

Interactive Visualization of Complex Structures in Modular Models for Systems Biology

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

vorgelegt der



Naturwissenschaftlichen Fakultät III der
Martin-Luther-Universität Halle-Wittenberg

von Herrn

Diplom-Ingenieur Sebastian Mirschel
geboren am 13.06.1977 in Görlitz

Gutachter:

Prof. Dr. Falk Schreiber

Prof. Dr.-Ing. Andreas Kerren

Verteidigung am: 02. Juli 2012

Magdeburg, im Juli 2012

Forschungsberichte aus dem Max-Planck-Institut
für Dynamik komplexer technischer Systeme

Band 35

Sebastian Mirschel

**Interactive Visualization of Complex Structures
in Modular Models for Systems Biology**

Shaker Verlag
Aachen 2012

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <http://dnb.d-nb.de>.

Zugl.: Halle, Univ., Diss., 2012

Copyright Shaker Verlag 2012

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publishers.

Printed in Germany.

ISBN 978-3-8440-1275-0

ISSN 1439-4804

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen

Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9

Internet: www.shaker.de • e-mail: info@shaker.de

Acknowledgments

This thesis is the summary of recent work at the Max Planck Institute for Dynamics of Complex Technical Systems. In the System Biology Group, I encountered an excellent working atmosphere, and got the chance to work at the interface between mathematics, engineering, biology, and computer science.

I would like to thank the many individuals who have helped me on the path towards this dissertation. I am deeply grateful to my advisors, Prof. Ernst Dieter Gilles and Prof. Falk Schreiber.

Prof. Ernst Dieter Gilles has supported me throughout my thesis with his wisdom, inspiration and motivation. He has always given me the freedom to work in my own way. Prof. Falk Schreiber has provided me with thoughtful comments and encouragement. I would like to thank him for his uncomplicated, open and helpful way of thinking. Without their patient guidance and generous support, this dissertation would not have been possible. I am also in debt to Prof. Andreas Kerren, who accepted to review my thesis.

I sincerely appreciate the friendship and inspiration of the people I have had the good fortune of working with during this thesis.

I gratefully acknowledge my co-workers from the PROMOT Development Team: Katrin Kolczyk, Michael Rempel, Gerrit Danker and Martin Ginkel. I would also like to thank Erik Aschenbrenner for supporting me with the realization of ideas and concepts of the HyCoGLa approach.

I would like to thank my colleagues Regina Samaga, Jeremy Húard and Julio Saez-Rodriguez for the discussions we had and their ideas and comments with respect to modeling issues and visualization methods. They also provided me with complex, challenging models and gave me permission to use their data and images in this thesis.

I would like to thank those with whom I shared my office, Odón Angeles Palacios, Axel von Kamp and Oliver Hädicke for their friendship, help, and patience.

I would like to express my gratitude to all my colleagues from the Systems Biology Group and the Max Planck Institute, who provided invaluable support and infrastructure along the way. Especially, I would like to thank Steffen Klamt, Renate Wagner, Janine Holzmann, Robert Rehner, Gerry Truschkewitz and the library team.

My deepest thanks go to my parents and especially to my family, Thaddeus, Alina and Christine for their love and support throughout. They have kept me grounded and provided me with the necessary ambience to be relaxed.

The work presented in this thesis was part of the Systems Biology Competence Network ‘HepatoSys’ and the research initiative ‘FORSYS’. Both were financially supported by the German Federal Ministry for Education and Research (BMBF).

List of Figures	ix
List of Tables	xiii
List of Algorithms	xv
List of Symbols and Abbreviations	xvii
1. Introduction	1
1.1. Motivation	1
1.2. Research Challenges and Objectives	2
1.3. Organization	3
1.4. VISUAL EXPLORER	5
2. Modeling in Systems Biology – Fundamentals	7
2.1. Biological Background	7
2.1.1. Intra-Cellular Processes	7
2.1.2. Cellular Signaling	8
2.2. The Modeling Process	8
2.2.1. The Process of Knowledge Generation	9
2.2.2. Types of Models	12
2.3. Modeling Approaches	13
2.3.1. Logical Modeling	13
2.3.2. Dynamic Modeling	14
2.3.3. Modular Modeling	15
2.4. Modeling Tools and PROMOT	18
2.4.1. Modeling Languages, Tools and Formats	18
2.4.2. Modeling with PROMOT	19
2.5. Application to Models	22
2.5.1. The EGF Receptor Models	22
2.5.2. The T Cell Receptor Model	24
2.6. Conclusions	24

3. Graphs and Visualization in Systems Biology – Fundamentals	27
3.1. Graph Theory	27
3.1.1. Graph Classes	28
3.1.2. Characteristics of Graphs	29
3.2. Graph Drawing	30
3.2.1. Graph Layout	30
3.2.2. Layout Strategies	33
3.2.3. Methods on Physical Analogies	34
3.2.4. Domain-Specific Knowledge	35
3.3. Visualization Principles	36
3.3.1. Information Visualization	36
3.3.2. The Visualization Pipeline	36
3.3.3. Perceptual and Cognitive Principles	38
3.3.4. Interactive Exploration	40
3.4. Visual Interface Schemes	42
3.4.1. Overview+Detail Interfaces	42
3.4.2. Zooming Interfaces	43
3.4.3. Focus+Context Interfaces	43
3.5. Networks and Modular Models	45
3.5.1. Cellular Networks and their Visual Representation	45
3.5.2. Visualizing Modular Models	46
3.6. Tools for Visualization	48
3.6.1. ZUI Toolkits	49
3.6.2. Visualization Tools for Systems Biology	49
3.6.3. Visualization with PROMOT	52
3.7. Conclusions	52
4. Visual Representation of Dynamic and Logical Modular Models	55
4.1. Research Problem	55
4.2. Towards a Single Integrated View	56
4.3. Entities for Dynamic Modeling and their Visualization	58
4.3.1. Basic Elements	58
4.3.2. Visualizing a Composite Module	59
4.3.3. 'Phosphorylation'-Example	60
4.4. Entities for Logical Modeling and their Visualization	61
4.4.1. Basic Elements	62
4.4.2. Visualizing a Composite Module	66
4.4.3. Visual Handling of Parameters	68
4.4.4. The Logical Toy Model	69
4.5. Applications	69
4.5.1. The Dynamic EGF Model	69
4.5.2. The Logical Models: EGFR/ErbB and TCR	71
4.5.3. Integration with CELLNETANALYZER – Multiple Maps	71
4.6. Conclusions	75
5. A Visual Interface for Interactive Exploratory Model Analysis	77
5.1. Research Problem	77
5.2. Interactive Exploration Schemes for Modular Models	78

5.2.1.	InteractiveZoom	79
5.2.2.	ContextualZoom	79
5.2.3.	HierarchicalZoom	82
5.3.	Animations, Semantic Zooming and LOD	84
5.3.1.	Smooth (Non-)Uniform Animations	85
5.3.2.	Semantic Zooming and LOD	85
5.3.3.	Efficient Pan+Zoom Operations	89
5.4.	Combining and Comparing Interactive Exploration Schemes	89
5.4.1.	Overview+Detail	90
5.4.2.	Comparison	90
5.5.	Applications	93
5.5.1.	Search the Modular Model	93
5.5.2.	Visual Correlation Using Multiple Foci and Contextual Visualization	95
5.5.3.	Topology-Aware Exploring of the Local Context	95
5.6.	Conclusions	97
6.	Visual Scenarios	99
6.1.	Research Problem	99
6.2.	Concept	101
6.2.1.	Visual Mappings Using Transfer Functions	101
6.2.2.	Filter and Analysis	102
6.2.3.	Layout and Exploration Strategy	103
6.2.4.	Visual Mapping in Modular Models	105
6.2.5.	Pre- and User-Defined Visual Scenarios	105
6.3.	Specific Visual Scenarios	107
6.3.1.	Visual Scenarios ‘Dynamic Model’ and ‘Logical Model’	107
6.3.2.	Visual Scenario ‘Hierarchical Level’	107
6.3.3.	Visual Scenario ‘Connectivity’	109
6.3.4.	Visual Scenario ‘Interaction Distance’	109
6.3.5.	Defining Local Visual Scenarios	113
6.4.	Applications	113
6.4.1.	Overlay Data in Context of Networks	115
6.4.2.	Visual Analysis of the Logical TCR Model	117
6.4.3.	Visual Analysis of the Logical EGFR/ErbbB Model	118
6.4.4.	Visualization of Time-Series Data	118
6.5.	Conclusions	119
7.	Layout Strategies for Modular Models	125
7.1.	Research Problem	126
7.2.	HyCoGLa	128
7.2.1.	The HyCoGLa Algorithm	128
7.2.2.	Domain-Specific Knowledge	131
7.2.3.	Visual Handling of Multiple Instances	133
7.2.4.	Generalization of the Concept	133
7.2.5.	Runtime and Layout Quality	133
7.3.	Flat Layout	135
7.3.1.	The Flattening Algorithm	136
7.3.2.	Applications	137

7.4. Conclusions	138
8. Concluding Remarks	141
8.1. Scientific Contributions	141
8.2. Open Issues and Research Directions	143
Bibliography	145
Index	161
A. CD Information, Software Architecture and UML Diagrams	163
A.1. Supplied CD	163
A.2. Architecture of PROMoT	163
A.3. UML Diagrams	164
B. Visual Representations and Visual Scenarios	169
B.1. Symbol Reference Cards	169
B.2. Comparison of Elements	169
B.3. Visual Handling of Additional Parameters	169
B.4. Visual Representation of the Logical Toy Model	173
B.5. Logical Model of EGFR/ErbB Signaling	175
B.6. Logical Model of TCR Signaling	175
B.7. Visual Properties and Mappings	175
B.8. Visual Scenarios and Model Types	175
C. Interactive Exploration Schemes	181
C.1. InteractiveZoom versus ContextualZoom	181
C.2. Zoom Sequence with LOD	181
C.3. Shortcuts and Mouse Controls	181
D. Layout and Algorithms	185
D.1. Applying FlatLayout	185
D.2. Algorithms for FlatLayout	185
D.3. Algorithms for HyCoGLa	185

1.1. Relations of individual chapters	4
2.1. Overview of signal transduction pathways	9
2.2. Knowledge generation as an iterative process	11
2.3. Different model types	14
2.4. Structural and hierarchical organization	15
2.5. Hierarchical and modular modeling with PROMOT	20
2.6. Schematic view of the modeling workflow in PROMOT	21
2.7. Network of the EGFR/ErbB signaling	23
2.8. Network of the TCR signaling	24
3.1. Graph classes	28
3.2. Different views of hierarchy	31
3.3. Typical visual properties of nodes and edges	32
3.4. Aesthetic criteria and conventions	33
3.5. Standard layout methods	34
3.6. Visualization pipeline	37
3.7. Preattentive features	39
3.8. Relative effectiveness of visual cues	39
3.9. Mental maps in graph drawing	40
3.10. Overview of animations functions	42
3.11. Space-scale diagrams	44
3.12. Semantic zooming in space-scale diagrams	44
3.13. Views of modular models	48
3.14. Visualization of modular EGF model using disjunct views	52
4.1. Comparison of visualization of a modular model	57
4.2. Visualization of storages and reactions	60
4.3. Visualization of links and terminals	60
4.4. Visualizing a composite module (dynamic)	61
4.5. Dynamic model using typical components	62
4.6. Visualization options for a reservoir element	65

List of Figures

4.7. Visualization and usage of a dummy element	65
4.8. Visualization of logical gates	66
4.9. Visualization of different interaction types in logical models	66
4.10. Visualizing a composite module (logical)	67
4.11. VARIABLE EDITOR for a somehow element	68
4.12. Logical Toy model in PROMOT	70
4.13. Visual representation of the dynamic EGF model.	72
4.14. Visual representation of the logical EGFR/ErbB model	73
4.15. Visual representation of the logical TCR model	74
4.16. Multiple maps for CELLNETANALYZER	76
5.1. Focus+Context using the ContextualZoom	81
5.2. ContextualZoom with consideration of edge bend points	81
5.3. Focus+Context with multiple foci	83
5.4. Different scale factors	83
5.5. Zooming based on topological features	84
5.6. HierarchicalZoom with sensitive areas	85
5.7. Semantic zooming	86
5.8. LOD applied to the modular EGF model	87
5.9. Visual states of an enzymatic reaction	88
5.10. Different strategies for Pan+Zoom operations	89
5.11. Logical Toy model with Overview+Detail	91
5.12. Overview+Detail using two cameras	91
5.13. HierarchicalZoom versus ContextualZoom	92
5.14. Comparison of views of different exploration schemes	94
5.15. Exploration schemes applied to space-scale diagrams	94
5.16. Search the modular EGF model	95
5.17. ContextualZoom	96
5.18. Exploring the local context	97
6.1. Different visual transfer functions	103
6.2. Different components for network visualization	104
6.3. Constant width of lines and borders	106
6.4. Handling of of line and border width in modular models	106
6.5. Color coding of hierarchical levels	108
6.6. Computation of the interaction distance	112
6.7. Interaction distance applied to the TCR model	112
6.8. Options for handling hierarchical relations	114
6.9. Options for directionality of interactions	114
6.10. Options for handling multiple foci	116
6.11. VSs encoding different distance metrics	116
6.12. Different VSs applied to the logical TCR model	117
6.13. Different visual scenarios of the TCR model	121
6.14. Different visual scenarios of the EGFR/ErbB model	122
6.15. Visualization of time-series data	123
7.1. Different layouts of the dynamic EGF model	126
7.2. Traversal of the inclusion tree of the Toy model	129

