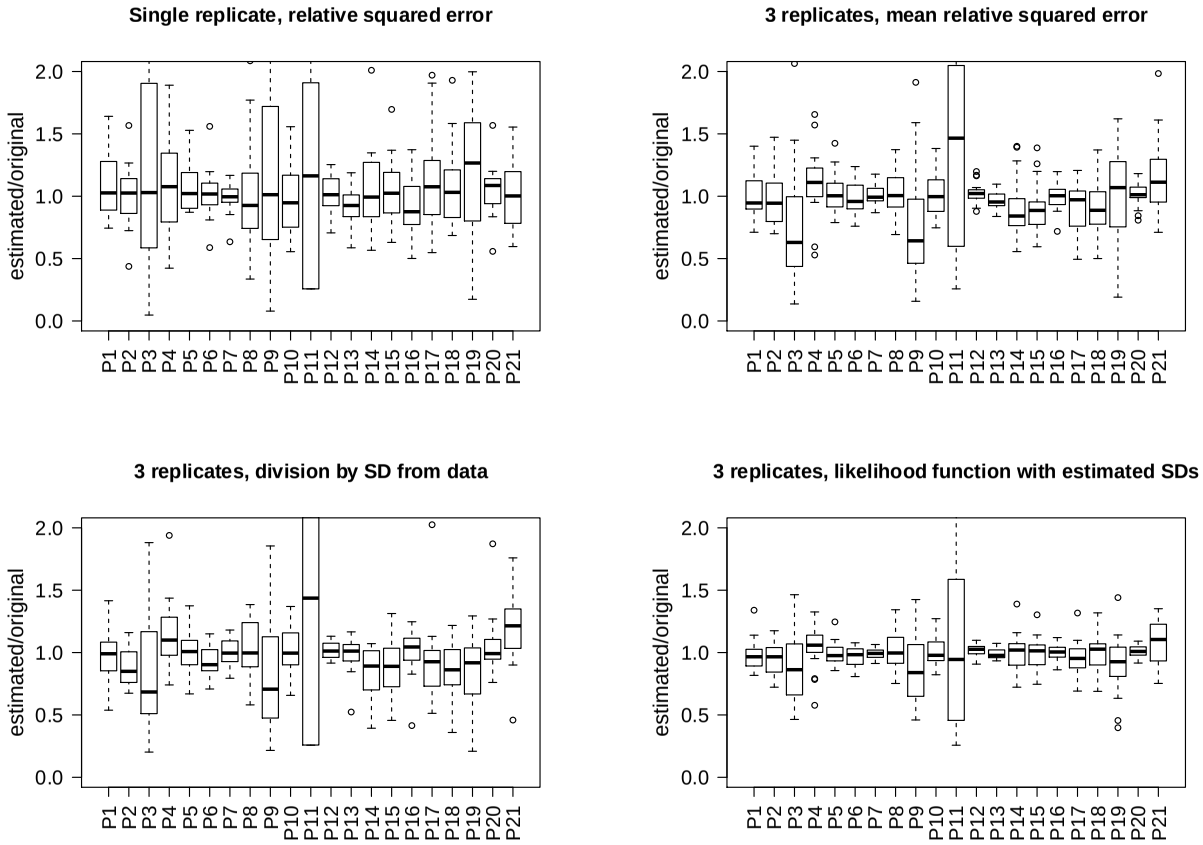

Figure B.4: Distributions of the estimated quantities of the atorvastatin biotransformation model with 15% measurement noise using different fitness functions. This figure shows the estimation results with 15 % noise and is otherwise similar to Figure B.3.

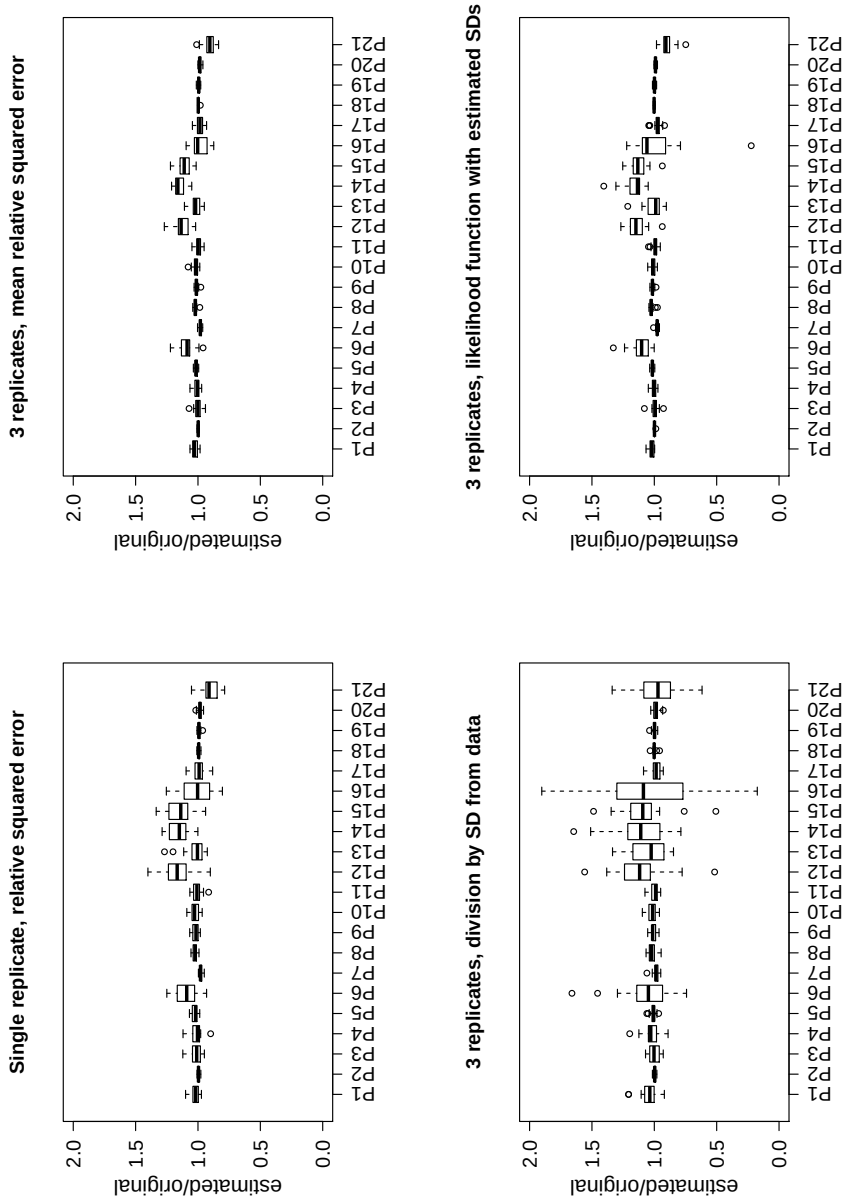

Figure B.5: Distributions of the estimated quantities of the ERK signaling model with 5% measurement noise using different fitness functions. The figure shows the distributions of the quantity estimates of the ERK signaling model for 20 noisy data sets using different fitness functions for parameter estimation: relative squared error for single replicates and triplicates (Equation (7.5), Equation (7.6)), division by the standard deviation of each data point in the fitness function (Equation (7.7)), fitness function comprising estimated standard deviations (Equation (7.11)). Short identifiers are applied for the quantities (see Table 7.3). The estimated values were divided by the original quantity values before plotting.