7.3. Model of forces	130
7.4. Comparison of different layouts	132
7.5. HyCoGLa with domain-specific constraints	134
7.6. Handling of multiple instances	134
7.7. Traversing of inclusion tree with different parameter values	135
7.8. Results for runtime and quality of the layout algorithm	136
7.9. Example graph with applied local method and HyCoGLa	137
7.10. Important steps of FlatLayout algorithm	138
7.11. FlatLayout applied to the Toy model	139
7.12. Manual, modular layout versus FlatLayout	139
A.1. Diagram of the software architecture of PROMoT	165
A.2. UML class diagram compiling important classes	166
A.3. UML diagram compiling the package structure	167
B.1. Symbol reference card for dynamic models	170
B.2. Symbol reference card for logical models	171
B.3. Visual representation of a simplified Toy model	174
B.4. Alternative visualizations of the Toy model using local VSs	174
B.5. Visual representations in the modeling workflow of the logical EGFR/ErbB model . . .	176
B.6. Different visual scenarios of logical TCR model	177
B.7. Additional visual scenarios of logical TCR model	178
C.1. InteractiveZoom versus ContextualZoom	182
C.2. LOD applied to the Ecoli model	183
D.1. Comparison of manual, modular layout and FlatLayout	186

List of Figures

3.1. Comparison of tools and their visualization features	51
5.1. Features of interactive exploration schemes in VISUAL EXPLORER	93
B.1. Graphical representations of logical elements	172
B.2. Visual mappings for nodes and edges in VISUAL EXPLORER	179
B.3. Global visual scenarios and their components	180
B.4. Global visual scenarios and their valid model types	180
B.5. Model types and invalid network elements	180
C.1. Shortcuts and modifiers used in VISUAL EXPLORER.	184

List of Tables

List of Algorithms

1.	General FlatLayout algorithm	187
2.	Traverse hierarchy	187
3.	General HyCoGLa algorithm	187
4.	Extended spring layout algorithm	188
5.	Calculate spring forces	188
6.	Calculate repulsive forces	189
7.	Calculate gravitational forces	189
8.	Calculate magnetic forces	189

List of Symbols and Abbreviations

cf.	confer, compare
e.g.	exempli gratia
etc.	et cetera
i.e.	id est
vs.	versus
2D	<u>2</u> - <u>D</u> imensional
3D	<u>3</u> - <u>D</u> imensional
CNA	<u>C</u> ell <u>N</u> et <u>A</u> nalyzer
DAE	<u>D</u> ifferential <u>A</u> lgebraic <u>E</u> quation
DIVA	<u>D</u> ynamische <u>S</u> imulation <u>v</u> erfahrenstechnischer <u>A</u> nlagen
DNA	<u>D</u> eoxyribo <u>n</u> ucleic <u>A</u> cid
Diana	<u>D</u> ynamic <u>S</u> imulation <u>a</u> nd <u>N</u> umerical <u>A</u> nalysis Tool
E. coli	Escherischa Coli
EGF	<u>E</u> pidermal <u>G</u> rowth <u>F</u> actor
EGFR/ErbB	<u>E</u> pidermal Growth <u>F</u> actor <u>R</u> eceptor / Receptor Family ErbB
FEV	<u>F</u> ish <u>E</u> ye <u>V</u> iew
GUI	<u>G</u> raphical <u>U</u> ser <u>I</u> nterface
GraphML	Description format for graphs based on XML
HyCoGLa	<u>H</u> ybrid <u>C</u> ompound <u>G</u> raph <u>L</u> ayout
In-silico	simulated experiments
In-vitro	experiments under artificial conditions in a test tube
In-vivo	experiments with living cell/system
LOD	<u>L</u> evel of <u>D</u> etail
MAPK	<u>M</u> itogen- <u>A</u> ctivated Protein <u>K</u> inase
MDL	<u>M</u> odeling <u>D</u> efinition <u>L</u> anguage
MMC	<u>M</u> odular <u>M</u> odeling <u>C</u> oncept
NT	<u>N</u> etwork <u>T</u> heory
ODE	<u>O</u> rdinary <u>D</u> ifferential <u>E</u> quation
OVL	<u>O</u> mix <u>V</u> isualization <u>L</u> anguage
Omics	technological methods for gathering massive datasets from biological relations (e. g., metabolomics, proteomics, genomics)

List of Algorithms

PDE	<u>P</u> artial <u>D</u> ifferential <u>E</u> quation
ProMoT	<u>P</u> rocess <u>M</u> odeling <u>T</u> ool
SBGN	<u>S</u> ystems <u>B</u> iology <u>G</u> raphical <u>N</u> otation
SBML	<u>S</u> ystems <u>B</u> iology <u>M</u> arkup <u>L</u> anguage
TCR	<u>T</u> umor <u>C</u> ell <u>R</u> eceptor
UML	<u>U</u> nified <u>M</u> odeling <u>L</u> anguage
VS	<u>V</u> isual <u>S</u> cenario
XML	<u>E</u> xtensible <u>M</u> arkup <u>L</u> anguage
ZUI	<u>Z</u> oomable <u>U</u> ser <u>I</u> nterface

We now have unprecedented ability to collect data about nature but there is now a crisis developing in biology, in that completely unstructured information does not enhance understanding. We need a framework to put all of this knowledge and data into – that is going to be the problem in biology.

We've reached the stage where we can't talk to each other – we've all become highly specialized. We need a framework, a framework where people can come back to us and say, 'Yes, I understand.' Driving toward that framework is really the big challenge.

Sydney Brenner, Nobel Prize, 2002 [27]

CHAPTER 1

Introduction

1.1. Motivation

In recent years, the modeling of biochemical reaction systems has tended to become more complex. They are omnipresent and can be found in various domains such as process engineering in the form of fuel cells or membrane reactors, or in systems biology as complicated intra-cellular processes. Especially systems biology is being pushed along by some trends, which are highlighted in the following paragraph.

Researchers in the molecular life sciences are faced with exploding quantities of data. By using new parallel techniques many projects are generating an escalating amount of data¹, including whole genomes, the human interactome or huge signaling networks. Research in systems biology is characterized by multidisciplinary teams. Organized into huge consortia, groups with different scientific backgrounds, often highly specialized, work closely together on a common objective. Often, these collaborations result in data with heterogeneous characteristics or data distributed across many different locations worldwide. Collaboration between different disciplines needs a common and intuitive basis for understanding and discussing complex structures of biochemical systems. In order to obtain a holistic understanding of the biological processes, heterogeneous and distributed data need to be integrated. Recently, diverse databases such as KEGG [93], BioModels [117] or JWS ONLINE [142] and model description formats such as CELLML [67], BioPAX [44] or SBML [86] have provided ways to consolidate the accumulated data. Indeed researchers are faced with models growing in size, complexity and scope. Accordingly, the amount of work needed to evaluate and integrate the data is increasing rapidly.

In this sense, *visualization* can be understood as a domain-independent, universal tool and a common basis for communication and transfer of knowledge. Visual representations can assist the development of mental models that allow researchers to access, integrate and analyze complex biochemical information, and support the process of decision making. Most of the recent software systems dedicated to systems biology use visual output for the representation of their data and results. However, traditional approaches have several limitations due to the increasing size and complexity of biochemical models and visualization trade-offs including the lack of screen real estate, namely, limited screen size and resolution. Innovative visualization techniques are rarely applied. As a consequence, in the area of systems biology new methods and tools for efficient, interactive visualization need to be developed [69].

¹It is estimated that the amount of biological data doubles every six months – one of the fastest rates in a scientific discipline.

1. Introduction

For instance, splitting the model up into different units by applying hierarchical and module-based decomposition (results in modular models), consequently visualizing them (e. g., as subnetworks or using different network perspectives), and providing ways to explore them can help to understand complex model structures and make them more manageable in terms of computational purposes, formal representation, and explaining the knowledge to other researchers.

This thesis exploits existing and contributes novel visualization and exploration techniques that can be used to tackle current challenges in systems biology.

1.2. Research Challenges and Objectives

Researchers in systems biology are faced with several research challenges in the visualization and exploration of complex models. This thesis focuses on the particular research questions and challenges listed below. Most have also been highlighted in recently published reviews [4, 57, 69, 77, 145, 170, 188].

1. Visual representation of dynamic and logical modular models

Model decomposition based on modules and hierarchy is a powerful concept. However, such complex model structures needs to be visually represented in order to improve their interpretation. This is relevant both for dynamic and logical models. Dynamic models are characterized by a detailed mathematical description of the underlying processes. This detailed knowledge should be visualized in a systematic manner. Logical models, typically, are characterized by many components and complex relationships. Thus, visual support is greatly needed to represent modular logical networks and to facilitate their convenient visual exploration. In this regard, a central point is the formalized visual representations of modular structures specifically adapted to dynamic and logical models.

2. Exploration of and navigation in complex structures of modular models

For convenient use, not only a graphical representation of the network is crucial; a graphical network needs to be combined with exploration techniques that are able to handle navigation in complex structures in an intuitive way. Since many current approaches are based on models without hierarchical structures, less research has been done in linking techniques to content and its structure. Specific research questions in this context are: How can modular models be interactively explored? What are efficient navigation and interaction patterns? How can submodels be explored in the context of the entire system? Providing an integrated visualization of a modular model combined with adequate exploration and navigation schemes would help users manage large and complex networks.

3. Flexible graphical representations

A standardized representation is essential in order to provide a common and unambiguous notation for exchanging knowledge [114, 170]. However, a standard specification is not necessarily the appropriate way to graphically depict knowledge. In many cases, visual representations have to be tailored to a specific modeling task or individual requirements and preferences. Since the requirements can frequently change (i. e., the current modeling task), it should be possible to modify and switch complex visual settings in an easy and flexible manner. The visual appearance of the network with complex structures is directly affected not only by the visual properties of the network elements, but also by the exploration scheme and the overall layout. Hence, a strategy is desired that will define individual views that take into account major aspects of visual representation while still facilitating high-performance switching.

4. Overlaying data on top of modular models

A huge amount of datasets is derived from external in-vivo, in-vitro and in-silico experiments. Different ways exist to integrate the collected data using visual methods. A common approach is overlaying this external data on top of the network model using data glyphs, subgraphics or animated transitions. When working with modular models, the integration and subsequent interpretation of overlay data is challenging due to the complex structures. Hence, there is a strong need for adequate visualization techniques.

5. Layout of modular models

The drawing of complex model structures can be partly covered by graph layout strategies. Layout algorithms are able to reveal global patterns within the context of the network model such as structures of protein complexes and relationships between them. However, general purpose layout algorithms do not consider the specific structure of modular models. Furthermore, several domain-specific requirements need to be incorporated into the biologically motivated layout process. A relevant example is the direction of reactions in logical models of signaling networks where all reactions point from top to bottom, thus creating an overall causal signaling flow in the same direction. An adapted layout incorporating different domain-specific constraints can result in an improved representation of modular structures and consequently enhance the understanding of the biological meaning.

In summary, the overall research objective for the work described in this thesis is to support modelers in visualization and exploratory analysis of modular models by supplying them with appropriate tools. Thereby, the focus of the cited and described literature as well as the research results is mainly on the application for systems biology and in particular signal transduction systems.

1.3. Organization

The remainder of this thesis is organized into eight chapters. The first part contains two chapters, which cover the relevant background information and related work.

Chapter 2, entitled *Modeling in Systems Biology – Fundamentals*, introduces fundamental information on modeling in systems biology including a short introduction to modular modeling. Furthermore, some biological concepts from the area in cellular signaling are presented.

Chapter 3, *Graphs and Visualization in Systems Biology – Fundamentals*, contains an overview of foundations of graphs including different graph classes, theoretical properties and background information on how graphs can be drawn. Furthermore, it is concerned with the foundations of visualization. Thereby, different aspects of ‘information visualization’ ranging from perceptual and cognitive principles to visual interfaces, are discussed. In the main part of this thesis, four separate chapters present the actual work and its results.

Chapter 4, *Visual Representation of Dynamic and Logical Modular Models*, presents formalized visual representations specifically adapted to dynamic and logical models.

In Chapter 5, entitled *A Visual Interface for Interactive Exploratory Model Analysis* a novel approach, which allows interactive navigation and exploration of modular models, is presented.

Chapter 6, *Visual Scenarios*, proposes a new approach for visual settings that defines primarily how geometrical objects are rendered. Hereby, different visual scenarios are discussed. Two examples are discussed already in Chapter 4.

Chapter 7, entitled *Layout Strategies for Modular Models* is focused on layout methods that are able to layout modular models and draw elements on different hierarchical levels. HyCOGLa and FlatLayout, two new layout methods, are described.

1. Introduction

At the end of each of Chapters 4-7 the applicability of the given approaches is illustrated, which shows their relevance for modeling projects in systems biology. Based on examples and case studies it is shown how visualizations support the modeler exactly.

Chapter 8 summarizes the contributions of the work and evaluates them in a larger context. Furthermore, possible future directions for research in this area are given.

Finally, Appendices A-D provide further information on the supplied CD, software architecture, visual representation of dynamic and logical models, visual scenarios, shortcuts and mouse controls and layout algorithms.

Since this work covers many visualization topics, Figure 1.1 illustrates the structure of the thesis in a visual sense. It depicts the relations and important dependencies of the different chapters and parts comprising the thesis.

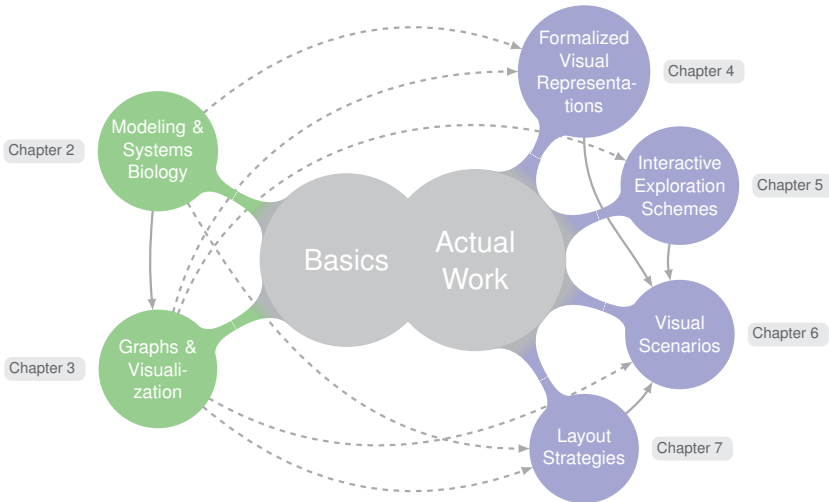


Figure 1.1.: Graph depicting relations and important dependencies of the individual chapters. Background information from different scientific domains: systems biology and related modeling techniques (Chapter 2), graph drawing and information visualization (Chapter 3). The actual work is divided into four chapters: ‘Formalized Visual Representations’ (Chapter 4), ‘Interactive Exploration Schemes’ (Chapter 5), ‘Visual Scenarios’ (Chapter 6) and ‘Layout Strategies’ (Chapter 7). ‘—>’ denotes a dependency between chapters of one part, either ‘Basics’ or ‘Actual Work’; ‘- ->’ denotes a dependency between chapters from both parts, ‘Basics’ and ‘Actual Work’.

1.4. VISUAL EXPLORER

VISUAL EXPLORER – newly developed as a software component of the modeling tool PROMOT – implements the approaches described in this thesis.

VISUAL EXPLORER uses a multilevel graph drawing approach based on the hierarchical and modular structure of the model and supports the interactive exploration of the graph representation. The developed techniques are designed for a general purpose and applied to different application areas including chemical engineering and systems biology. However, the main focus is on intra-cellular reaction systems in particular signal transduction processes.

VISUAL EXPLORER addresses mainly two challenges: first, the interactive visual exploration of model parts in the context of the entire modular model and second, a highly flexible handling of visual properties of the model tailored to the modeling task and individual preferences of the modeler. The former is discussed in Chapter 5; the latter is described in Chapter 6.

In further chapters, it is shown how the developed techniques are used for the adequate visual representation of the dynamic and logical modeling formalisms (Chapter 4) and for the visualization of data derived from diverse external investigations (Chapter 6). Furthermore, a strategy for drawing modular models in a ‘flat’ fashion is integrated (Chapter 7).

As already mentioned, VISUAL EXPLORER is part of the PROMOT software tool. PROMOT itself is freely available for download from <http://www.mpi-magdeburg.mpg.de/projects/promot/>. The source code of VISUAL EXPLORER is distributed under an open-source software license.

1. Introduction

Nowadays models should be consistently clear, reusable, maintainable and extendable. Reusable mathematical models become more critical to research, guidelines are needed to ensure that models can be easily combined to modular models.

Mike T. Cooling [208]

CHAPTER 2

Modeling in Systems Biology – Fundamentals

This chapter gives general background information on modeling in systems biology. It starts with the biological background, in particular, with information on processes of cellular signal transduction. Further, it gives a basic introduction on how new knowledge is generated, and in which way the knowledge is modified, refined and validated. It also highlights different model types, modeling approaches and existing modeling tools in the context of modular modeling. Finally, it introduces three different models of well-known signaling systems that are used throughout this thesis.

2.1. Biological Background

“In every second, a cell makes hundreds of decisions, based on its internal status and its inputs. The underlying decision-making mechanism is a complex network of molecular level interactions.” [42]

As stated in the quote, in a complex, intra-cellular network numerous entities such as macromolecules (e. g., proteins or DNA), small molecules (e. g., ATP or lipids) or physical events (e. g., heat or mechanical stress) interact in a very dynamic sense by regulatory processes. In the following, the intra-cellular processes on the molecular level are described in more detail.

2.1.1. Intra-Cellular Processes

Important processes inside a cell are *metabolic processes*, *signal transduction processes* and *gene regulatory processes*.

Metabolism Metabolic processes describe the complete interactions of metabolites (e. g., proteins or enzymes) by substance flow. Primarily, metabolic reactions are needed for the conversion of substances in the cell in order to store or release energy. Well-known metabolic processes, for example, are glycolysis or the citric acid cycle.

Signal Transduction Signal transduction describes the communication between proteins, enzymes and other parts within the cell by information flow. Signaling processes are very complex and regulate mainly metabolic processes. Different phenomena like inhibition and activation, or more complex patterns like feed-forward and feedback loops play an important role.

Gene Regulation Gene regulation (or transcription) processes describe relations between genes, in particular, transcription factor proteins and target gene products. Transcription factors regulate the transcription rate of genes according to external signals.

A cell can be divided into substructures and subsystems such as organelles and complexes. Because of its complexity even in subsystems, researchers introduce further abstract constructs such as pathways or motifs¹. In the remainder of this thesis the focus is on signal transduction systems and their inherent processes.

2.1.2. Cellular Signaling

Cellular signaling is the transmission and flow of information within the cell. Mostly signaling processes are started by an extracellular molecule that binds to a membrane receptor. The receptor transfers the information from the environment to the interior of the cell by modifying intra-cellular molecules. This process results in, for instance, changes in enzyme activity or gene expression.

Figure 2.1 depicts common signal transduction pathways found in an animal cell. Some of them (e. g., EGF receptor signaling or T Cell receptor signaling) are under extensive study and count among the best-characterized intra-cellular systems.

Sometimes signals pass a *cascade* of events within the cell (e. g., MAPK signaling cascade, or cyclic-AMP signal cascade). With the help of cascades the signal can be amplified in a stepwise manner; that is, a small stimulus can result in a large response. Signaling is also involved in periodic biological processes, such as the cell cycle regulation or circadian rhythms [71].

Signal transduction processes have a broad temporal scale ranging from milliseconds (e. g., for ion flux), minutes (e. g., for protein- and lipid-mediated kinase cascades), hours, or even days (e. g., for gene expression). The knowledge about cellular signaling results in a more holistic understanding of the intra-cellular processes. Sophisticated models of integrated cellular subsystems such as signaling feedback loops derived from knowledge of molecular interactions can be used to generate and test hypotheses. For instance, in EGF receptor signaling several models have revealed important features of the network such as modularity, redundancy and combinatorial interactions [36]. At the end it might contribute to medical science by providing insights into cellular function and dysfunction.

As stated in Gilbert et al. [71]: “Of all biochemical networks, signaling networks exhibit the highest degree of complexity. This is not only due to their non-linear network topology, but also because of the different types of interactions that these proteins can undergo: protein associations, enzymatic catalysis and reversible or irreversible protein modification, to name only the most common types.”

In this thesis the EGF receptor signaling and the T Cell receptor signaling are of particular interest (see highlighted areas in Figure 2.1).

2.2. The Modeling Process

About a decade ago the mathematical modeling of cellular processes was rarely used in biology. As a result of new technological advances (e. g., high-throughput measurement techniques) in screening the intra-cellular processes, engineering methods, in particular modeling and simulation, are nowadays important tools of research as in synthetic disciplines such as systems science, computer science or engineering [204].

¹A motif is a general principle that encodes basic topological patterns of processes in cells [6, 127]. Examples for motifs are time-delays, feed-forward and feedback loops, or biological switches.

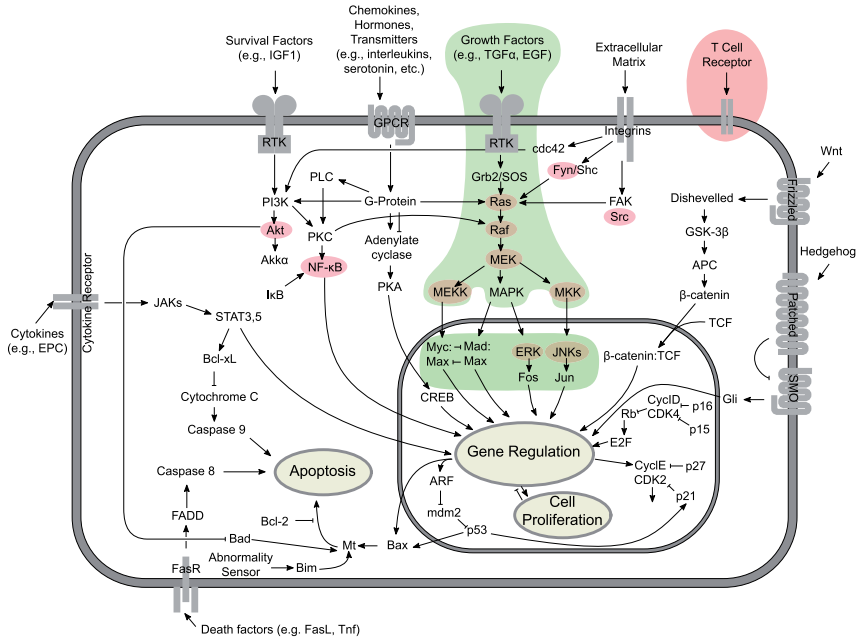


Figure 2.1.: Overview of common signal transduction pathways found in an animal cell. EGF (epidermal growth factor) and TCR (T Cell receptor) signaling processes are of particular interest. TCR signaling is only relevant for T Cells. Downstream events of EGF and TCR are highlighted in green and red, respectively. Adapted from Wikipedia [206].

In most cases *modeling* describes the whole modeling cycle as described in Section 2.2.1 including setup, analysis and simulation of a model – all of them are essential tasks in a project and an integral part of systems biology. From this point of view *systems biology* is systems science applied to biology. It combines experiment, theory, and modeling in order to understand cellular processes. More specifically, systems biology studies the structure, dynamics, and emergent functionality of the complex system and aims to understand the system as a whole.

The main purpose of modeling in systems biology is to organize, integrate and analyze the heterogeneous data from literature and experiments, and extract understanding about the cellular processes under investigation. The collected knowledge is primarily pooled in models, which in most cases are described by universal, mathematical equations.

2.2.1. The Process of Knowledge Generation

Research in systems biology, as in many other domains, is a strongly iterative and evolving process. It is characterized by a very high integration of experiment, theory and computation [99]. Different

2. Modeling in Systems Biology – Fundamentals

flavors of an ‘optimal’ process of knowledge generation exist, e.g. from the experimental [5] and the theoretical [2, 25, 73] point of view.

In this thesis the process of knowledge generation follows a theoretical or more systematic point of view and includes software engineering strategies. The following three premises are focused on.

1. The process is accompanied by different visualization techniques that are essential in order to support several steps.
2. The process is assumed as model-driven; that is, the knowledge and interpretation is based on the model. In this sense, the model is the central integrator that summarizes the current knowledge about a system under investigation.
3. The process focuses on a particular modeling approach – the modular modeling that results in modular models.

The overall process is depicted in Figure 2.2. In the beginning a cellular system is investigated and the problem, for example, a biological phenomenon, and corresponding tasks and aims are formulated. The first step in the process is the gathering of external data that can be of qualitative or quantitative type, for example, a-priori knowledge about the components, interactions and functions of the cellular system. External data might be obtained from different sources such as experimental investigations, computational prediction, human expertise, or public data repositories including a rapidly growing pool of databases² and peer-reviewed scientific literature. Often external data are in various scopes, time scales and levels of granularity. Additionally, data noisiness generated by modern high-throughput techniques poses significant problems [154, 184]. Consequently, based on the external data, several steps of data mining, data integration and data validation must be performed.

When sufficient, relevant details are gathered, a first draft of the model is set up. The model setup includes the careful analysis of the given data, the decision for the design of the model (specifying the modeling approach including scope and level of detail), and the precise implementation (construction, aggregation and calibration of the model). In some cases, several small (sub-)models or model parts (e.g., downloaded from different databases) must be converted into modules (with defined interfaces) for an integrated larger model. After setting up the basic model, a testing cycle follows including verification of topology or checking of equations, and if necessary a debugging step which, again, directly influences the model design or the specific implementation.

Based on the model, novel and useful hypotheses can be formulated. Hypotheses and predictions can be evaluated and cross-validated by further in-vitro, in-vivo or in-silico experiments. Cross-validation can be done automatically by, for instance, parameter estimation that tries to fit the results of a model simulation to the experimental data. Before performing in-silico experiments the modular model often must be optimized and flattened in order to run in existing simulation frameworks [73, 156]. In many cases, the experiments conducted give new observations that can strengthen or modify hypotheses or discover new facts. The interpretation of generated data resulting in new requirements and these, again, can be the input for an earlier stage in the overall process.

Such iterative processes of model review and refinement often introduce different model variants (or a family of models) which can result in models of the same system but with different granularities, time scales, parameter sets or even variations in the modular structure. If the basic model is changed, usually all derived versions have to be changed. In order to avoid redundant work, techniques from software engineering can be utilized [122]. These model variants must be managed in an efficient way, for instance, analogous to the development of computer programs in software engineering.

² 96 databases were registered in 2001 in NAR (Nucleic Acids Research) compared to 1380 databases registered in 2012 [66].

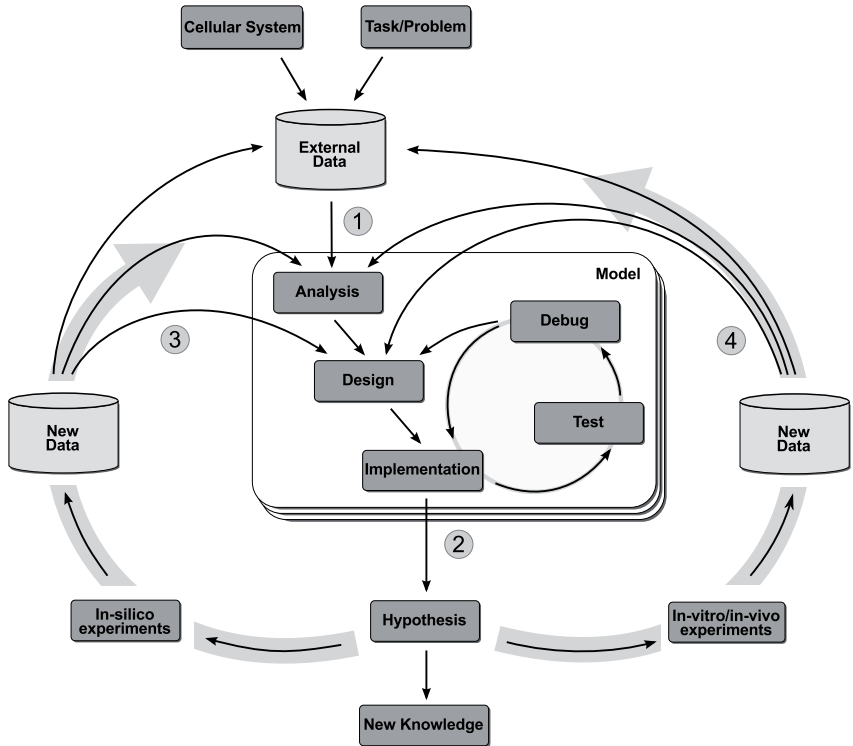


Figure 2.2.: Knowledge generation in systems biology as an iterative sequence of tasks. The figure depicts how new hypotheses can be an input to the earlier stages in the process. The overall impact of visualization is shown using circled numbers at different stages in the process. Container indicates data types. The process is depicted as model-driven. Compiled from a variety of sources [5, 25, 73].