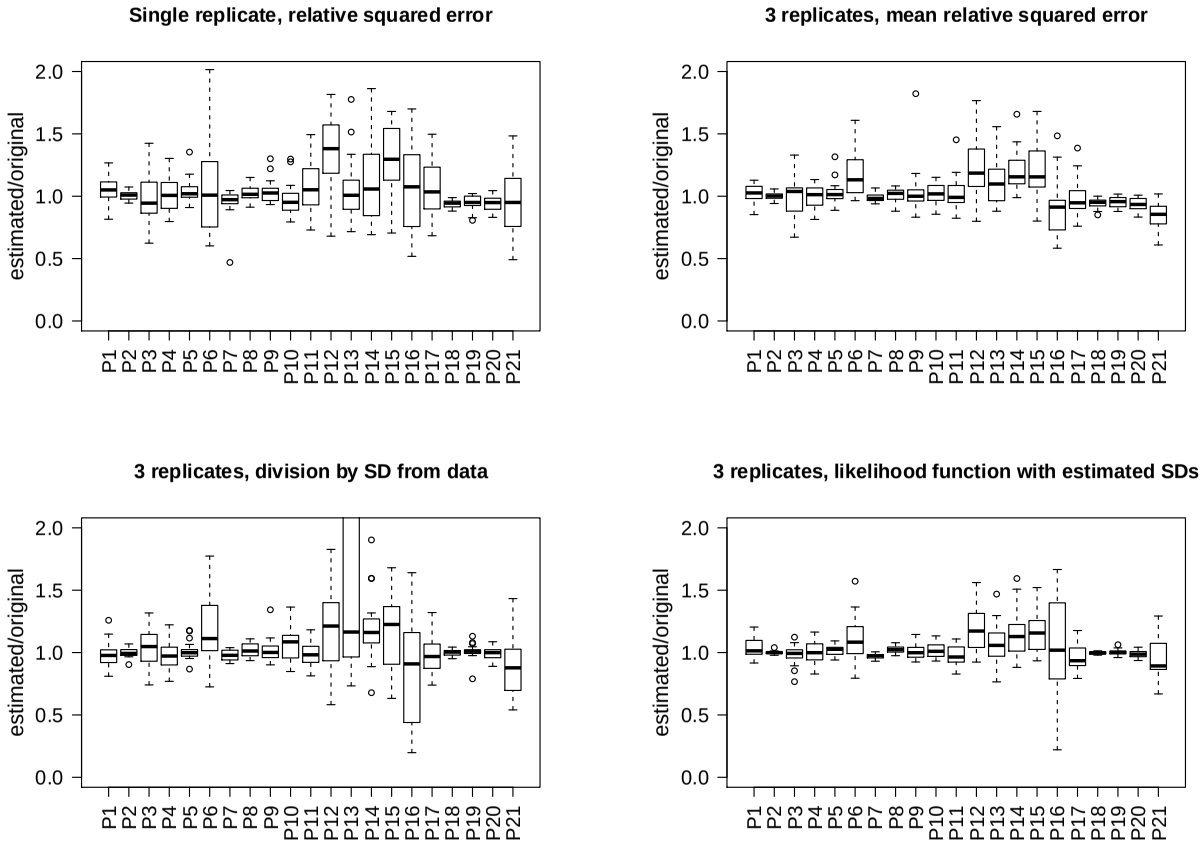

Figure B.6: Distributions of the estimated quantities of the ERK signaling model with 15% measurement noise using different fitness functions. This figure shows the estimation results with 15 % noise and is otherwise similar to Figure B.5.

Symbols

α	Weighting factor in CellNetOptimizer
$\vec{C}$	Vector of sizes of compartments (e.g., the cell)
CR	Crossover probability in differential evolution
$\overline{Ct(G,T)}$	Mean Ct value of the gene G for treatment T
$C(T)$	Control treatment for treatment T
$d(c_1,c_2)$	Distance between two individual cluster elements c_1 and c_2
$D(C_1,C_2)$	Distance between two clusters C_1 and C_2
$\Delta Ct(G,T)$	Ct value for gene G and treatment T normalized to a reference value
$\Delta\Delta Ct(G,T)$	ΔCt value for gene G and treatment T normalized to the respective control treatment $C(T)$
$\text{delay}(e,\tau)$	Delay function for expression e in an SBML model
E	Set of all events in an SBML model
E_A	Set of currently active events in an SBML model
E_I	Set of currently inactive events in an SBML model
F	Parameter for differential evolution
$\vec{f}_E(\vec{Q},t)$	Event in an SBML model for the quantity vector $\vec{Q}$ at time t
$fc(G,T)$	Fold change of a gene G for a treatment T
G	Some gene; generation in an evolutionary algorithm
$\vec{g}(\vec{Q},t)$	Vector of rate rules in an SBML model
h	Step size for model simulation algorithm; coefficient in Hill function
h_{min}	Minimum step size for model simulation algorithm with step size adaptation
J_j	Current velocity of reaction R_j of an SBML model
λ	Specific parameter for differential evolution
m	Midpoint of a Hill function
n	Number of experimental replicates, Hill coefficient
N	Total number of measurements in a data set
n_{ij}	Element in stoichiometry matrix representing the stoichiometry of the species with index i in reaction R_j
$\mathbf{N}$	The stoichiometric matrix of a model
N_s	Number of measured model species in a data set
N_c	Number of experimental condidtions in a data set

$\vec{v}$	Vector of reaction velocities of an ODE model
P	A variant of a logical model obtained from a superstructure
$\vec{p}$	Vector of local parameters in an SBML model
$\vec{P}$	Vector of global parameters in an SBML model
$\vec{Q}$	Vector of quantities (e.g., species and compartments) in an SBML model
$\vec{R}$	Vector of reactions in an ODE model
R_j	Reaction in an SBML model
$\vec{r}(\vec{Q},t)$	Vector of assignment rules (including transformed algebraic rules) in an SBML model
$\vec{S}$	Vector of species in an ODE model
t	The time, value of the student's t-distribution
t_T	A certain time point
T	Experimental treatment of cells with specific substances
τ	Delay for some SBML expression
$\theta(P)$	Fitness of a model variant P
$\theta_f(P)$	Mean squared error between the normalized experimental data and the predicted logical steady states in CellNetOptimizer
$\theta_s(P)$	Penalization term for large models in CellNetOptimizer
u	Vector of parameters obtained after crossover during differential evolution
v	Intermediate parameter vector while creating a new generation in differential evolution
$\mathbf{W}$	Modulation matrix comprising the regulatory influences of the species in an ODE model on the reactions
$\bar{x}$	Mean of the values in x
x_m	Measured data
x_{pred}	Simulated (predicted) data
$x_{best,G}$	Parameter vector with best fitness in generation G of differential evolution
$x_{i,G}$	Some parameter vector in generation G of differential evolution

Abbreviations

ABC	ATP-binding cassette
ADP	Adenosine diphosphate
AhR	Aryl hydrocarbon receptor
ATP	Adenosine triphosphate
API	Application programming interface
APP	Acute phase protein
APR	Acute phase response
BSA	Bovine serum albumin
CAR	Constitutive androstane receptor
cDNA	Complementary DNA
CNO	CellNetOptimizer
CNS	Central nervous system
CRP	C-reactive protein
Ct	Cycle threshold
CYP	Cytochrome P450
DFBA	Dynamic flux balance analysis
DHAP	Dihydroxyacetone phosphate
DAE	Differential algebraic equation
DDE	Delay differential equation
DME	Drug metabolizing enzyme
DMET	Drug metabolizing enzymes and transporters
DMSO	Dimethyl sulfoxide
DNA	Deoxyribonucleic acid
ERK	Extracellular regulated kinase
F1,6BP	Fructose 1,6-bisphosphate
FBA	Flux balance analysis
FC	Fold change
FXR	Farnesoid X receptor
GA3P	Glyceraldehyde 3-phosphate
GAPDH	Glyceraldehyde 3-phosphate dehydrogenase
GR	Glucocorticoid receptor
GUI	Graphical user interface
HNF	Hepatocyte nuclear factor
IKP	Dr. Margarete Fischer-Bosch-Institute of Clinical Pharmacology

IL	Interleukin
IL-6R	Interleukin-6 receptor
JAK	Janus kinase
JAR	Java Archive
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KEGG	Kyoto Encyclopedia of Genes and Genomes
KGML	KEGG Markup Language
KiSAO	Kinetic Simulation Algorithm Ontology
MIASE	Minimum Information About a Simulation Experiment
MSE	Mean squared error
NMI	Natural and Medical Sciences Institute
mRNA	Messenger RNA
LGPL	GNU Lesser General Public License
NAD^+	Nicotinamide adenine dinucleotide
NR	Nuclear receptor
ODE	Ordinary differential equation
PCR	Polymerase chain reaction
qPCR	Quantitative PCR
PBS	Phosphate buffered saline
PHH	Primary human hepatocyte
RuBisCO	Ribulose-1,5-bisphosphate carboxylase oxygenase
RNA	Ribonucleic acid
RPA	Reverse phase microarray
SAA	Serum amyloid A
SBML	Systems Biology Markup language
SBSCA	Systems Biology Simulation Core Algorithm
SBSCL	Systems Biology Simulation Core Library
SED-ML	Simulation Experiment Description Markup Language
SIF	Simple Interaction Format
siRNA	Small interfering RNA
VLN	Virtual Liver Network
XML	Extensible Markup Language

Bibliography

Agrawal, A., Cha-Molstad, H., Samols, D., and Kushner, I. (2001). Transactivation of C-reactive protein by IL-6 requires synergistic interaction of CCAAT/enhancer binding protein beta (C/EBP beta) and rel p50. *Journal of Immunology*, **166**(4), 2378–2384.

Aitken, A. E., Richardson, T. A., and Morgan, E. T. (2006). Regulation of drug-metabolizing enzymes and transporters in inflammation. *Annual Review of Pharmacology and Toxicology*, **46**, 123–149.