Finally, new knowledge and insights might be generated. They can guide further research or lead to new drug target strategies in medical science or a potential production process in industry.

As stated in [?]: “Having no possibility to adequately visually explore the large amounts of data which have been collected because of their potential usefulness, the data becomes useless and the databases become data ‘dumps’.” Thus, different visualization strategies guide the distinct stages of the iterative process and primary support the generation of new knowledge and insights [183]. Major stages and their associated visualizations are listed in detail below.

2. Modeling in Systems Biology – Fundamentals

- ① The preparation of external data requires strong visual support. Huge databases such as KEGG, BioModels or JWS ONLINE provide visual interfaces for retrieving, filtering and exploring experimental or theoretical knowledge.
- ② The setup, editing and exploration of models is supported by tools providing different visualization methods. This facilitates the visual representation of the model structure and dynamics, and therefore a more intuitive and comprehensible modeling process.
- ③ The interpretation of results from in-silico experiments (e. g., data from qualitative analysis or simulation) are often not trivial. Hence, the adequate visualization of such data (e. g., overlaid on top of the network model) can lead to a deeper understanding.
- ④ Generated data from in-vivo and in-vitro experiments (e. g., omics data such as microarray data or proteomics) performed by a variety of technologies can be visualized either directly in the related instrument of the experimental setup or later, for example, in the context of the network structure of the model.

Corresponding to the list above, in Figure 2.2 circled numbers indicate different stages in the process of knowledge generation where visualization has a major impact. In this thesis, visualization challenges and solutions of stages ② and ③ are described and discussed in more detail. Stages ① and ④ are not in the scope of this thesis.

2.2.2. Types of Models

In this thesis an interdisciplinary research topic is discussed where in the different scientific domains the term *model* is used in quite different ways. Thus, the meaning of a model depends on a certain perspective, for instance, biology, mathematics or computer graphics. In general, a model is an abstract representation of objects and processes that describes the structure and the behavior of a specific system [105]. Thereby, a model focuses on the essential properties of the system relevant for the particular modeling task.

In research there are a plethora of different model types such as stochastic models, cellular automata, or Petri nets [106]. However, due to lack of space only model types relevant for this thesis are given and described in the following (see also Figure 2.3 for a pictorial summary).

Mathematical/Computational Model A *mathematical model* describes objects and processes based on mathematical equations. In contrast to mathematical models, a *computational model* benefits from its systematic structure and the resulting exploratory power. For instance, a systematic structure of the biological system like a modular approach can be easily exploited by graphical computer tools [29].

Dynamic Model The traditional approach describing biological and chemical systems familiar to most of the researchers is using algebraic or differential equations resulting in *dynamic models*. Several predefined kinetics can be used, e. g., mass action kinetics or Michaelis-Menten kinetics. For detailed information on dynamic modeling the reader is referred to Section 2.3.2.

Logical Model Another popular approach is using logical equations resulting in a *logical model*. It describes components and their relations but no dynamics (e. g., kinetic data). More information on logical modeling can be found in Section 2.3.1.

Modular Model Computational models can be defined using a hierarchical and modular structure resulting in *modular models*. They describe objects and processes on different levels and scopes, encapsulated in units also called *modules*. Detailed information can be found in Section 2.3.3. In contrast, a model without any modular structure is called a *flat model* or *simple model*.

Graphical Model A *graphical model* represents a mathematical/computational model in a diagrammatic form. In systems biology often graphical models are used to depict elements and the relations between them. Most common graphical models to describe cellular processes are *graphs* and more specifically *networks*. They are discussed in more detail in Chapter 3, Sections 3.2 and 3.5.1.

Biological Model A *biological model* does not describe the mathematical approach or the type of representation, but rather the biological system behind it. In this thesis, mainly the domain of cellular signal transduction is investigated. Hence, here ‘biological model’ means a model describing signaling processes in the cell. Detailed information about specific signaling models is given in Section 2.1.2.

Mental Model From each model type discussed earlier a *mental model* can be created. A mental model is a very simplified version with general assumptions of the underlying model. Related to graphical models and networks they are called *mental maps*. This special model type is further described in Chapter 3, Section 3.3.3.3.

2.3. Modeling Approaches

As stated in Wiechert et al. [204], this is no unique way to model a given system – modeling is always a creative activity in which a real system is interpreted in terms of mental concepts (e. g., mental models). Basically, computational modeling³ can be done in various fashions like mechanistic-based (dynamic modeling) or interaction-based (logical modeling). There are also approaches lying between the logical and dynamic modeling like constraint-based or rule-based modeling. All of them differentiate in their a-priori knowledge of the system and in their level of detail. The choice of the appropriate modeling approach depends primarily on the specific modeling task. For an overview of different approaches applied to systems biology the reader is referred to recent reviews [5, 71, 204] and the references therein. In the following, first, two popular modeling approaches are presented: logical modeling and dynamic modeling. Then the approach of modular modeling is introduced in detail.

2.3.1. Logical Modeling

Logical modeling represents a logical view of the system. It is the simplest view on a system of reactions and probably used as a first approach in a modeling project. More generally, it is also called *structural modeling*. When modeling large systems with few known kinetic mechanisms and parameters it is reasonable to consider solely structural and stoichiometric information of cellular networks independently of the kinetics. This more qualitative approach results in structural models that do not describe quantitative and dynamic properties. They only answer qualitative questions. As stated in Mirschel et al. [129], “for the setup of such models the literature at least in the area of systems biology offers plenty of information, which is also better comparable than kinetic parameters, measured under different conditions”.

³It means here the process of pooling the knowledge into a model.

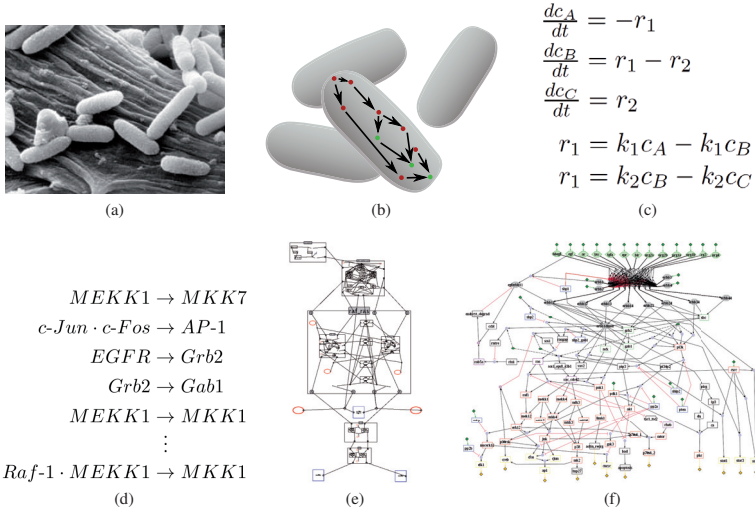


Figure 2.3.: Different model types: (a) real biological system, (b) mental model, (c) mathematical model using ODEs for a quantitative description, (d) mathematical model using logical equations for a qualitative description, (e) graphical model of a dynamic system, (f) graphical model of a logical system. Inspired by Klipp et al. [105].

Moreover, experimental data may be available only at discrete values, for example, a protein is active or inactive.

The logical modeling approach used in this thesis and the associated theoretical framework were elaborated in Klamt et al. [104]. It facilitates modeling and analysis based on a qualitative description in the form of logical equations. This approach provides on Boolean logic, but it is also possible to use multilevel logic. The modeling framework is based on different representations of signaling networks: *interactions graphs* and *logical networks*. In logical networks entities are combined with logical operators such as *and*, *or* and *not*. For analysis, methods of graph theory and structural analysis can be used.

2.3.2. Dynamic Modeling

Dynamic modeling represents a continuous view of the system – it is a more detailed view and not only requires information about the structure (relational data) but also the dynamics (kinetic data). Therefore, it is also called *kinetic modeling*. Mechanistic-based methods are the classical way of modeling biochemical reaction systems. Here, *ordinary differential equations* (ODE) or *differential algebraic equations* (DAE) are used to describe and analyze the dynamics and kinetics of the model. When modeling spatial information *partial differential equations* (PDE) can be used. Dynamic models can be either defined in deterministic or stochastic form.

In practice, often modelers encounter some problems when dealing with dynamic modeling. Kinetic data are hard to find (widely distributed among scientific publication), access (hard to get experimental design data) and unify (heterogeneous/ambiguous data, not standardized). For these reasons knowledge of kinetic laws and parameters is rare and of uncertain nature. If accurate kinetic data are not available, in some cases they can be substituted by standard kinetic laws. As described above, the accurate mechanistic modeling is difficult or sometimes even impossible due to the lack of quantitative knowledge. In contrast to logical modeling, this approach is not a broad, stationary map of many components and their interactions but rather a very detailed description of (mostly a few) intra-cellular components [181]. Logical and dynamic modeling approaches are very popular in the modeling community. Both of them can be combined with *modular modeling* which often results in a more enriched and systematic modeling workflow.

2.3.3. Modular Modeling

Modular modeling is well suited to describing modular structures of intra-cellular systems [79]. This is beneficial in order to model systems at different levels and granularity. Applying modular modeling, complex systems can be decomposed into smaller, simpler subunits resulting in a hierarchical and nested structure. Analogous to engineering disciplines⁴, these individual, reusable model parts can be studied in isolation, and tested and characterized separately prior to their incorporation into larger systems [113]. This principle is illustrated in Figure 2.4 where the overall process can be described on various hierarchical levels. For instance, basic or elementary units (e. g., transcription factor, target gene, binding site, certain reaction) are the basic elements that cannot be further decomposed into parts. From these, intermediate units that comprise specific network patterns (e. g., motifs like feed-forward loop) can be composed. Complex units (e. g., pathways, signaling cascades, compartments) again consist of intermediate units and present units that are independent in some way. At the top level of the hierarchy these complex units form the entire network.

Another example is the signaling EGF model published in Saez-Rodriguez et al. [164] and further described in Section 2.5.1. On the highest level, it was decomposed into five distinct functional units including a receptor and a MAPK cascade.

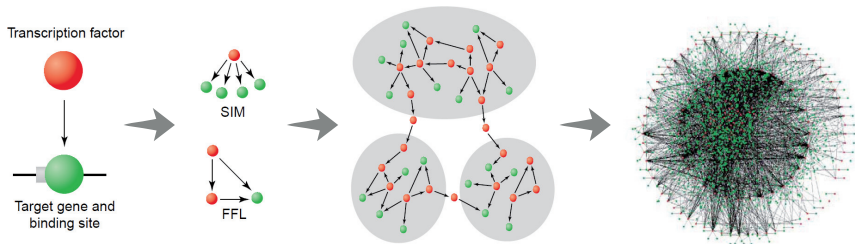


Figure 2.4.: Structural and hierarchical organization of a gene transcription network. From left to right: basic unit, motifs (single input (SIM), feed-forward loop (FFL)), composite modules, complex regulatory network. Adapted from Babu et al. [10].

The process of modular modeling results in *modular models*. A central component of modular modeling is the *module*. The term ‘module’ is rather general and can be defined from different point of views, for

⁴The idea is to break down a complex (engineering) problem into smaller manageable subproblems.

instance, biologically (physiologically) or theoretically motivated and for different hierarchical levels. When different hierarchical levels are discussed, modules can have different complexity (cf. Figure 2.4). Well-known examples of modules from systems biology are physical or spatial units (e.g., cell compartments such as cytoplasm or nucleus) and functional units where components contribute to the same task such as pathways, cascades or protein complexes. They can be found in metabolism [157], cellular signaling and gene regulation [79].

Common metabolic units. Elements that facilitate the same physiological/metabolic task can be grouped. Well-known examples are the specific catabolic pathways for individual carbohydrates, or metabolic functions, such as glycolysis.

Common signal transduction units. All elements of a functional unit are interconnected within a common signal transduction system. The signal flow across the unit border (i.e., cross-talk) is smaller than the flow within the unit. An example is the MAPK signaling cascade or the process of receptor binding.

Common genetic units. The genes for all enzymes of a functional unit are organized in genetic circuits; that is, they are expressed in a coordinated manner. Known examples are the machinery for protein synthesis or DNA replication.

Functional modules can also be motivated by a theoretical point of view. A promising property might be retroactivity⁵ [41, 163]. From the implementation point of view, modeling intra-cellular systems in a modular fashion has several advantages [73].

- The user works with comprehensible networks of modules rather than with reaction networks with hundreds or thousands of parts. In most cases, it is easier to distribute tasks among different modelers working on parts of the same system. That is, responsibilities and limitations are well defined and documented.
- The interface of a module can be specified separately from its implementation. This leads to a simplified exchangeability of modules with the same defined interface, but with a different granularity or scope. In this way, a model part can be simplified or extended.
- It is easier to separately test a specific, simple-structured module with its input-output behavior in a well defined test environment before several modules are combined into a larger system.
- The modular structure forms an inherent topological property of the model. This can be easily visualized and help to guide the user, for example, to hide unnecessary details within modules or navigation cues based on the modular structure.

As pointed out in Alberghina and Westerhoff [3], a major challenge is to find objective criteria for the demarcation of modules. In literature, several approaches can be found. For instance, modules can be created manually, calculated using computational power [162] or measured directly, for example, using experimental techniques.

In summary, modular modeling provides a strategy to set up composite models where the detailed functionality is encapsulated in modules and different abstraction levels. In this way, model complexity can be managed, for example, by hiding it from the user [209].

⁵Retroactivity means, for example in cell signaling, an effect on the functioning of an upstream module that comes from a downstream module [162]. Thus, ideally modules without retroactivity could be studied and analyzed independently of each other.

2.3.3.1. Reuse of Modules

In intra-cellular systems often units (i. e., group of elements) representing a known biological mechanism that occurs repeatedly [29, 208]. As already pointed out, such units are, for instance, motifs that are frequent in many systems. Once such units are found, they can be systematically analyzed, classified and implemented as modules (e. g., in a specific library) by skilled modelers. Afterwards, these modules⁶ including their complex mathematical description can be reused. This has several advantages:

- It is feasible to easily build upon pre-defined functional components organized in libraries. There is no need to start from scratch [209].
- Considerably, it simplifies and fastens the setup of large models because the module description (relations, parameters, etc.) is defined once but can be used as many times as required within a model [208]. In this way, specific details of biological concepts can be distributed within a larger model.
- Reused modules have an inherent consistence. The information within the module must be checked for correctness only once and can be used several times without further in-depth checking.

When considering models as composite modules that can be integrated, combined and adapted, the need to reuse modules becomes essential [100].

2.3.3.2. Modular Modeling Concept

The *modular modeling concept (MMC)* [73] bases on the network theory [72, 124] and the idea of hierarchical and modular modeling outlined in Section 2.3.3. Hierarchical aggregation of modules can be used to set up more complex processes such as signaling cascades [110]. The most important modeling entity of the MMC is the *module*. Modular models are mainly composed of modules.

The MMC is inspired by software engineering strategies. In particular these are: (i) work with comprehensible modules in order to manage complexity, (ii) define and document the responsibilities and limitations of a module, (iii) test the modules separately, (iv) keep interfaces stable, and (v) reuse modules.

2.3.3.3. Flattening a Model

When setting up models with hierarchical and modular structure it is important to have methods that are able to remove structure. The process of *flattening* converts a structured, modular model into a model without hierarchical and modular structure. The explicit description of coupling elements (relations among the submodules) is lost in the sense that they are implicit in the single large (flat or simplified) model after converting. Hence, the information on how the model was composed is lost.

As highlighted already in Section 2.2.1, flattening a model is valuable since it allows optimized and efficient model code to be generated for existing simulation tools⁷ which have no support for composition or aggregation [156]. In modular models this can be done by obtaining a global DAE system generated from all local equations of the modules. In some cases, algebraic equations are used to describe the coupling of modules. They are simplified or removed in the process of flattening [124].

⁶They may appear across different models or several times in a single model.

⁷In contrast, in multi-scale modeling or agent-based modeling the hierarchical model structure is useful for simulation [156].

2.4. Modeling Tools and ProMoT

Different software projects exist that provide modular modeling or related strategies mentioned in the previous sections. A great deal of work on such systematic approaches has been done in software engineering and other engineering fields. However, in the following, the overview is limited to tools developed for system biology. Several of these software projects are introduced first. Afterwards ProMoT, a modeling tool that combines many of the systematic approaches, is described in detail.

2.4.1. Modeling Languages, Tools and Formats

In recent years many tools for solving systems of ODEs and DAEs were developed. Considerably fewer tools are developed for the support of logical modeling. Even fewer tools are dedicated to modular or object-oriented modeling. An overview of modeling tools in general can be found in Alves et al. [7] and Kim et al. [98], or, with a focus on simulation, in Pettinen et al. [147].

From the diversity of these tools a few representatives are highlighted in the following sections. They are divided by their major characteristics: modeling languages, specialized modeling tools, exchange formats, and analysis/simulation tools. Software tools with intensive use of graphical representations are discussed in Chapter 3, Section 3.6.

2.4.1.1. Modeling Language: MODELICA

MODELICA [61] is a general modeling language used in several commercial tools, for instance, in OPEN-MODELICA [132] or DYMOLA [9]. MODELICA supports an equation-based modular specification of arbitrary hierarchical DAE models and object-oriented methods. It is a general-purpose tool, not specialized to modeling in systems biology.

2.4.1.2. Modeling Tools: ANTIMONY, LITTLE B and E-CELL

ANTIMONY [180] focuses on the modular construction of models for systems biology. The construction is performed using a text editor. No graph-based setup is provided. ANTIMONY is able to convert to SBML models by flattening the modular model.

LITTLE B [123] is a modular modeling language that can be used to build mathematical models of complex systems with a focus on systems biology. However, it lacks an appropriate graphical representation. E-CELL [194] provides a modular approach for modeling where the complete model is composed of compartments. These compartments are almost physically motivated; that is, a module represents a compartment from the topology of the cell.

2.4.1.3. Exchange Formats: CELLML and SBML

CELLML [120] has been designed to support multi-scale modeling. It is specifically used for representing intra-cellular processes in systems biology [138]. CELLML models are based on ODEs. It facilitates modular modeling by construction of model parts independently of each other and their later integration into a core model. The modularity concept supports the use of building blocks, encapsulation and different import mechanisms for easier integration. For established tools using CELLML the reader is referred to Garny et al. [67].

SBML [86] is a community-driven quasi-standard for describing elements in biochemical networks and its kinetics. The SBML consortium is currently working on an extension of the SBML syntax called ‘Hierarchical Model Composition’ in order to describe modular structures of models, for example, include models as subunits within other models.

2.4.1.4. Analysis/Simulation Tools: DIANA and CELLNETANALYZER

DIANA [109] and its predecessor DIVA [133] are intended for stationary and dynamic simulations and analysis of higher-order DAEs mainly for engineering and cellular processes. DIANA is a flexible and extensible numerical toolbox. Major features are an object-oriented architecture, the handling of equation-based models and the enhanced scripting facilities. DIANA is based on free numerical tools and hence provides efficient and sophisticated methods for numerical solutions. However, there is no support for visualizations.

CELLNETANALYZER [103] is dedicated to the structural and functional analysis of cellular networks. CELLNETANALYZER is used for the analysis of metabolic but increasingly also for signaling networks. Most analysis methods are performed on the network topology and do not consider kinetic mechanisms and parameters. CELLNETANALYZER supports two modeling frameworks for signal transduction systems: interaction graphs and logical networks. Thereby, an interaction graph is underlying the logical graph. Both are causality models and enable the computation of signaling pathways. The CELLNETANALYZER format is used for the storage of logical models built up by CELLNETANALYZER and as an exchange format for logical models. CELLNETANALYZER uses text boxes for visualizing elements and their properties. Editing is performed by GUI dialogs. Additionally, it has the possibility to place a background image for illustrating the respective model.

2.4.2. Modeling with PROMOT

The Process Modeling Tool (PROMOT) [130] is a general computational tool for modeling processes in complex technical⁸ and biological systems⁹. It facilitates the intuitive setup and editing of complex mathematical models. PROMOT supports several concepts to work with these complex models in an efficient manner. It facilitates a combination of the MMC and object-oriented modeling. Hereby, the modular structure of processes is defined and presented by hierarchical levels with varying organization and abstraction. The modular model structure consists of elementary building blocks (modules) organized in flexible modeling libraries, and concise interfaces. Modules which have been proven to work can and should be re-used. In contrast to E-CELL where modules are physically motivated (e.g., compartment), in PROMOT the definition of modules is more general.

Modeling with PROMOT is equation-based. Thus, the equations described in the model are used to describe, manipulate, analyze and optimize the system¹⁰. In this sense, PROMOT has a similar approach as MODELICA. In PROMOT dynamic modeling, based on a set of ODEs, algebraic equations and Petri nets (for event handling), is provided in order to describe the behavior of the system. In addition to the kinetic modeling, PROMOT also supports a logical modeling formalism elaborated in Klamt et al. [104], which results in a qualitative description of the model [165]. Spatially distributed systems and stochastic modeling is not supported.

Models can be set up either text-based using the Model Definition Language (MDL), or diagrammatically using a graphical editor. Since both methods work directly on the same data representation, the modeler can switch between these two ways of modeling. MDL is a declarative, object-oriented language used by PROMOT, in particular as storage format of libraries and models. More details on the graphical setup of models in PROMOT can be found in Chapter 3, Section 3.6.3.

⁸Applications in process engineering are, for example, reaction and separation systems, membrane reactors [125] catalytic reactors and fuel cells [78].

⁹Applications in systems biology are, for example, intra-cellular reaction systems such as metabolic systems like *E. coli* or yeast, and complex signal transduction processes [110, 164].

¹⁰Mechanistic modeling can also be done based on reactions. For example, SBML has a reaction-based approach.

PROMOT was designed for the modeling of engineering systems that incorporate a natural modular and hierarchical structure. For instance, elementary modules can be assembled into units which again can be connected to submodels (modules) on the next-higher hierarchical level, etc. Hence, the entire system is composed of modules. This concept was transferred to cellular systems [110] as depicted in Figure 2.5.

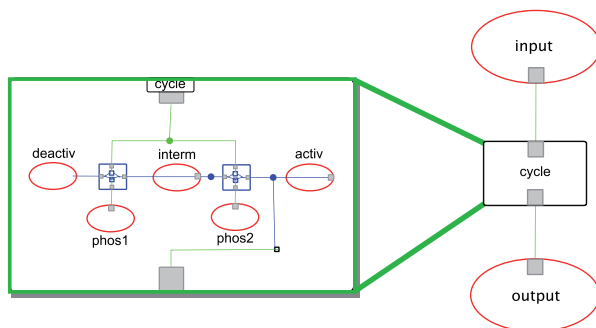


Figure 2.5.: Hierarchical and modular modeling with PROMOT. The module `cycle` encapsulates all elements that have the same functional concept. This results in a submodel (module) that is connected to other modules (e. g., `input` or `output`) at the highest organization level.

A known extension to modular and hierarchical modeling is the object-oriented concept using strategies from software engineering including *abstraction*, *encapsulation*, *aggregation*, and *multiple inheritance*. A powerful feature is abstraction where an abstract description is defined and then detailed instances are generated. For example, the user can set up a module describing a general enzymatic reaction. In a next step the main structure and behavior can be inherited which results in specialized types for reactions with different parameters and kinetics [29]. This concept can also be utilized for the generation of model variants. Class derivation can be used to quickly exchange kinetic laws and parameters while keeping the interface structure and the connectivity fixed [204].

2.4.2.1. The Modeling Workflow

The modeling workflow using PROMOT follows the general scheme depicted in Figure 2.2 and is illustrated in Figure 2.6. First, knowledge about a system (external data) must be imported into PROMOT. This can be done either by building up a model from scratch or by importing from several external resources.

A model can be set up from scratch using basic modeling entities taken from a standard modeling library or knowledge base¹¹. This is analogous to stages ② and ③ in Figure 2.2. Data import/integration can be performed through a generic SBML parser, or a more specific CELLNETANALYZER parser depending on the particular modeling approach. When the setup is finished, the model can be exported to different analysis and simulation tools, for example, DIANA [109], MATLAB or CELLNETANALYZER [103]. Before export, the model is flattened and optimized for efficient use in the respective tool. Finally,

¹¹ PROMOT provides a standard modeling library for each modeling approach (cf. Section 2.3). A knowledge base is a representation of the existing biological understanding. It also encodes the current understanding and expertise of the modeler.

externally generated data can be loaded and visualized in PROMOT and may lead to the refinement of the given model. In PROMOT a model consists of different basic elements that define the structure of the model – *modules*, *terminals* and *links*.

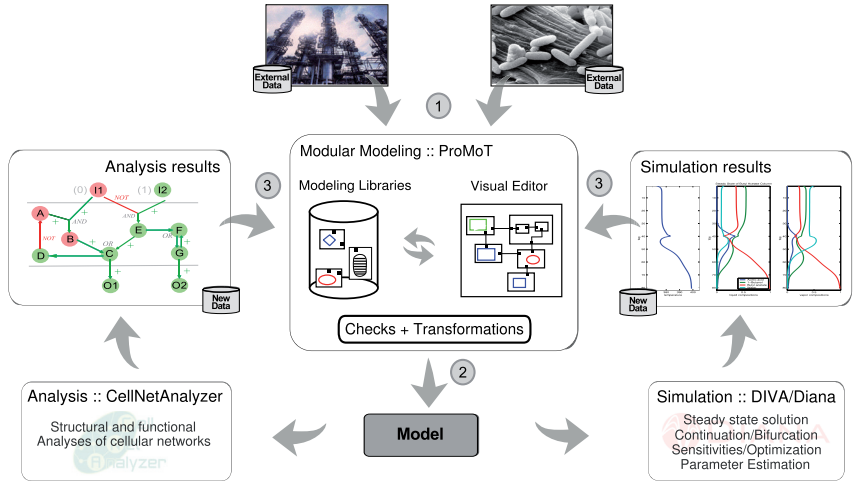


Figure 2.6.: Schematic view of the modeling workflow in PROMOT. A model can be set up either from scratch or using the import functionality (stage ①). Output from PROMOT are hypotheses in the form of model variants (logical/dynamic models), which can be evaluated by different computational methods, for example, analyses or simulation (stage ②). New-generated data can be visualized, again, in the context of the network model (stage ③). Parts taken from Mirschel et al. [129].

2.4.2.2. Modules, Terminals and Links

Modules are the most important modeling entities and are used to encapsulate a group of modeling elements which contain, for instance, reactions, species and/or signaling parts. They can be divided into *elementary* and *composite* modules. Elementary modules do not contain submodules, whereas composite modules have a complex structure with inner submodules. The coupling of modules is facilitated through special interface objects called terminals [73].

A terminal¹² is an interface for coupling one module to another. Through terminals, part of the inner state of a module is made accessible to the exterior part of the system [73]. Without terminals a module is separated from the model. It facilitates efficient modification and model variants by using modular plug-and-play components using standardized interfaces.

A link connects two or more terminals and the defined variables. In dynamic models the direction of information flow is encoded directly within the behavioral description of the module. In a biological

¹²In MODELICA terminology an interface is called a *connector*. It is similar to *ports* known from graph theory.

context a link between two entities implies a certain type of interaction such as stimulatory, inhibitory or catalytic.