Aldridge, B. B., Saez-Rodriguez, J., Muhlich, J. L., Sorger, P. K., and Lauffenburger, D. A. (2009). Fuzzy logic analysis of kinase pathway crosstalk in TNF/EGF/insulin-induced signaling. *PLoS Computational Biology*, **5**(4), e1000340.

Apache Software Foundation (2013). Commons Math: The Apache Commons Mathematics Library. `http://commons.apache.org/proper/commons-math/`.

Arnold, A. and Nikoloski, Z. (2011). A quantitative comparison of Calvin-Benson cycle models. *Trends Plant Sci*, **16**(12), 676–683.

Balsa-Canto, E. and Banga, J. R. (2011). AMIGO, a toolbox for advanced model identification in systems biology using global optimization. *Bioinformatics*, **27**(16), 2311–2313.

Balsa-Canto, E., Alonso, A. A., and Banga, J. R. (2010). An iterative identification procedure for dynamic modeling of biochemical networks. *BMC Systems Biology*, **4**, 11.

Becker, F. and Kronfeld, M. (2014). *EvA2 Documentation*. Centre for Bioinformatics Tübingen, University of Tübingen.

Becker, V., Schilling, M., Bachmann, J., Baumann, U., Raue, A., Maiwald, T., Timmer, J., and Klingmüller, U. (2010). Covering a broad dynamic range: information processing at the erythropoietin receptor. *Science*, **328**(5984), 1404–1408.

Bellu, G., Saccomani, M. P., Audoly, S., and D'Angiò, L. (2007). DAISY: a new software tool to test global identifiability of biological and physiological systems. *Computer Methods and Programs in Biomedicine*, **88**(1), 52–61.

Bergmann, F. (2013). SBML Test Suite Database—Test results for SBML-compatible software systems. `http://sbml.org/Facilities/Database/Simulator`.

Bergmann, F., Shapiro, B. E., and Hucka, M. (2012). SBML Software Matrix (October 8th 2012). `http://sbml.org/SBML_Software_Guide/SBML_Software_Matrix`.

Bergmann, F. T. and Sauro, H. M. (2008). Comparing simulation results of SBML capable simulators. *Bioinformatics*, **24**(17), 1963–1965.

Bernardo-Faura, M., Massen, S., Falk, C. S., Brady, N. R., and Eils, R. (2014). Data-derived modeling characterizes plasticity of MAPK signaling in melanoma. *PLoS computational biology*, **10**(9), e1003795.

Blommaart, E. F., Krause, U., Schellens, J. P., Vreeling-Sindelárová, H., and Meijer, A. J. (1997). The phosphatidylinositol 3-kinase inhibitors wortmannin and LY294002 inhibit autophagy in isolated rat hepatocytes. *European Journal of Biochemistry / FEBS*, **243**(1-2), 240–246.

Bolotin, E. (2010). HNF4α. *Transcription Factor Encyclopedia.*

Bornstein, B. J., Keating, S. M., Jouraku, A., and Hucka, M. (2008). LibSBML: an API library for SBML. *Bioinformatics*, **24**(6), 880–881.

Braeuning, A., Heubach, Y., Knorpp, T., Kowalik, M. A., Templin, M., Columbano, A., and Schwarz, M. (2011). Gender-Specific Interplay of Signaling through β-Catenin and CAR in the Regulation of Xenobiotic-Induced Hepatocyte Proliferation. *Toxicological Sciences*, **123**(1), 113–122.

Büchel, F., Rodriguez, N., Swainston, N., Wrzodek, C., Czauderna, T., Keller, R., Mittag, F., Schubert, M., Glont, M., Golebiewski, M., van Iersel, M., Keating, S., Rall, M., Wybrow, M., Hermjakob, H., Hucka, M., Kell, D. B., Müller, W., Mendes, P., Zell, A., Chaouiya, C., Saez-Rodriguez, J., Schreiber, F., Laibe, C., Dräger, A., and Le Novère, N. (2013). Path2Models: large-scale generation of computational models from biochemical pathway maps. *BMC Systems Biology*, **7**, 116.

Bucher, J., Riedmaier, S., Schnabel, A., Marcus, K., Vacun, G., Weiss, T. S., Thasler, W. E., Nüssler, A. K., Zanger, U. M., and Reuss, M. (2011). A systems biology approach to dynamic modeling and inter-subject variability of statin pharmacokinetics in human hepatocytes. *BMC Systems Biology*, **5**, 66.

Burgermeister, E., Lanzendoerfer, M., and Scheuer, W. (2003). Comparative analysis of docking motifs in MAP-kinases and nuclear receptors. *Journal of Biomolecular Structure & Dynamics*, **20**(5), 623–634.

Campbell, J. S., Prichard, L., Schaper, F., Schmitz, J., Stephenson-Famy, A., Rosenfeld, M. E., Argast, G. M., Heinrich, P. C., and Fausto, N. (2001). Expression of suppressors of cytokine signaling during liver regeneration. *Journal of Clinical Investigation*, **107**(10), 1285–1292.

Carlisle, D., Ion, P., Miner, R., and Poppelier, N. (2001). Mathematical Markup Language (MathML) 2.0. Technical report.

Castellano, E. and Downward, J. (2011). RAS interaction with PI3K. *Genes & Cancer*, **2**(3), 261–274. 00112 PMID: 21779497.

Chang, F., Lee, J. T., Navolanic, P. M., Steelman, L. S., Shelton, J. G., Blalock, W. L., Franklin, R. A., and McCubrey, J. A. (2003). Involvement of PI3K/Akt pathway in cell cycle progression, apoptosis, and neoplastic transformation: a target for cancer chemotherapy. *Leukemia*, **17**(3), 590–603.

Chaouiya, C., Keating, S. M., Berenguier, D., Naldi, A., Thieffry, D., van Iersel, M. P., and Helicar, T. (2013a). Qualitative Models. Technical report.

Chaouiya, C., Bérenguier, D., Keating, S. M., Naldi, A., van Iersel, M. P., Rodriguez, N., Dräger, A., Büchel, F., Cokelaer, T., Kowal, B., Wicks, B., Gonçalves, E., Dorier, J., Page, M., Monteiro, P. T., von Kamp, A., Xenarios, I., de Jong, H., Hucka, M., Klamt, S., Thieffry, D., Le Novère, N., Saez-Rodriguez, J., and Helikar, T. (2013b). SBML qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools. *BMC Systems Biology*, **7**, 135.

Chelliah, V., Laibe, C., and Le Novère, N. (2013). BioModels Database: A Repository of Mathematical Models of Biological Processes. In *In Silico Systems Biology*, volume 1021 of *Methods in Molecular Biology*, pages 189–199. Springer.

Chen, N., del, V. I., Kyriakopoulos, S., Polizzi, K., and Kontoravdi, C. (2012). Metabolic network reconstruction: advances in in silico interpretation of analytical information. *Curr Opin Biotechnol*, **23**, 77–82.

Clerc, M. (2005). *Particle Swarm Optimization*. ISTE Ltd, London, UK.

Clerc, M. and Kennedy, J. (2002). The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, **6**(1), 58–73.

Congiu, M., Mashford, M. L., Slavin, J. L., and Desmond, P. V. (2009). Coordinate regulation of metabolic enzymes and transporters by nuclear transcription factors in human liver disease. *Journal of Gastroenterology and Hepatology*, **24**(6), 1038–1044.

Cooling, M. T. (2010). A Primer on Modular Mass-Action Modelling with CellML. In *Systems Biology for Signaling Networks*, volume 1 of *Systems Biology*, pages 721–750. Springer.

Costa, R. S., Machado, D., Rocha, I., and Ferreira, E. C. (2011). Critical perspective on the consequences of the limited availability of kinetic data in metabolic dynamic modelling. *IET Systems Biology*, **5**(3), 157–163.

Courtot, M., Juty, N., Knüpfer, C., Waltemath, D., Zhukova, A., Dräger, A., Dumontier, M., Finney, A., Golebiewski, M., Hastings, J., Hoops, S., Keating, S., Kell, D. B., Kerrien, S., Lawson, J., Lister, A., Lu, J., Machne, R., Mendes, P., Pocock, M., Rodriguez, N., Villéger, A., Wilkinson, D. J., Wimalaratne, S., Laibe, C., Hucka, M., and Le Novère, N. (2011). Controlled vocabularies and semantics in systems biology. *Molecular Systems Biology*, **7**, 543.

Cox, A. D. and Der, C. J. (2002). Ras family signaling: therapeutic targeting. *Cancer Biology & Therapy*, **1**(6), 599–606.

Cray, C., Zaias, J., and Altman, N. H. (2009). Acute phase response in animals: A review. *Comparative Medicine*, **59**(6), 517–526.

Cuellar, A., Nielsen, P., Halstead, M., Bullivant, D., Nickerson, D., Hedley, W., Nelson, M., and Lloyd, C. (2006). CellML 1.1 Specification. Technical report, Bioengineering Institute, University of Auckland.

de Boussac, H., Ratajewski, M., Sachrajda, I., Köblös, G., Tordai, A., Pulaski, L., Buday, L., Váradi, A., and Arányi, T. (2010). The ERK1/2-hepatocyte nuclear factor 4alpha axis regulates human ABCC6 gene expression in hepatocytes. *The Journal of Biological Chemistry*, **285**(30), 22800–22808.

Delhase, M., Li, N., and Karin, M. (2000). Signalling pathways: Kinase regulation in inflammatory response. *Nature*, **406**(6794), 367–368.

Dorigo, M., Birattari, M., and Stützle, T. (2006). Ant colony optimization – artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, **1**, 28–39.

Dörr, A., Keller, R., Zell, A., and Dräger, A. (2014). SBMLsimulator: a Java tool for model simulation and parameter estimation in systems biology. *Computation*, **2**(4), 246–257.

Dräger, A. (2011). *Computational Modeling of Biochemical Networks*. Verlag Dr. Hut, Sternstraße 18, München, Eberhard Karls University of Tübingen.

Dräger, A. and Planatscher, H. (2013). *Encyclopedia of Systems Biology*, chapter Parameter Estimation, Metabolic Network Modeling, pages 1627–1631. Springer-Verlag, Springer New York Heidelberg Dorodrecht London.

Dräger, A., Kronfeld, M., Supper, J., Planatscher, H., Magnus, J. B., Oldiges, M., and Zell, A. (2007a). Benchmarking Evolutionary Algorithms on Convenience Kinetics Models of the Valine and Leucine Biosynthesis in *C. glutamicum*. In *2007 IEEE Congress on Evolutionary Computation*, pages 896–903, Singapore. IEEE Computational Intelligence Society, IEEE Press.

Dräger, A., Supper, J., Planatscher, H., Magnus, J. B., Oldiges, M., and Zell, A. (2007b). Comparing Various Evolutionary Algorithms on the Parameter Optimization of the Valine and Leucine Biosynthesis in *Corynebacterium glutamicum*. In *2007 IEEE Congress on Evolutionary Computation*, pages 620–627, Singapore. IEEE Computational Intelligence Society, IEEE Press.

Dräger, A., Hassis, N., Supper, J., Schröder, A., and Zell, A. (2008). SBMLsqueezer: a CellDesigner plug-in to generate kinetic rate equations for biochemical networks. *BMC Systems Biology*, **2**(1), 39.

Dräger, A., Kronfeld, M., Ziller, M. J., Supper, J., Planatscher, H., Magnus, J. B., Oldiges, M., Kohlbacher, O., and Zell, A. (2009). Modeling metabolic networks in *C. glutamicum*: a comparison of rate laws in combination with various parameter optimization strategies. *BMC Systems Biology*, **3**, 5.

Dräger, A., Rodriguez, N., Dumousseau, M., Dörr, A., Wrzodek, C., Le Novère, N., Zell, A., and Hucka, M. (2011). JSBML: a flexible Java library for working with SBML. *Bioinformatics*, **27**(15), 2167–2168.

Dräger, A., Zielinski, D. C., Keller, R., Rall, M., Eichner, J., Palsson, B. O., and Zell, A. (2015). SBMLsqueezer 2: context-sensitive creation of kinetic equations in biochemical networks. *BMC Systems Biology*, **9**, 68.

Eduati, F., De Las Rivas, J., Di Camillo, B., Toffolo, G., and Saez-Rodriguez, J. (2012). Integrating literature-constrained and data-driven inference of signalling networks. *Bioinformatics*, **28**(18), 2311–2317.

Eulenfeld, R., Dittrich, A., Khouri, C., Müller, P. J., Mütze, B., Wolf, A., and Schaper, F. (2012). Interleukin-6 signalling: more than jaks and STATs. *European Journal of Cell Biology*, **91**(6-7), 486–495.

Favata, M. F., Horiuchi, K. Y., Manos, E. J., Daulerio, A. J., Stradley, D. A., Feeser, W. S., Dyk, D. E. V., Pitts, W. J., Earl, R. A., Hobbs, F., Copeland, R. A., Magolda, R. L., Scherle, P. A., and Trzaskos, J. M. (1998). Identification of a novel inhibitor

of mitogen-activated protein kinase kinase. *Journal of Biological Chemistry*, **273**(29), 18623–18632.

Fehlberg, E. (1970). Klassische Runge-Kutta-Formeln vierter und niedrigerer Ordnung mit Schrittweiten-Kontrolle und ihre Anwendung auf Wärmeleitungsprobleme. *Computing*, **6**(1-2), 61–71.

Fessing, M. Y., Krynetski, E. Y., Zambetti, G. P., and Evans, W. E. (1998). Functional characterization of the human thiopurine S-methyltransferase (TPMT) gene promoter. *European Journal of Biochemistry / FEBS*, **256**(3), 510–517.

Finney, A. and Hucka, M. (2003). Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions. Technical report, Systems Biology Workbench Development Group JST ERATO Kitano Symbiotic Systems Project Control and Dynamical Systems, MC 107-81, California Institute of Technology.

Finney, A., Hucka, M., and Le Novère, N. (2006). Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions. Technical report.

Freitas, A. A. (2007). A review of evolutionary algorithms for data mining. In *Soft Computing for Knowledge Discovery and Data Mining*, pages 61–93.

Funahashi, A., Tanimura, N., Morohashi, M., and Kitano, H. (2003). CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *BioSilico*, **1**(5), 159–162.

Gauges, R., Rost, U., Sahle, S., and Wegner, K. (2006). A model diagram layout extension for SBML. *Bioinformatics*, **22**(15), 1879–1885.

Ghose, R., Zimmerman, T. L., Thevananther, S., and Karpen, S. J. (2004). Endotoxin leads to rapid subcellular re-localization of hepatic RXRα: A novel mechanism for reduced hepatic gene expression in inflammation. *Nuclear Receptor*, **2**, 4.

Gille, C., Bölling, C., Hoppe, A., Bulik, S., Hoffmann, S., Hübner, K., Karlstädt, A., Ganeshan, R., König, M., Rother, K., Weidlich, M., Behre, J., and Holzhütter, H.-G. (2010). HepatoNet1: a comprehensive metabolic reconstruction of the human hepatocyte for the analysis of liver physiology. *Molecular Systems Biology*, **6**, 411.

Godoy, P., Hewitt, N., Albrecht, U., Andersen, M., Ansari, N., Bhattacharya, S., Bode, J., Bolleyn, J., Borner, C., Böttger, J., Braeuning, A., Budinsky, R., Burkhardt, B., Cameron, N., Camussi, G., Cho, C.-S., Choi, Y.-J., Craig Rowlands, J., Dahmen, U., Damm, G., Dirsch, O., Donato, M., Dong, J., Dooley, S., Drasdo, D., Eakins, R., Ferreira, K., Fonsato, V., Fraczek, J., Gebhardt, R., Gibson, A., Glanemann, M., Goldring, C., Gómez-Lechón, M., Groothuis, G., Gustavsson, L., Guyot, C., Hallifax, D., Hammad, S., Hayward, A., Häussinger, D., Hellerbrand, C., Hewitt, P., Hoehme,

S., Holzhütter, H.-G., Houston, Hrach, J., Ito, K., Jaeschke, H., Keitel, V., Kelm, J., Kevin Park, B., Kordes, C., Kullak-Ublick, G., LeCluyse, E., Lu, P., Luebke-Wheeler, J., Lutz, A., Maltman, D., Matz-Soja, M., McMullen, P., Merfort, I., Messner, S., Meyer, C., Mwinyi, J., Naisbitt, D., Nussler, A., Olinga, P., Pampaloni, F., Pi, J., Pluta, L., Przyborski, S., Ramachandran, A., Rogiers, V., Rowe, C., Schelcher, C., Schmich, K., Schwarz, M., Singh, B., Stelzer, E., Stieger, B., Stöber, R., Sugiyama, Y., Tetta, C., Thasler, W., Vanhaecke, T., Vinken, M., Weiss, T., Widera, A., Woods, C., Xu, J., Yarborough, K., and Hengstler, J. (2013). *Recent advances in 2D and 3D in vitro systems using primary hepatocytes, alternative hepatocyte sources and non-parenchymal liver cells and their use in investigating mechanisms of hepatotoxicity, cell signaling and ADME*, volume 87, pages 1315–1530. Springer Berlin Heidelberg.