2.4.2.3. Architecture

PROMOT is designed as a server-client architecture. The communication between server and client is provided via a CORBA middleware [135]. The PROMOT server (written in LISP) performs the management and manipulation of models. The client which is implemented as a GUI (written in JAVA) displays the modeling library and the model itself and facilitates the interaction with the modeler. The GUI can be divided into the *Class Browser* (management of models and modeling libraries) and *Visual Editor* (generic graphical setup and editing of models). The detailed architecture is not discussed here since the focus is on visualization. For further information the reader is directed to Appendix A containing diagrams depicting the general architecture and class structure.

For a summary of features and functions of PROMOT, and also for more detailed information the interested reader is directed to the given literature [73, 129, 130, 165].

2.5. Application to Models

Throughout this thesis three different models of well-studied signaling systems are used to illustrate the developed concepts as well as to show the successful application of the described methods. In particular, the models are two *Epidermal Growth Factor Receptor* (EGF, EGFR/Erbb) models and the *T Cell Receptor* (TCR) model. The models describe intra-cellular signal transduction processes of different functions and at different levels of detail. These models are amongst the best-characterized systems in signal transduction. For more detailed information about the models and the biological mechanisms behind them, the reader is referred to the literature [164, 166, 167].

2.5.1. The EGF Receptor Models

The EGF Receptor signaling network is the best-studied system in mammalian cells. Its growth and differentiation processes are involved in human cancer; some potential drug targets within this network are already located. In recent years the knowledge about the EGF Receptor signaling has increased to a point that a description in terms of an integrative systems biology approach is feasible [36]. In the following, two EGF Receptor models, EGF and EGFR/Erbb, are described. They differ in the modeling approach, scope and the level of detail. Important pathways of the EGFR/Erbb signaling are illustrated in Figure 2.7.

The EGF model, developed by Saez-Rodriguez et al. [164], uses several ODEs allowing a dynamic characterization of the signaling behavior. This allows a very detailed description of the biochemical processes involved in the EGF Receptor signaling (e.g., phosphorylation of proteins using kinetic parameters). The model describes the activation of the mitogen-activated protein kinases MAPK signaling cascade by EGF. In more detail, it includes the binding of the EGF ligand on the `erbbl1` receptor, the formation of signaling complexes by interaction of several signaling proteins (namely the kinases `shc`, `grb2` and `sos`), the activation of a signaling intermediate called `ras` and the initialization of the MAPK cascade as the core unit involving the kinases `raf`, `mek` and `erk`. It is a modularized version of the EGF model from Schoeberl et al. [173]; that is, important signaling units are encapsulated into modules. In contrast to the Schoeberl model, the EGF model also includes a detailed description of the lysosomal degradation and the internalization of protein `erbbl1` because this is considered as important for the dynamic behavior of the overall EGF Receptor signaling. The EGF model consists of 73 modules (including five high-level components) on five hierarchical levels. The internalization is considered in

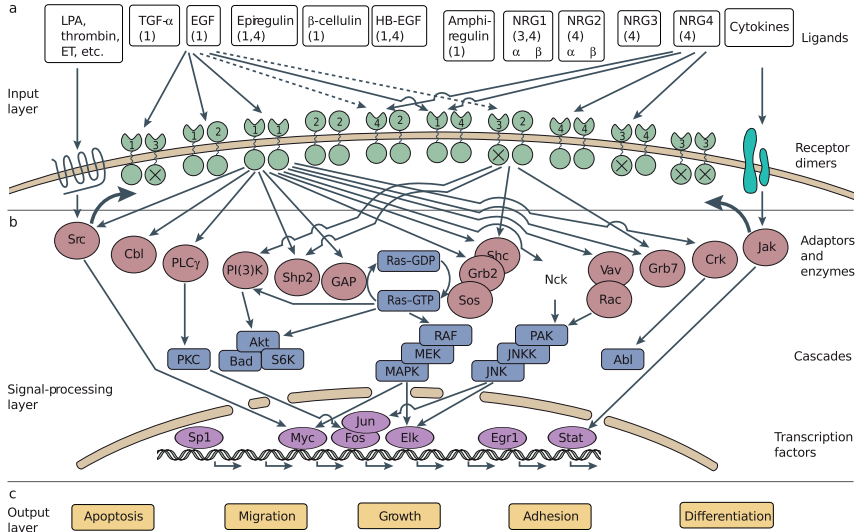


Figure 2.7.: Network of the EGFR/Erbb signaling depicting the most important pathways. The signaling network is divided into (a) an input layer, (b) a signal-processing layer and (c) an output layer. The input layer shows the ligands outside the cell and the receptor dimers at the cell surface. The intermediate layer comprises all signaling processes within the cell including enzyme activity and different cascades. The output layer points to different biological mechanisms like apoptosis or growth. Taken from Yarden and Slivkowsky [210].

all main modules. This results in a duplication of nearly all signaling steps described above (i.e., internalized and non-internalized) and an increased complexity of the model. This model is most suitable for illustrating the general concepts outlined in this thesis because of its complex hierarchical and modular structure.

The EGFR/Erbb model, developed by Samaga et al. [167] applies a qualitative formalism of the EGF Receptor signaling. Here, interactions of species are described in the form of logical equations. In contrast to the dynamic EGF model, it describes the processes in less detail but with a broader biological scope. For instance, the model involves the complete ligands family, a variety of receptors (including all four known EGF Receptor types), some players of the transcription level (the activated signaling molecule *erk* can translocate to the nucleus and there regulate gene expression) and additional signaling pathways such as *pi3k/act* or *jnk/MAPK*. The model includes two main modules describing the overall network structure and the combinatorial complexity formed by the interplay of the 13 growth factors and four receptor types at the uppermost layer. It comprises 104 species and 204 interactions.

the domain of systems biology or a method to define modules of general granularity or scope. A tool dedicated to modular modeling independent of the domain and scope is PROMOT.

The graphical representation of biological networks is a topic that has been largely neglected, and its importance has only recently been recognized because of the growing need to understand large scale biological networks depicted by genome-wide analysis and other comprehensive measurements.

Hiroaki Kitano [102]

CHAPTER 3

Graphs and Visualization in Systems Biology – Fundamentals

In many scientific and engineering domains ranging from physics to geographic information systems to software engineering, complex relational data structures are an integral part of day-to-day research and therefore have to be modeled and visually represented. Today, more and more applications can be found in the areas of modern bioinformatics, computational biology and systems biology. Especially in the latter disciplines a huge amount of relational data are produced due to technology advances and integration efforts.

For describing biological entities and their complex relationships, *graphs* can be used. Graphs are high-level constructs that provide a very intuitive and common representation, and therefore allow the user to explore and analyze the underlying data items and their relationships in convenient and comprehensible ways. Furthermore, powerful methods from graph theory can be used to analyze the data in a systematic manner. In order to elucidate the properties of complex systems the drawing of graphs plays a central role. Using the graph metaphor as a visual interface, the large and complex structure of systems can be visually depicted and, if necessary, explored further. The work of Di Battista et al. [45], Díaz et al. [46], Herman et al. [81], and von Landesberger et al. [201] provided comprehensive surveys on theory, layout, visualization and navigation of graphs.

In this chapter the fundamentals of graphs including different graph classes, theoretical properties and background information on how graphs can be drawn are introduced. In this context, important layout strategies are given. In the second part, different visualization aspects and techniques are presented. The focus is on information visualization, perceptual and cognitive principles, visual interfaces and interactive exploration. A final section on visualization tools for systems biology complements this chapter.

3.1. Graph Theory

Graphs are commonly used to represent data items (vertices) and relationships between them (edges). Relations can be subdivided into *adjacent relations* – correlations between pairs of elements, and *inclusion relations* – hierarchical correlation among elements. For the representation of biological systems different graphs are in use. Hu et al. [84] provided a classification which differentiates graphs according to their structure into *simple graphs*, *multigraphs*, *compound graphs*, *metagraphs* and *hypergraphs*. In Figure 3.1, examples of the different graph classes are given. Whereas simple graphs and compound graphs are applied frequently, hypergraphs are rarely used in systems biology. Also metagraphs are ap-

plied rarely but they seem to have some potential [84]. Furthermore, according to Di Battista et al. [45] and more recently von Landesberger et al. [201] graphs can be distinguished into *dynamic* and *static* graphs. In dynamic graphs the graph structure, the properties of vertices and edges, or both can be affected depending on time. Contrary, static graphs do not change over time.

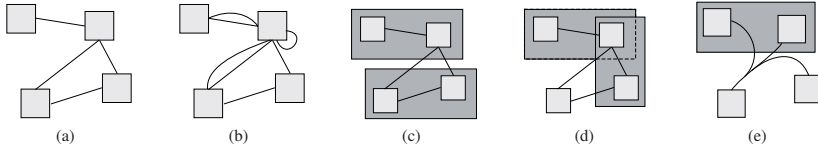


Figure 3.1.: Classification of graphs currently used in systems biology: (a) simple graph, (b) multigraph, (c) compound graph, (d) metagraph and (e) hypergraph. (a) and (b) depicts graph classes without any hierarchy whereas (c), (d) and (e) shows graphs that are able to represent inclusion relations; dark boxes depict subgraphs. Adapted from Hu et al. [84].

In the following section graph classes important for the area of systems biology are briefly introduced. Thereby, graph classes describing hierarchical structures are in focus.

3.1.1. Graph Classes

Simple graphs are the smallest common base for all classes of graphs and are the simplest form of a graph. They do not have any hierarchy, no multiple edges between pairs of vertices and no loops¹ (see Figure 3.1a). *Multigraphs* are an extension of simple graphs and allow multiple edges between pairs of vertices and loops (see Figure 3.1b).

Certainly, there are many possible extensions to this very basic concept. When graphs become huge and more complex, their handling can be eased by introducing a hierarchical structure, for instance, decomposing the graph into several subgraphs and, if necessary, decomposing each of them into further subunits². In this way, a graph can be created which describes model components at different levels of organization and abstraction. That results in clear and small subgraphs which are more manageable and comprehensible (cf. Section 2.3.3). The process of decomposition leads to an inclusion tree encoding the hierarchy.

A *compound graph* extends the multigraph approach by *subgraphs*. Each of the vertices can have any number of subgraphs. However, a subgraph may belong to only one supergraph. In Figure 3.1c and Figure 3.2d, two examples are shown. The compound graph facilitates the unification of a graph (network) and a tree (hierarchy) that results in a single representation; it allows subgraphs which can be nested. The nested subgraphs of the compound graph form the hierarchical structure, while the edges represent relations between subgraphs. Compound graphs can be further differentiated into *cluster graphs* and *nested graphs*. A cluster graph is a compound graph where edges are only allowed between leaves of the decomposition tree. In contrast, a nested graph has edges only between children of the same parent. A subgraph in a compound graph can also have defined *ports* for interfacing to other subgraphs. By using ports in visualization, additional constraints on how subgraphs are connected can be defined. The ports of a vertex are children of that corresponding vertex. An edge connects a *source* port and a *target* port.

¹Edge which starts and ends on the same vertex.

²There exist various ways to decompose a given graph, for example, using predefined hierarchical structure, or clustering based on graph properties.

Compound graphs do not allow an overlapping of subgraphs. Thus, a vertex cannot belong to different subgraphs of the same hierarchical level.

However, for some modeling tasks an overlapping may be useful. This restriction can be solved by *metagraphs*. Metagraphs like compound graphs facilitate adjacent as well as inclusion relations (see Figure 3.1.d). *Hypergraphs* represent a generalization of different graph classes. In hypergraphs individual edges, called *hyperedges*, can connect to any number of vertices (see Figure 3.1.e).

The transformation of simple graphs to hierarchical graphs (introduction of hierarchy, clustering) and vice versa (removal of hierarchy, flattening) is an important topic. Especially for the latter, only limited research has been carried out. In the following section important theoretical properties of graphs are presented.

3.1.2. Characteristics of Graphs

A *graph* $G = (V, E_G)$ is defined by a finite set of vertices V and a finite set of edges E_G . An edge $e \in E_G$ is an (un-)ordered pair of vertices (u, v) with $u, v \in V$. A *subgraph* $G_S = (V, E_G)$ is a graph whose sets of vertices and edges are subsets of G . An *inclusion tree* $T = (V, E_T, r)$ is a *rooted tree*. For every vertex $u \in V$, except the root vertex r , there exist a unique vertex $v \in V$, called the parent of u , such that $(u, v) \in E_T$. If u has no child, u is called a *leaf* of the tree.

A *directed graph* (or *digraph*) is defined by a set of vertices V and a set of ordered pairs (u, v) with $u, v \in V$. The vertex u is the *source vertex* and v the *target vertex* of the edge. Directed graphs represent structures having a general flow (e. g., of information or direction). In contrast, *undirected graphs* are defined by unordered pairs of vertices.

A *compound graph* $G_C = (G, T)$ is described by a (un-)directed graph $G = (V, E_G)$ and an inclusion tree $T = (V, E_T, r)$ that share the same set of vertices. The elements of E_G are called *adjacent edges*; those of E_T *inclusion edges*. A *cluster graph* G_{CL} is a specialized compound graph with $G_{CL} \subseteq G_C$, where adjacent edges of E are only incident to leaf vertices of T . A *port* $p \in P$ is a special vertex which is an interface of G_S to the exterior. A *terminal cluster graph* G_{TCL} is a specialized cluster graph G_{CL} where ports exist $G_{TCL} = (V, P, E)$, with $P \subseteq V$. Every compound graph G_C can be transformed into a terminal cluster graph G_{TCL} where each time an edge is intersecting the border of a subgraph a port $p \in P$ is inserted.

Two vertices u and v are called *adjacent* if they share a common edge $e = (u, v)$. The *degree* k of a vertex v is the number of adjacent vertices. For directed graphs the degree can be further differentiated into *in-degree* and *out-degree*. The in-degree k_{in} of a vertex v is the number of adjacent nodes with v as the *target vertex*. The out-degree k_{out} of a vertex v is the number of adjacent nodes with v as the *source vertex*. The sum of k_{in} and k_{out} is the degree k of the vertex v in the underlying undirected graph. A vertex v with $k_{in} = 0$ is called a *source*, as it is the origin of each of its edges. Similarly, a vertex v with $k_{out} = 0$ is called a *target*. The degree is equal to the number of direct *neighbors* of a vertex v . The *neighborhood* of a vertex $N(v)$ is a subgraph G_S with a set of vertices adjacent to v . G_S does not include v itself.