Goueli, S. A., Hsiao, K., Lu, T., and Simposn, D. (1998). U0126: A novel, selective and potent inhibitor of MAP kinase kinase (MEK). *Promega Notes*, (69), 6.

Gruys, E., Toussaint, M., Niewold, T., and Koopmans, S. (2005). Acute phase reaction and acute phase proteins. *Journal of Zhejiang University. Science. B*, **6**(11), 1045–1056.

Gu, X., Ke, S., Liu, D., Sheng, T., Thomas, P. E., Rabson, A. B., Gallo, M. A., Xie, W., and Tian, Y. (2006). Role of NF-kappaB in regulation of PXR-mediated gene expression: a mechanism for the suppression of cytochrome P-450 3A4 by proinflammatory agents. *The Journal of Biological Chemistry*, **281**(26), 17882–17889.

Guillén-Gosálbez, G., Miró, A., Alves, R., Sorribas, A., and Jiménez, L. (2013). Identification of regulatory structure and kinetic parameters of biochemical networks via mixed-integer dynamic optimization. *BMC Systems Biology*, **7**, 113.

Hagihara, K., Nishikawa, T., Sugamata, Y., Song, J., Isobe, T., Taga, T., and Yoshizaki, K. (2005). Essential role of STAT3 in cytokine-driven NF-kappaB-mediated serum amyloid a gene expression. *Genes to Cells: Devoted to Molecular & Cellular Mechanisms*, **10**(11), 1051–1063.

Hairer, E., Nørsett, S. P., and Wanner, G. (2000). *Solving ordinary differential equations. 1, Nonstiff problems*. Springer, Berlin, Germany.

Hartigan, J. A. (1975). *Clustering Algorithms*. John Wiley & Sons Inc., New York.

Harvey, R. D. and Morgan, E. T. (2014). Cancer, inflammation, and therapy: Effects on cytochrome P450–mediated drug metabolism and implications for novel immunotherapeutic agents. *Clinical Pharmacology & Therapeutics*, **96**(4), 449–457.

Hayden, M. S. and Ghosh, S. (2004). Signaling to NF-κB. *Genes & Development*, **18**(18), 2195 –2224.

Hecker, M., Lambeck, S., Toepfer, S., van Someren, E., and Guthke, R. (2009). Gene regulatory network inference: data integration in dynamic models-a review. *Bio Systems*, **96**(1), 86–103.

Heid, C. A., Stevens, J., Livak, K. J., and Williams, P. M. (1996). Real time quantitative PCR. *Genome Research*, **6**(10), 986–994.

Heinrich, R. and Schuster, S. (1996). *The Regulation Of Cellular Systems*. Springer.

Hennessy, B. T., Smith, D. L., Ram, P. T., Lu, Y., and Mills, G. B. (2005). Exploiting the PI3K/Akt pathway for cancer drug discovery. *Nature Reviews. Drug Discovery*, **4**(12), 988–1004.

Hettling, H. and van Beek, J. H. G. M. (2011). Analyzing the functional properties of the creatine kinase system with multiscale 'sloppy' modeling. *PLoS Computational Biology*, **7**(8), e1002130.

Higgins, L. G. and Hayes, J. D. (2011). Mechanisms of induction of cytosolic and microsomal glutathione transferase (GST) genes by xenobiotics and pro-inflammatory agents. *Drug Metabolism Reviews*, **43**(2), 92–137.

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., and Woodward, C. S. (2005). SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM T Math Software*, **31**(3), 363–396.

Hoesel, B. and Schmid, J. A. (2013). The complexity of NF-κB signaling in inflammation and cancer. *Molecular Cancer*, **12**(1), 86.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA.

Holzhütter, H.-G., Drasdo, D., Preusser, T., Lippert, J., and Henney, A. M. (2012). The virtual liver: a multidisciplinary, multilevel challenge for systems biology. *Wiley Interdiscip Rev Syst Biol Med*, **4**(3), 221–235.

Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., and Kummer, U. (2006). COPASI–a COmplex PAthway SImulator. *Bioinformatics*, **22**(24), 3067–3074.

Hopcroft, J. E. and Karp, R. M. (1973). An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J Comput*, **2**(4), 225–231.

Hoque, M. T., Robillard, K. R., and Bendayan, R. (2012). Regulation of breast cancer resistant protein by peroxisome proliferator-activated receptor α in human brain microvessel endothelial cells. *Molecular pharmacology*, **81**(4), 598–609.

Hösel, M., Quasdorff, M., Wiegmann, K., Webb, D., Zedler, U., Broxtermann, M., Tedjokusumo, R., Esser, K., Arzberger, S., Kirschning, C. J., Langenkamp, A., Falk, C., Büning, H., Rose-John, S., and Protzer, U. (2009). Not interferon, but interleukin-6 controls early gene expression in hepatitis B virus infection. *Hepatology (Baltimore, Md.)*, **50**(6), 1773–1782.

Hucka, M., Finney, A., Sauro, H., and Bolouri, H. (2003). Systems Biology Markup Language (SBML) Level 1: Structures and Facilities for Basic Model Definitions. Technical Report 2, Systems Biology Workbench Development Group JST ERATO Kitano Symbiotic Systems Project Control and Dynamical Systems, MC 107-81, California Institute of Technology, Pasadena, CA, USA.

Hucka, M., Finney, A., Bornstein, B. J., Keating, S. M., Shapiro, B. E., Matthews, J., Kovitz, B. L., Schilstra, M. J., Funahashi, A., Doyle, J. C., and Kitano, H. (2004). Evolving a lingua franca and associated software infrastructure for computational systems biology: the Systems Biology Markup Language (SBML) project. *Syst Biol*, **1**(1), 41–53.

Hucka, M., Finney, A., Hoops, S., Keating, S. M., and Le Novère, N. (2008). Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions. Technical report, Nature Precedings.

Hucka, M., Bergmann, F. T., Hoops, S., Keating, S. M., Sahle, S., Schaff, J. C., Smith, L., and Wilkinson, D. J. (2010). The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core. Technical report, Nature Precedings.

Hug, S., Raue, A., Hasenauer, J., Bachmann, J., Klingmüller, U., Timmer, J., and Theis, F. J. (2013). High-dimensional bayesian parameter estimation: case study for a model of JAK2/STAT5 signaling. *Mathematical Biosciences*, **246**(2), 293–304.

Israel, A. (2010). The IKK complex, a central regulator of NF-κb activation. *Cold Spring Harbor Perspectives in Biology*, **2**(3).

Jancova, P., Anzenbacher, P., and Anzenbacherova, E. (2010). Phase II drug metabolizing enzymes. *Biomedical Papers of the Medical Faculty of the University Palacký, Olomouc, Czechoslovakia*, **154**(2), 103–116.

Janes, K. A. and Lauffenburger, D. A. (2013). Models of signalling networks - what cell biologists can gain from them and give to them. *Journal of Cell Science*, **126**(Pt 9), 1913–1921.

Johnson, K. A. and Goody, R. S. (2011). The Original Michaelis Constant: Translation of the 1913 Michaelis–Menten Paper. *Biochemistry*, **50**(39), 8264–8269.

Johnson, S. G. (2014). The nlopt nonlinear-optimization package. `http://ab-initio.mit.edu/nlopt`.

Joshi, M., Seidel-Morgenstern, A., and Kremling, A. (2006). Exploiting the bootstrap method for quantifying parameter confidence intervals in dynamical systems. *Metabolic Engineering*, **8**(5), 447–455.

Jover, R., Bort, R., Gómez-Lechón, M. J., and Castell, J. V. (2002). Down-regulation of human CYP3A4 by the inflammatory signal interleukin-6: molecular mechanism and transcription factors involved. *FASEB journal: official publication of the Federation of American Societies for Experimental Biology*, **16**(13), 1799–1801.

Jover, R., Moya, M., and Gómez-Lechón, M. J. (2009). Transcriptional regulation of cytochrome P450 genes by the nuclear receptor hepatocyte nuclear factor 4-alpha. *Current Drug Metabolism*, **10**(5), 508–519.

Kanehisa, M. and Goto, S. (2000). KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, **28**(1), 27–30.

Kanehisa, M., Goto, S., Sato, Y., Furumichi, M., and Tanabe, M. (2012). KEGG for integration and interpretation of large-scale molecular data sets. *Nucleic Acids Research*, **40**(Database issue), D109–D114.

Kaplan, U., Türkay, M., Biegler, L. T., and Karasözen, B. (2009). Modeling and simulation of metabolic networks for estimation of biomass accumulation parameters. *Discrete Applied Mathematics*, **157**(10), 2483–2493.

Kast, H. R., Goodwin, B., Tarr, P. T., Jones, S. A., Anisfeld, A. M., Stoltz, C. M., Tontonoz, P., Kliewer, S., Willson, T. M., and Edwards, P. A. (2002). Regulation of multidrug resistance-associated protein 2 (ABCC2) by the nuclear receptors pregnane X receptor, farnesoid X-activated receptor, and constitutive androstane receptor. *The Journal of Biological Chemistry*, **277**(4), 2908–2915.

Keating, S. M., Bergmann, F., Smith, L., Hucka, M., and Begley, K. (2013). SBML Test Suite. `http://sbml.org/Software/SBML_Test_Suite`.

Keller, R., Dörr, A., Tabira, A., Funahashi, A., Ziller, M. J., Adams, R., Rodriguez, N., Le Novère, N., Hiroi, N., Planatscher, H., Zell, A., and Dräger, A. (2013). The systems biology simulation core algorithm. *BMC Systems Biology*, **7**(1), 55.

Keller, R., Klein, M., Thomas, M., Dräger, A., Metzger, U., Templin, M., Joos, T., Thasler, W. E., Zell, A., and Zanger, U. M. (2016). Coordinating role of RXRα in downregulating hepatic detoxification during inflammation suggested by fuzzy-logic modeling. *PLoS Comput Biol*, **12**(1), e1004431.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Perth, Australia. IEEE.

Kirk, P. D. W. and Stumpf, M. P. H. (2009). Gaussian process regression bootstrapping: exploring the effects of uncertainty in time course data. *Bioinformatics*, **25**(10), 1300–1306.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, **220**(4598), 671–680.

Kitano, H. (2002). Systems biology: A brief overview. *Science*, **295**(5560), 1662–1664.

Klamt, S., Rodriguez, J. S., Lindquist, J., Simeoni, L., and Gilles, E. (2006). A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*, **7**(1), 56+.

Klein, M., Thomas, M., Hofmann, U., Seehofer, D., Damm, G., and Zanger, U. M. (2014). A systematic comparison of the impact of inflammatory signaling on ADME gene expression and activity in primary human hepatocytes and HepaRG cells. *Drug Metabolism and Disposition: The Biological Fate of Chemicals*.

Kolpakov, F. A., Tolstykh, N. I., Valeev, T. F., Kiselev, I. N., Kutumova, E. O., Ryabova, A., Yevshin, I. S., and Kel, A. E. (2011). BioUML–open source plug-in based platform for bioinformatics: invitation to collaboration. In *Moscow Conference on Computational Molecular Biology*, pages 172–173. Department of Bioengineering and Bioinformatics of MV Lomonosov Moscow State University.

Kotcon, B., Mesuro, S., Rozenfeld, D., and Yodpinyanee, A. (2011). *Final Report for Community of Ordinary Differential Equations Educators*. Harvey Mudd College Joint Computer Science and Mathematics Clinic, 301 Platt Boulevard, Claremont, CA 91711.

Kreutz, C., Raue, A., Kaschek, D., and Timmer, J. (2013). Profile likelihood in systems biology. *The FEBS Journal*, **280**(11), 2564–2571.

Kronfeld, M., Dräger, A., Aschoff, M., and Zell, A. (2009). On the Benefits of Multimodal Optimization for Metablic Network Modeling. In *German Conference on Bioinformatics*, pages 191–200.

Kronfeld, M., Planatscher, H., and Zell, A. (2010). The EvA2 Optimization Framework. In C. Blum and R. Battiti, editors, *Learning and Intelligent Optimization Conference, Special Session on Software for Optimization (LION-SWOP)*, number 6073 in Lecture Notes in Computer Science, LNCS, pages 247–250, Venice, Italy. Springer Verlag.

Krumsiek, J., Pölsterl, S., Wittmann, D. M., and Theis, F. J. (2010). Odefy–from discrete to continuous models. *BMC Bioinformatics*, **11**, 233.

Kühn, C., Wierling, C., Kühn, A., Klipp, E., Panopoulou, G., Lehrach, H., and Poustka, A. J. (2009). Monte Carlo analysis of an ODE Model of the Sea Urchin Endomesoderm Network. *BMC Systems Biology*, **3**(3), 83.

Le Novère, N., Bornstein, B. J., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J. L., and Hucka, M. (2006). BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, **34**, D689–D691.

Le Vee, M., Jouan, E., Stieger, B., and Fardel, O. (2013). Differential regulation of drug transporter expression by all-trans retinoic acid in hepatoma HepaRG cells and human hepatocytes. *European Journal of Pharmaceutical Sciences: Official Journal of the European Federation for Pharmaceutical Sciences*, **48**(4-5), 767–774.

LeCluyse, E. L. and Alexandre, E. (2010). Isolation and culture of primary hepatocytes from resected human liver tissue. *Methods in Molecular Biology*, **640**, 57–82.

Lee, H. Y., Suh, Y. A., Robinson, M. J., Clifford, J. L., Hong, W. K., Woodgett, J. R., Cobb, M. H., Mangelsdorf, D. J., and Kurie, J. M. (2000). Stress pathway activation induces phosphorylation of retinoid X receptor. *The Journal of Biological Chemistry*, **275**(41), 32193–32199.

Lefebvre, P., Benomar, Y., and Staels, B. (2010). Retinoid X receptors: common heterodimerization partners with distinct functions. *Trends in Endocrinology & Metabolism*, **21**(11), 676–683.

Li, C., Donizelli, M., Rodriguez, N., Dharuri, H., Endler, L., Chelliah, V., Li, L., He, E., Henry, A., Stefan, M. I., Snoep, J. L., Hucka, M., Le Novère, N., and Laibe, C. (2010). BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology*, **4**, 92.

Li, D., Zimmerman, T. L., Thevananther, S., Lee, H.-Y., Kurie, J. M., and Karpen, S. J. (2002). Interleukin-1 beta-mediated suppression of RXR:RAR transactivation of the Ntcp promoter is JNK-dependent. *The Journal of Biological Chemistry*, **277**(35), 31416–31422.

Liebermeister, W. and Klipp, E. (2006). Bringing metabolic networks to life: convenience rate law and thermodynamic constraints. *Theoretical Biology and Medical Modelling*, **3**(42), 41.

Liebermeister, W., Uhlendorf, J., and Klipp, E. (2010). Modular rate laws for enzymatic reactions: thermodynamics, elasticities and implementation. *Bioinformatics*, **26**(12), 1528–1534.

Livak, K. J. and Schmittgen, T. D. (2001). Analysis of relative gene expression data using real-time quantitative PCR and the 2(-Delta Delta C(T)) Method. *Methods (San Diego, Calif.)*, **25**(4), 402–408.

Lloyd, C. M., Halstead, M. D. B., and Nielsen, P. F. (2004). CellML: its future, present and past. *Progress in Biophysics & Molecular Biology*, **85**(2-3), 433–450.

Lowry, R. (2015). Concepts and Applications of Inferential Statistics. `http://faculty.vassar.edu/lowry/webtext.html`.

Madsen, C., Myers, C. J., Patterson, T., Roehner, N., Stevens, J. T., and Winstead, C. (2012). Design and test of genetic circuits using iBioSim. *Design Test of Computers, IEEE*, **29**(3), 32–39.

Mahadevan, R., Edwards, J. S., and Doyle, F. J. (2002). Dynamic flux balance analysis of diauxic growth in Escherichia coli. *Biophysical Journal*, **83**(3), 1331–1340.

Mahmood, T. and Yang, P.-C. C. (2012). Western blot: technique, theory, and trouble shooting. *North American Journal of Medical Sciences*, **4**(9), 429–434.

Maiwald, T. and Timmer, J. (2008). Dynamical modeling and multi-experiment fitting with PottersWheel. *Bioinformatics*, **24**(18), 2037–2043.

Matys, V., Kel-Margoulis, O. V., Fricke, E., Liebich, I., Land, S., Barre-Dirrie, A., Reuter, I., Chekmenev, D., Krull, M., Hornischer, K., Voss, N., Stegmaier, P., Lewicki-Potapov, B., Saxel, H., Kel, A. E., and Wingender, E. (2006). TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Research*, **34**(Database issue), D108–110.

Melas, I. N., Mitsos, A., Messinis, D. E., Weiss, T. S., and Alexopoulos, L. G. (2011). Combined logical and data-driven models for linking signalling pathways to cellular response. *BMC Systems Biology*, **5**(1), 107.

Migita, K., Miyashita, T., Maeda, Y., Nakamura, M., Yatsuhashi, H., Ishibashi, H., and Eguchi, K. (2005). An active metabolite of leflunomide, A77 1726, inhibits the production of serum amyloid A protein in human hepatocytes. *Rheumatology*, **44**(4), 443–448.