The *size* of a graph $|G|$ defines either the number of edges $|E|$, or more commonly, the number of vertices $|V|$ in G . A *path* on a graph G is a sequence of consecutive vertices and edges $v_1, e_1, v_2, \dots, v_{n-1}, e_n, v_n$ with $e_i = (v_{i-1}, v_i), 1 \leq i \leq n$. v_i are vertices while e_i are edges of the graph such that vertices and edges adjacent in the sequence are incident. The length n of a path is the number of edges $|e_i|$ traversed. A *shortest path* between two vertices u and v in a directed graph is a path from the source vertex u to the target vertex v where the number of edges is minimal. The *hierarchical level* L_T of a vertex v in T is the length of the path between the root vertex r and the vertex v . By convention $L_T(r) = 0$. The *depth* of T is the maximum of L_T for all vertices in T .

A *drawing of a graph* D is a representation of the graph in (usually 2D) Euclidean space $D : G \rightarrow \mathbb{R}^2$, including the drawing of the underlying graph G (and of the inclusion tree T). Each vertex of G is represented as a node; each edge (u, v) is represented as a curve between $D(u)$ and $D(v)$.

3.2. Graph Drawing

One important subarea of graph theory is *graph drawing* where relational structures (e. g., protein A binds to protein B) modeled by a graph are presented in a visual manner. This has several advantages: Using visualization the powerful human perception system can be used. It processes visual representations more comprehensible as formal descriptions such as formula or text lists. Furthermore, visual data mining is often more effective because similarities, correlations or interrelations in complex graph data can be found faster. Finally, a ‘good’ visualization results in an increase of usable information that can help solve the investigated problem.

Exemplarily, different graphical representations of a compound graph are shown in Figure 3.2. For instance, Subfigures 3.2a and 3.2b depict a hierarchy of vertices in the form of a tree without adjacent relations. In contrast, Subfigures 3.2c-3.2e depict adjacent and inclusion relations, whereas Subfigures 3.2d and 3.2e additionally encode layout information.

For the purpose of visualizing graphs, nodes and edges have visual properties like position, color, transparency or line width. In Figure 3.3, an overview of visual properties is given. How these visual properties can be applied in a useful manner using perceptual and cognitive principles is discussed in Section 3.3.3. In the following, different graph layout methods are introduced.

3.2.1. Graph Layout

The layout of the graph is not defined by the graph itself. There are many possible drawings for a particular graph definition. Hence, the problem in graph layout is finding appropriate 2D- (or 3D-)coordinates for the components of a given graph. With an adequate layout the user has an improved understanding of complex data encoded with the graph while with a poor layout the user may not comprehend the important aspects of the data.

A layout defines a position for each node and the route for each edge. The route is in the simplest case a straight line between two nodes. The layout can be done manually. In small graphs manual layout is generally straightforward; but it is a difficult task in large graphs and even more challenging in complex-structured graphs. Then, it is often a tedious and error-prone process. A computed or (semi-)automatic layout would be more convenient.

The automatic layout of graphs is a well-studied problem and a large research area. An extensive survey of proposed methods is beyond the scope of this thesis. Herman et al. [81] provided a profound overview of work on visualizing graph hierarchies. A more general overview of graph drawing problems can be found in the books by Di Battista et al. [45] or Kaufmann and Wagner [94]. A review of available methods in graph drawing related to molecular biology was provided by Uetz et al. [199].

Graph layout methods try to satisfy specific *aesthetic criteria* [143]. The compliance with different criteria is a requirement to obtain aesthetically pleasing drawings of a graph. Aesthetic criteria are needed for quality measurements and performance evaluation. They are investigated in the following.

3.2.1.1. Layout Aesthetics

For finding suitable positions of nodes and edges one or more layout aesthetics can be considered. They can be described and measured by several criteria. A list of relevant aesthetic criteria is given below, taken from several sources in literature, for example, [45, 94, 143] and more recently [16].

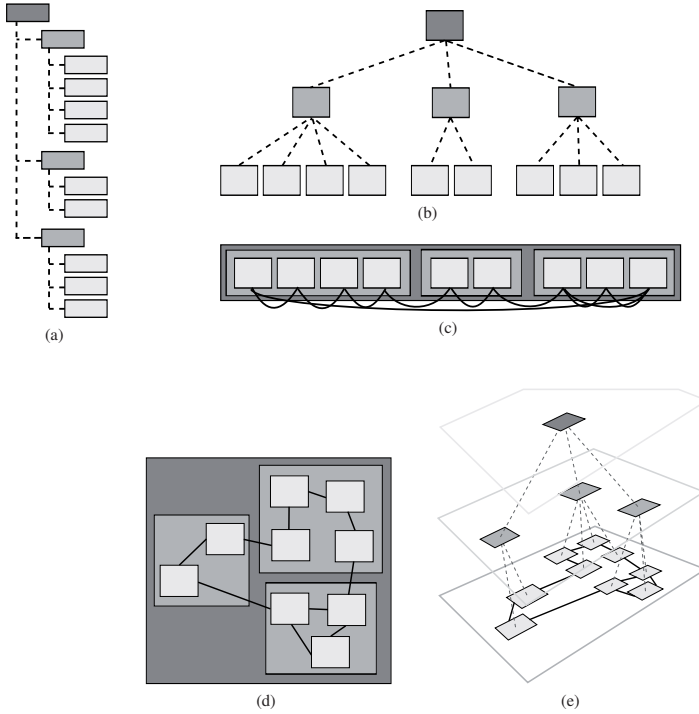


Figure 3.2.: Different views of the same hierarchical graph with adjacent relations (drawn as solid lines) and inclusion relations (drawn as dashed lines): (a) tree browser view (2D, explicit representation of inclusion relations, no adjacent relations, no data layout), (b) tree view (2D, explicit representation of inclusion relations, no adjacent relations, no data layout), (c) arc tree (2D, implicit representation of inclusion relations, adjacent relations, no data layout) [136], (d) nested graph view (2D, implicit representation of inclusion relations, adjacent relations, data layout) and (e) three-dimensional view (3D, explicit representation of inclusion relations, adjacent relations, data layout). Figure inspired by [54].

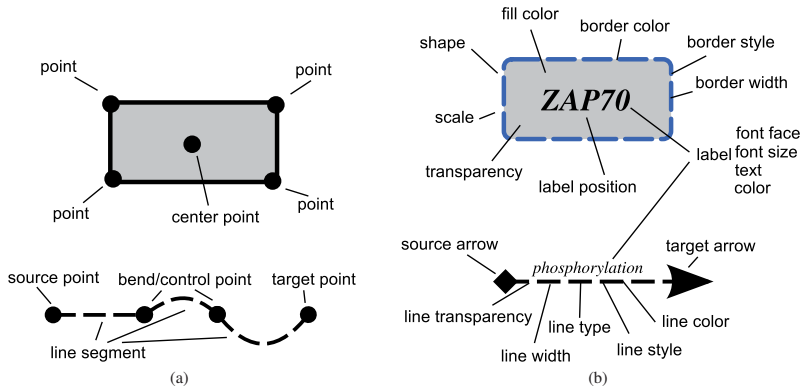


Figure 3.3.: Typical visual properties of nodes and edges. (a) Properties like *center point* defining the position by x- and y-coordinates; (b) properties like *node fill color* or *edge line width*. The figure is inspired by Droste et al. [50].

- C1. *Vertex-overlap reduction*: The number of overlaps of nodes should be minimized in order to get a clear and unambiguous graph representation.
- C2. *Edge-crossing reduction*: The number of crossings of edges should be minimized in order to get a clear and unambiguous graph representation.
- C3. *Area reduction*: The drawing area should be minimized. This criterion is mainly motivated by the limited screen real estate.
- C4. *Edge-length uniformness*: The length of edges should be relatively small. This results in graphs with minimal area. Furthermore, deviation in edge lengths should be small for more comprehensible graphs.
- C5. *Edge-bends reduction*: The total number of bends as the sum of all edges in the graph or per edge should be minimized. Edges with many bends are difficult to read.
- C6. *Angular resolution*: The angle between two straight edges incident to the same vertex should be maximized.
- C7. *Symmetry*: The more balanced (axial and rotational) a graph is, the better its readability. To improve the comprehension of graphs, equal things should be drawn the same way (e. g., isomorphic subgraphs).
- C8. *Vertex proximity*: In order to preserve structural proximity the graph should have a small geometric distance between adjacent vertices.
- C9. *Vertex inclusion*: Child vertices are within the area of their parent vertices. Child and parent vertex do not overlap. Thus, the hierarchical structure is encoded implicitly.

C10. *Vertex-even distributions*: The vertices should be evenly distributed among the layout area.

It is not possible to optimize all given criteria at the same time due to conflicts or even incompatible combinations. For instance, the reduction of bends or edge crossings (C2) contradicts the minimization of the drawing area (C3). Thus, the aesthetic criteria have to be prioritized according to the semantics, relevance and the application area. Figure 3.4 depicts different criteria prioritization of the same underlying graph that results in completely different layouts.

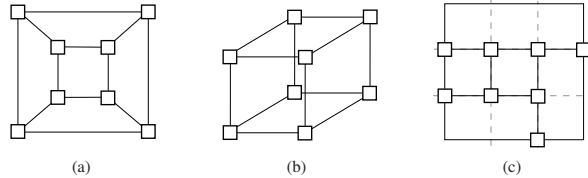


Figure 3.4.: Drawing conventions and rules optimized for a particular task for the same graph: (a) criteria C2 and C5, (b) criterion C4, and (c) orthogonal edges.

In addition, drawing aesthetics that are very effective when small graphs are involved tend to be less useful in the case of large graphs or even less so on structured graphs. For instance, the criterion of edge-crossing reduction (C2) is often suitable for small graphs but very hard to compute for large graphs with many relations. Also, there are different studies for ranking the importance of graph aesthetics [152].

3.2.2. Layout Strategies

For some tasks (i. e., same or identical set of criteria) widely used layout strategies have been established. For instance, the criterion of orthogonal edges results in an orthogonal layout.

In the following sections, first standard layout strategies are introduced. These are in particular hierarchical, orthogonal and force-directed methods. Thereafter, it is discussed how the basic concept of force-directed methods can be extended or refined for application to compound graphs.

Further popular approaches such as tree graph drawing, radial [207] or circular drawing [95, 179] are not discussed in this thesis. For more detailed information on standard layout methods, the interested reader is referred to Tamassia [191], Di Battista et al. [45] and Kaufmann and Wagner [94].

3.2.2.1. Layered Graph Drawing

Methods for layered graph drawing (also known as hierarchical graph drawing) place nodes on several parallel horizontal layers while, among other things, trying to minimize the number of crossings (C2) or the area of the layout (C3) as depicted in Figure 3.5b. They are mainly applied to directed graphs. The most widely used method for drawing these layered graphs was proposed by Sugiyama [190]. This method can be divided into four basic steps: (i) cycle removal, (ii) assignment of vertices to levels, (iii) edge crossing reduction, and (iv) assignment of coordinates to nodes.

3.2.2.2. Orthogonal Graph Drawing

Orthogonal layout methods place nodes on virtual grid points and edges are aligned with the grid using only horizontal and vertical polyline segments. See Figures 3.4c and 3.5c for examples. Orthogonal

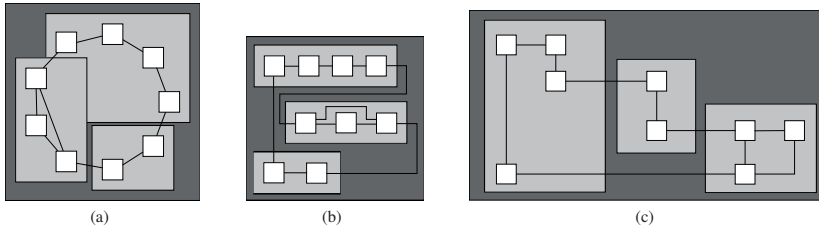


Figure 3.5.: One compound graph can be represented by many different drawings. Well-known layout methods are applied to the graph from Figure 3.2. (a) Circular layout with overlapping subgraphs, (b) hierarchical layout, and (c) orthogonal layout. Layouts are created using the software yEd [205, 211].

layout methods have been developed for undirected graphs. These methods try to optimize several drawing criteria for instance maximization of the angle between adjacent edges (C6), minimization of edge crossings (C2) and minimization of drawing area (C3). The methods can be divided into three phases according to the *topology-shape-metrics approach* [191]: (i) planarization (topology), (ii) normalization and orthogonalization (shape) and (iii) compaction (metric).

3.2.3. Methods on Physical Analogies

Methods on physical analogies interpret a graph as a physical system with forces (e. g., defined by repulsive particles or a spring model) between nodes. They try to minimize the overall energy of the system (equilibrium state) in order to get an adequate drawing. Methods on physical analogies are based on position and distances, not direction; that is, they can be applied to both directed and undirected graphs. Two physical analogies exist – *force-directed placement* [53] and *energy-based placement* [92]. The first approach minimizes the energy in the system by modifying the vertex positions (forces between vertices) iteratively whereas the second minimizes the energy in a direct manner [94].

3.2.3.1. Basic Concept and its Extensions

The method of Eades [53] is based on the *spring-embedder* model where *repulsive (spring) forces* between all pairs of vertices and *attractive forces* between adjacent vertices are defined. The spring-embedder initially complies several aesthetic criteria: uniform edge lengths (C4), as much symmetry as possible (C7), minimization of edge crossings (C2) and even distribution of vertices (C10).

The method of Fruchterman and Reingold [62] extends the spring-embedder by the concept of *temperature*. The global temperature controls the displacement of vertices, meaning that with decreasing temperature the distance which a vertex can move also decreases. This is similar to the more general concept of simulated annealing [40].

Subsequent works focused on extending the functionality and improving the quality of force-directed methods with respect to new heuristics such as detection of oscillations and rotations on the entire graph, an additional gravity force to keep unconnected elements from drifting apart, an improved concept of temperature (local instead of global) [59], directed edges [189], node overlap removal [118] or large graphs with thousands of vertices [202].