Moraru, I. I., Schaff, J. C., Slepchenko, B. M., Blinov, M. L., Morgan, F., Lakshminarayana, A., Gao, F., Li, Y., and Loew, L. M. (2008). Virtual Cell modelling and simulation software environment. *IET Systems Biology*, **2**(5), 352–362.

Morgan, E. T. (2009). Impact of infectious and inflammatory disease on cytochrome P450-mediated drug metabolism and pharmacokinetics. *Clinical Pharmacology and Therapeutics*, **85**(4), 434–438.

Morgan, E. T., Goralski, K. B., Piquette-Miller, M., Renton, K. W., Robertson, G. R., Chaluvadi, M. R., Charles, K. A., Clarke, S. J., Kacevska, M., Liddle, C., Richardson, T. A., Sharma, R., and Sinal, C. J. (2008). Regulation of Drug-Metabolizing Enzymes and Transporters in Infection, Inflammation, and Cancer. *Drug Metabolism and Disposition*, **36**(2), 205–216.

Morris, M. K., Saez-Rodriguez, J., Sorger, P. K., and Lauffenburger, D. A. (2010). Logic-Based Models for the Analysis of Cell Signaling Networks. *Biochemistry*, **49**(15), 3216–3224.

Morris, M. K., Saez-Rodriguez, J., Clarke, D. C., Sorger, P. K., and Lauffenburger, D. A. (2011). Training signaling pathway maps to biochemical data with constrained fuzzy logic: quantitative analysis of liver cell responses to inflammatory stimuli. *PLoS Computational Biology*, **7**(3), e1001099.

Myers, C. J., Barker, N., Jones, K., Kuwahara, H., Madsen, C., and Nguyen, N.-P. D. (2009). iBioSim: a tool for the analysis and design of genetic circuits. *Bioinformatics*, **25**(21), 2848–2849.

Nickerson, D. P., Corrias, A., and Buist, M. L. (2008). Reference descriptions of cellular electrophysiology models. *Bioinformatics*, **24**(8), 1112–1114.

Orth, J. D., Thiele, I., and Palsson, B. . (2010). What is flux balance analysis? *Nature Biotechnology*, **28**(3), 245–248.

Ozes, O. N., Mayo, L. D., Gustin, J. A., Pfeffer, S. R., Pfeffer, L. M., and Donner, D. B. (1999). NF-kappaB activation by tumour necrosis factor requires the Akt serine-threonine kinase. *Nature*, **401**(6748), 82–85.

Pascussi, J.-M., Gerbal-Chaloin, S., Duret, C., Daujat-Chavanieu, M., Vilarem, M.-J., and Maurel, P. (2008). The tangle of nuclear receptors that controls xenobiotic metabolism and transport: crosstalk and consequences. *Annual Review of Pharmacology and Toxicology*, **48**, 1–32. 00184 PMID: 17608617.

Perissi, V. and Rosenfeld, M. G. (2005). Controlling nuclear receptors: the circular logic of cofactor cycles. *Nature Reviews Molecular Cell Biology*, **6**(7), 542–554.

Petrovic, V., Teng, S., and Piquette-Miller, M. (2007). Regulation of drug transporters during infection and inflammation. *Molecular Interventions*, **7**(2), 99–111.

Poetz, O., Ostendorp, R., Brocks, B., Schwenk, J. M., Stoll, D., Joos, T. O., and Templin, M. F. (2005). Protein microarrays for antibody profiling: specificity and affinity determination on a chip. *Proteomics*, **5**(9), 2402–2411.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1993). *Numerical Recipes in FORTRAN; The Art of Scientific Computing*. Cambridge University Press, NY, USA.

Quaiser, T. and Mönnigmann, M. (2009). Systematic identifiability testing for unambiguous mechanistic modeling–application to JAK-STAT, MAP kinase, and NF-kappaB signaling pathway models. *BMC Systems Biology*, **3**, 50.

R Development Core Team (2011). R: A language and environment for statistical computing.

Raue, A., Kreutz, C., Maiwald, T., Bachmann, J., Schilling, M., Klingmüller, U., and Timmer, J. (2009). Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, **25**(15), 1923–1929.

Raue, A., Schilling, M., Bachmann, J., Matteson, A., Schelke, M., Kaschek, D., Hug, S., Kreutz, C., Harms, B. D., Theis, F. J., Klingmüller, U., and Timmer, J. (2013). Lessons learned from quantitative dynamical modeling in systems biology. *PLoS One*, **8**(9).

Raymond, G. M., Butterworth, E., and Bassingthwaighte, J. B. (2003). JSIM: Free software package for teaching physiological modeling and research. *The Journal of Experimental Biology*, **280**, 102–107.

Rechenberg, I. (1973). *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Number 15 in Problemata. Frommann-Holzboog, Stuttgart-Bad Cannstatt.

Renton, K. W. (2005). Regulation of drug metabolism and disposition during inflammation and infection. *Expert Opinion on Drug Metabolism & Toxicology*, **1**(4), 629–640.

Resasco, D. C., Gao, F., Morgan, F., Novak, I. L., Schaff, J. C., and Slepchenko, B. M. (2012). Virtual Cell: computational tools for modeling in cell biology. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, **4**(2), 129–140.

Rodriguez, N., Thomas, A., Watanabe, L. H., Vazirabad, I. Y., Kofia, V., Gómez, H. F., Mittag, F., Matthes, J., Rudolph, J., Wrzodek, F., Netz, E., Diamantikos, A., Eichner, J., Keller, R., Wrzodek, C., Fröhlich, S., Lewis, N. E., Myers, C. J., Novère, N. L., Palsson, B. ., Hucka, M., and Dräger, A. (2015). JSBML 1.0: providing a smorgasbord of options to encode systems biology models. *Bioinformatics*, **31**(20), 3383–3386.

Rodriguez-Fernandez, M., Egea, J. A., and Banga, J. R. (2006). Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. *BMC Bioinformatics*, **7**, 483+.

Romashkova, J. A. and Makarov, S. S. (1999). NF-kappaB is a target of AKT in anti-apoptotic PDGF signalling. *Nature*, **401**(6748), 86–90.

Rowan, T. (1990). *Functional Stability Analysis of Numerical Algorithms*. Ph.D. thesis, Department of Computer Sciences, University of Texas at Austin.

Runge-Morris, M. and Kocarek, T. A. (2009). Regulation of sulfotransferase and UDP-glucuronosyltransferase gene expression by the PPARs. *PPAR Research*, **2009**, 728941.

Ryll, A., Samaga, R., Schaper, F., Alexopoulos, L. G., and Klamt, S. (2011). Large-scale network models of IL-1 and IL-6 signalling and their hepatocellular specification. *Molecular BioSystems*, **7**, 3253–3270.

Saez-Rodriguez, J., Alexopoulos, L. G., Epperlein, J., Samaga, R., Lauffenburger, D. A., Klamt, S., and Sorger, P. K. (2009). Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Molecular Systems Biology*, **5**(1).

Saez-Rodriguez, J., Alexopoulos, L. G., Zhang, M., Morris, M. K., Lauffenburger, D. A., and Sorger, P. K. (2011). Comparing Signaling Networks between Normal and Transformed Hepatocytes Using Discrete Logical Models. *Cancer Research*, **71**(16), 5400–5411.

Samaga, R. and Klamt, S. (2013). Modeling approaches for qualitative and semi-quantitative analysis of cellular signaling networks. *Cell Communication & Signaling*, **11**(1), 1 – 19.

Samaga, R., Saez-Rodriguez, J., Alexopoulos, L. G., Sorger, P. K., and Klamt, S. (2009). The Logic of EGFR/ErbB Signaling: Theoretical Properties and Analysis of High-Throughput Data. *PLoS Comput Biol*, **5**(8), e1000438.

Santillán, M. (2008). On the Use of the Hill Functions in Mathematical Models of Gene Regulatory Networks. *Mathematical Modelling of Natural Phenomena*, **3**, 85–97.

Schilling, M., Maiwald, T., Hengl, S., Winter, D., Kreutz, C., Kolch, W., Lehmann, W. D., Timmer, J., and Klingmüller, U. (2009). Theoretical and experimental analysis links isoform- specific ERK signalling to cell fate decisions. *Molecular Systems Biology*, **5**(1).

Schmidt, H. and Jirstrand, M. (2006). Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology. *Bioinformatics*, **22**(4), 514–515.

Schust, J., Sperl, B., Hollis, A., Mayer, T. U., and Berg, T. (2006). Stattic: A Small-Molecule Inhibitor of STAT3 Activation and Dimerization. *Chemistry & Biology*, **13**(11), 1235–1242. 00320.

Schwefel, H.-P. (1975). *Evolutionsstrategie und numerische Optimierung*. Dr.-Ing. Thesis, Technical University of Berlin, Department of Process Engineering.

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, **13**(11), 2498–2504.

Slaviero, K. A., Clarke, S. J., and Rivory, L. P. (2003). Inflammatory response: an unrecognised source of variability in the pharmacokinetics and pharmacodynamics of cancer chemotherapy. *The Lancet Oncology*, **4**(4), 224–232. 00129.

Spurgeon, S. L., Jones, R. C., and Ramakrishnan, R. (2008). High Throughput Gene Expression Measurement with Real Time PCR in a Microfluidic Dynamic Array. *PLoS ONE*, **3**(2), e1662. 00189.

Storn, R. (1996). On the Usage of Differential Evolution for Function Optimization. In *1996 Biennial Conference of the North American Fuzzy Information Processing Society*, pages 519–523, Berkeley, CA, USA. IEEE, New York, USA.

Storn, R. and Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Glob Opt*, **11**(4), 341–359.

Sun, J., Garibaldi, J. M., and Hodgman, C. (2012). Parameter estimation using metaheuristics in systems biology: a comprehensive review. *IEEE/ACM Transactions on Computational Biology and Bioinformatics / IEEE, ACM*, **9**(1), 185–202.

Sun Kim, M., Sweeney, T. R., Shigenaga, J. K., Chui, L. G., Moser, A., Grunfeld, C., and Feingold, K. R. (2007). TNF and IL-1 decrease RXRα, PPARα, PPARγ, LXRα, and the coactivators SRC-1, PGC-1α, and PGC-1β in liver cells. *Metabolism: Clinical and Experimental*, **56**(2), 267–279.

Takizawa, H., Nakamura, K., Tabira, A., Chikahara, Y., Matsui, T., Hiroi, N., and Funahashi, A. (2013). LibSBMLSim: A reference implementation of fully functional SBML simulator. *Bioinformatics*, **29**(11), 1474–6.

Teng, S. and Piquette-Miller, M. (2005). The involvement of the pregnane X receptor in hepatic gene regulation during inflammation in mice. *The Journal of Pharmacology and Experimental Therapeutics*, **312**(2), 841–848.

Terfve, C., Cokelaer, T., Henriques, D., MacNamara, A., Goncalves, E., Morris, M. K., van Iersel, M., Lauffenburger, D. A., and Saez-Rodriguez, J. (2012). CellNOptR: a flexible toolkit to train protein signaling networks to data using multiple logic formalisms. *BMC Systems Biology*, **6**, 133.

Tham, L.-S., Wang, L., Soo, R. A., Lee, S.-C., Lee, H.-S., Yong, W.-P., Goh, B.-C., and Holford, N. H. G. (2008). A pharmacodynamic model for the time course of tumor shrinkage by gemcitabine + carboplatin in non-small cell lung cancer patients. *Clinical Cancer Research*, **14**(13), 4213–4218.

Thomas, M., Burk, O., Klumpp, B., Kandel, B. A., Damm, G., Weiss, T. S., Klein, K., Schwab, M., and Zanger, U. M. (2013). Direct transcriptional regulation of human hepatic cytochrome P450 3A4 (CYP3A4) by peroxisome proliferator-activated receptor alpha (PPARα). *Molecular Pharmacology*, **83**(3), 709–718.

Tibes, R., Qiu, Y., Lu, Y., Hennessy, B., Andreeff, M., Mills, G. B., and Kornblau, S. M. (2006). Reverse phase protein array: validation of a novel proteomic technology and utility for analysis of primary leukemia specimens and hematopoietic stem cells. *Molecular Cancer Therapeutics*, **5**(10), 2512–2521.

Tolson, A. H. and Wang, H. (2010). Regulation of drug-metabolizing enzymes by xenobiotic receptors: PXR and CAR. *Advanced Drug Delivery Reviews*, **62**(13), 1238–1249.

Tovey, C. A. (1985). Hill climbing with multiple local optima. *SIAM Journal on Algebraic and Discrete Methods*, **6**(3), 384–393.

Töpfer, N., Kleessen, S., and Nikoloski, Z. (2015). Integration of metabolomics data into metabolic networks. *Frontiers in Plant Science*, **6**, 49.

Vanlier, J., Tiemann, C. A., Hilbers, P. a. J., and van Riel, N. a. W. (2013). Parameter uncertainty in biochemical models described by ordinary differential equations. *Mathematical Biosciences*, **246**(2), 305–314.

Vee, M. L., Lecureur, V., Stieger, B., and Fardel, O. (2009). Regulation of drug transporter expression in human hepatocytes exposed to the proinflammatory cytokines tumor necrosis factor-α or interleukin-6. *Drug Metabolism and Disposition*, **37**(3), 685–693.

Wajima, T., Isbister, G. K., and Duffull, S. B. (2009). A comprehensive model for the humoral coagulation network in humans. *Clinical Pharmacology & Therapeutics*, **86**(3), 290–298.

Waltemath, D., Adams, R., Beard, D. A., Bergmann, F. T., Bhalla, U. S., Britten, R., Chelliah, V., Cooling, M. T., Cooper, J., Crampin, E. J., Garny, A., Hoops, S., Hucka, M., Hunter, P., Klipp, E., Laibe, C., Miller, A. K., Moraru, I., Nickerson, D., Nielsen, P., Nikolski, M., Sahle, S., Sauro, H. M., Schmidt, H., Snoep, J. L., Tolle, D., Wolkenhauer, O., and Le Novère, N. (2011a). Minimum Information About a Simulation Experiment (MIASE). *PLoS Computational Biology*, **7**(4), e1001122.

Waltemath, D., Adams, R., Bergmann, F. T., Hucka, M., Kolpakov, F., Miller, A. K., Moraru, I. I., Nickerson, D., Sahle, S., Snoep, J. L., and Le Novère, N. (2011b). Reproducible computational biology experiments with SED-ML–the Simulation Experiment Description Markup Language. *BMC Systems Biology*, **5**, 198.

Wang, Z., Salih, E., and Burke, P. A. (2011). Quantitative Analysis of Cytokine-Induced Hepatocyte Nuclear Factor-4α Phosphorylation by Mass Spectrometry. *Biochemistry*, **50**(23), 5292–5300.

Warnes, G. R., Bolker, B., Bonebakker, L., Gentleman, R., Liaw, W. H. A., Lumley, T., Maechler, M., Magnusson, A., Moeller, S., Schwartz, M., and Venables, B. (2013). *gplots: Various R programming tools for plotting data*.

Wittig, U., Kania, R., Golebiewski, M., Rey, M., Shi, L., Jong, L., Algaa, E., Weidemann, A., Sauer-Danzwith, H., Mir, S., Krebs, O., Bittkowski, M., Wetsch, E., Rojas, I., and Müller, W. (2012). SABIO-RK–database for biochemical reaction kinetics. *Nucleic Acids Res*, **40**(Database issue), D790–D796.

Wolf, J., Passarge, J., Somsen, O. J. G., Snoep, J. L., Heinrich, R., and Westerhoff, H. V. (2000). Transduction of intracellular and intercellular dynamics in yeast glycolytic oscillations. *Biophysical Journal*, **78**(3), 1145–1153.

Wrzodek, C., Dräger, A., and Zell, A. (2011). KEGGtranslator: visualizing and converting the KEGG PATHWAY database to various formats. *Bioinformatics*, **27**(16), 2314–2315.

Xie, W., editor (2009). *Nuclear receptors in drug metabolism*. John Wiley & Sons, Hoboken, NJ.

Zanger, U. M. and Schwab, M. (2013). Cytochrome P450 enzymes in drug metabolism: Regulation of gene expression, enzyme activities, and impact of genetic variation. *Pharmacology & Therapeutics*, **138**(1), 103–141.

Zanger, U. M., Turpeinen, M., Klein, K., and Schwab, M. (2008). Functional pharmacogenetics/genomics of human cytochromes P450 involved in drug biotransformation. *Analytical and Bioanalytical Chemistry*, **392**(6), 1093–1108.

Zhao, J., Liu, J., Pang, X., Wang, S., Wu, D., Zhang, X., and Feng, L. (2013). Angiotensin II induces c-reactive protein expression via AT1-ROS-MAPK-NF-κb signal pathway in hepatocytes. *Cellular Physiology and Biochemistry: International Journal of Experimental Cellular Physiology, Biochemistry, and Pharmacology*, **32**(3), 569–580.

Zordoky, B. N. M. and El-Kadi, A. O. S. (2009). Role of NF-kappaB in the regulation of cytochrome P450 enzymes. *Current Drug Metabolism*, **10**(2), 164–178.