3.2.3.2. Extension of Force-Directed Methods to Hierarchical Graphs

Several force-directed methods for the application on compound graphs have been the subject of considerable research. Different strategies for how these hierarchical structures can be handled exist [139]. In general, existing methods for drawing undirected compound graphs based on force-directed algorithms can be divided into the following two classes: *local* and *global* methods.

Local methods, as described in Bertault and Miller [22], traverse the inclusion tree in level order and calculate the layout for each subgraph of the compound graph separately. The advantage of local methods is the independent calculation of each subgraph. Consequently, the forces have to be calculated in their local context, which reduces calculation costs. However, the relationships between nodes of different subgraphs are not taken into account which results in additional edge-edge crossing and long edges between different subgraphs.

Contrary, global approaches consider the complete compound graph during the layout computation. Consequently, all relationships between nodes of different hierarchical levels are considered. In Eades and Huang [55] an extension of the spring-embedder model where additional forces are implemented for different types of edges (internal, external) are presented. A further global method is introduced in Dogrusoz et al. [48]. There, various optimizations are applied to enhance the layout quality and to improve the runtime performance. The expensive calculation of the repulsive forces of all nodes is only performed between nodes of the same subgraph. The use of the skeleton technique, for example, as described in Dogrusoz et al. [48] where initially only the coarse structure of the graph (also referred to as skeleton graph) is laid out, leads to a further optimization. Finally, the layout of the complete graph based on this coarse structure is determined.

3.2.3.3. Advantages and Disadvantages

Force-directed methods have several advantages: They are very intuitive because the layout process is analogous to common objects in the physical world. Their basic procedure is easy to understand and implement. They often produce good-quality results for at least medium-sized graphs. They have an inherent flexibility and can be easily adapted and extended. Furthermore, they are able to work in an interactive manner; that is, the user can follow how the graph layout evolves over time.

Significant drawbacks of force-directed methods include a high runtime, typically $\mathcal{O}(n^3)$, with the cost of computation for all forces in $\mathcal{O}(n^2)$ and number of iterations in $\mathcal{O}(n)$. They produce a graph with minimal energy which can also be, of course, a local minimum. The problem of poor local minima becomes more important as the number of vertices of graphs increases. Further discussions about their advantages and disadvantages can be found in Di Battista et al. [45].

Global methods solve the disadvantages of local methods considering the relationships between nodes of all hierarchical levels. Their main disadvantage are the high computational costs through taking into account all nodes of the compound graph in each single layout step.

3.2.4. Domain-Specific Knowledge

Many models in systems biology have been visually represented using standard graph layout methods. These approaches only take into account the topological information given by the graph structure and do not fit very well with common representations of signal transduction networks [28].

In order to obtain an improved representation of the properties of the modeled system, domain-specific knowledge must be included in the layout process. In systems biology such contextual information includes *subcellular localization*, *functional annotation*, or the *direction* of a particular cellular process.

3.3. Visualization Principles

Visualization is the process of transforming data into a visual form; that is, transforming the abstract and symbolic into the geometric dimension. Thereby, the highly efficient human perception can be utilized to comprehend huge amounts of data. In Dwyer [52], two main purposes of this process are stated: *communication* and *investigation*.

Communication refers to the process of conveying knowledge about data to diverse communities. This practice is also described as *visual explanation* [198]. Visual explanation is of specific importance for successful work in interdisciplinary teams because this defines a common language for direct communication about the data, for example, computational models without the need for in-depth mathematical knowledge.

Investigation means the visual exploration of a dataset by applying various analysis and visualization techniques. This process refers to the area of *visual analytics*. Visual analytics combines visualization, data analysis and human factors. For that, computers handle huge datasets and generate visual representations; in addition, humans search for pattern and interpret the data. In this decision-making process human factors (e. g., powerful visual channel) play a key role [96]. In this sense, visual analytics supports the user in finding hidden relationships, identifying unexpected (structural) properties or characterizing issues with the data itself, for example, detecting outliers or anomalies.

Both purposes of visualization can be greatly supported by using *interactive animations* and *interactive elements* where the user can interact with the data in an immediate and creative way. However, it is challenging to provide visualizations with transparent interaction mechanisms in order to facilitate an effective transfer between the user and the investigated datasets [97]. By appropriate visualizations the user is enabled to analyze, interpret and translate the data into meaningful models and hypotheses, and finally might generate new insights.

The following sections provide an overview of visualization in systems biology. In this context, the research area of *information visualization* is discussed. This includes the visualization pipeline as a conceptual framework on how graphical representations are created in general, the research on cognitive principles and visual interfaces. In more detail the visualization of modular models is motivated and described. Finally, visualization tools dedicated to systems biology research are reviewed.

3.3.1. Information Visualization

An important research area is *information visualization*. It provides concepts and methods to facilitate computer-supported, interactive, graphical representations of abstract data [31] using the capabilities of the powerful human visual and cognitive system³. Thus, the visual perception has the ability to process many different pieces of complex and highly structured data at the same time. Information visualization facilitates the visual exploration, which is an important part of the analysis of large datasets. The graphical representation of these datasets is as compact as possible. As stated in Shneiderman [177], “Effective information visualizations enable users to make discoveries, decisions or explanations about patterns (e. g., correlations, clusters, gaps, or outliers), groups of items or individual items.” The design process in information visualization consists of different data transformation steps – from raw data to graphical data. These steps are outlined in the following section.

3.3.2. The Visualization Pipeline

The aim of the process of creating graphical representations is to present raw, mostly abstract data in different perspectives that are more used to humans. A conceptual framework [31], also known as the

³According to Ware [203], 40% of a human’s cortical activities are dedicated to processing visual signals.

visualization pipeline, is depicted in Figure 3.6. Thereby, the data has to undergo several step-wise transformations. At the beginning of the pipeline, raw data are used as input while at the end the visualization serves as a visual output. The pipeline⁴ consists of four main transformations [35]:

1. **Data Analysis**

Data are prepared for visualization (e. g., by interpolating missing values, or correcting erroneous measurements). Usually, this is computer-centered and provides little or no user interaction.

2. **Data Filtering**

Selection of data portions (focus data) to be visualized. Usually, this is user-centered.

3. **Data Mapping**

Focus data are mapped to geometric primitives (e. g., points, lines) and their properties (e. g., color, position, size) encode the underlying information. This is the most critical step for achieving expressiveness and effectiveness. Based on the same data, different data mapping functions yield in a high variety of resulting visual representation.

4. **Data Rendering**

Geometric data are transformed to visual data, for example, images or visual representations.

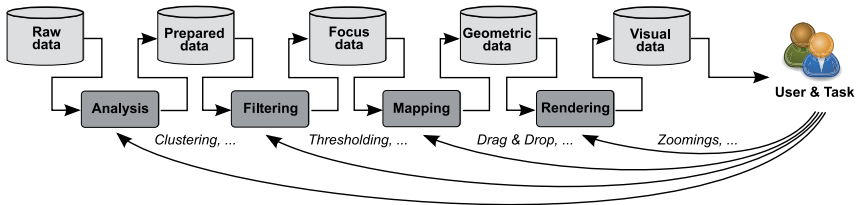


Figure 3.6.: The visualization pipeline describes the fundamental process of generating a graphical representation – actually a step-wise transformation from raw data to graphical data. Containers are data types; rounded rectangles are processes. User interactions can trigger changes at any step in the pipeline. Taking into account these interactions, the pipeline can be described as a feedback process. Adapted from dos Santos and Brodlie [49], see also Card et al. [31], Chi [35] and Janecek [88].

The steps ‘Data Analysis’ and ‘Data Filtering’ operate on the data and their values, whereas the operations of the steps ‘Data Mapping’ and ‘Data Rendering’ perform on the view [35]. In order to control the visual output, highly interactive environments allow the user to manipulate the visualization process, for example, modify model structure, set different filter methods, change mapping functions, and generate different views by interactive zooming. These interaction mechanisms feed back information and force modifications of the different steps (see Figure 3.6). In this way, the visualization can be tailored to the modeling task or individual preferences.

In practice, each research topic discussed in this thesis is closely related to the visualization pipeline and its specific steps in order to generate adequate visual output. For instance, the question of how complex structures of modular models can be explored and navigated is strongly correlated to the steps ‘Filtering’, ‘Mapping’ and ‘Rendering’, and the specific user interactions associated with them.

⁴Several flavors of the visualization pipeline exist, which differ according to the specific intended application.

3.3.3. Perceptual and Cognitive Principles

General principles of the human cognitive and perceptual system are very important in the area of (graph) visualization, particularly when dealing with complex datasets. Some commonly accepted research results are described in the following sections.

3.3.3.1. Preattentive Processing

Preattentive processing of visual information is performed automatically on the entire visual field, detecting basic features of objects in the display. Such basic features include color, closure, line ends, contrast, curvature and size. These simple features are extracted from the visual display in the preattentive system and later joined in the focused attention system into coherent objects. Preattentive processing is done quickly⁵, with little effort and in parallel without any attention being focused on the display [196].

When displaying information, it is often useful to be able to show things at a glance. Here, preattentive processing plays an important role. Examples of preattentive features⁶ are shown in Figure 3.7. Remarkably, the ‘connect’-feature that is commonly used in graphs is more dominant than, for instance, shape, color or size [144]. This is illustrated in Figure 3.7d.

3.3.3.2. Visual Cues

Working with large biochemical networks requires not only information visualization techniques, but also visual cues that were adapted from the human visual system. Different preattentive visual cues are color (hue, saturation and value), shape, texture, size (width and length), intensity and blink. Another phenomenon is that the combination of visual cues is more than sum of its parts. There are three data types that can be visualized⁷:

1. **categorical (nominal, discrete, qualitative)**,
(e. g., kinase, phosphatase, transferase, $\ominus > 12$ hues, $\oplus \leq 12$ hues),
2. **ordered (ordinal, discrete, qualitative)**,
(e. g., small, medium, big, \ominus hue, \oplus saturation/brightness), and
3. **continuous (quantitative)**,
(e. g., 2mm, 4mm, 6mm, \ominus rainbow color maps, \oplus interpolate between two colors).

Bertin [23] identified and classified a basic set of visual variables according to the scale of information that they are most appropriate for encoding (i.e., categorical, ordered, and continuous). Figure 3.8 shows a summary of Bertin’s classification, modified to include more recent research (for more detailed information see Janecek and Pu [89] or Qeli [153]). The most effective visual cue for presenting quantitative information is the property *position* [37]. For this reason, position is almost always used by visualization techniques to encode the relationships between the primary data values.

One of the problems of information visualization is its inability to formulate precise criteria for the effectiveness of graphical representations. Such a formulation is difficult since the effectiveness depends largely on conventions and the perceptual capabilities of the viewer. Figure 3.8 presents a ranking of encoding techniques according to three types of data to be displayed.

⁵Typically, tasks that can be performed on large multi-element displays in less than 200 to 250 milliseconds are considered as preattentive.

⁶Features are based on Gestalt principles [74] such as law of similarity, law of proximity or law of connectedness.

⁷ \ominus indicates negative examples, whereas \oplus lists positive examples.

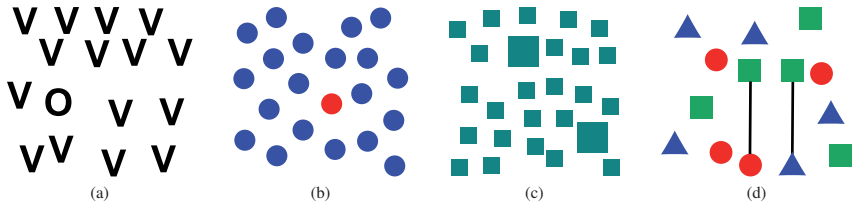


Figure 3.7.: Examples of different preattentive features – shape (curvature), color (hue) and size. (a) The letter ‘O’ can be detected preattentively because it has a different shape. (b) The red circle is easy to find because color is a preattentive feature. (c) The two bigger squares are recognized preattentively. (d) Connectedness is more dominant than shape, color and size.

Visual Cue \ Data Type	Shape	Texture	Color (Hue)	Color (Saturation)	Grayscale	Area	Length	Position
Categorical	★★	★★★	★★★	★★	★★	★	★★	★★★★
Ordered	☆	★★	★★	★★★★	★★★★	★	★★	★★★★
Continuous	☆	☆	★	★	★	★★	★★★★	★★★★

Figure 3.8.: The relative effectiveness of visual cues for representing three different data types – categorical, ordered, and continuous data. Data types with one white star are not applicable; data types with one black star are less effective for the visual cue and, therefore, not relevant for the encoding task. Data types with three black stars are the most effective. Diagram adapted from Janeczek and Pu [89].

3.3.3.3. Mental Map

When working with graphical representations, usually users build a mental model or *mental map* of the visual information they consume. The mental map refers to an internal cognitive structure representing the underlying understanding of the visual information by the user. The concept of the mental map was first described by Tolman [192] and later introduced to diagrams by Eades et al. [56]. In interactive tools where complex visualizations frequently change over time, it is not useful to recapitulate the unchanged information but rather concentrate on the changes between successive visualization states. This will lead to an immediate capturing of the modifications and finally to an improved understanding of the visualization over time. It is essential to keep the mental model consistent throughout the interactive visualization – otherwise confusion or misleading may occur.

A popular area where mental maps are intensively discussed is graph drawing. Interactive graph drawing changes the properties of graphs over time, for instance node positions. A consistent mental map allows the user to immediately understand the modifications between two graph states [115]. An example is shown in Figure 3.9. Of course, other visual graph properties can change over time as well, for example, size or color. In these cases the preservation of the mental map is also essential.

For preserving the mental map in a more formal way it is important to provide different constraints that can be added to the visualization. The following constraints and guidelines are relevant:

- Distort the graphical representation as low as possible.
- Geometric and spatial constraints such as *orthogonal ordering* (describes the alignment of elements), *proximity relations* (describes the distance between elements) or *topology* (depicts the regions in space) are essential [131].
- Use animated transition between visualization states, for example, nodes are moved to new positions. Thus, the user can recognize the changes.
- Landmarks should be used frequently.

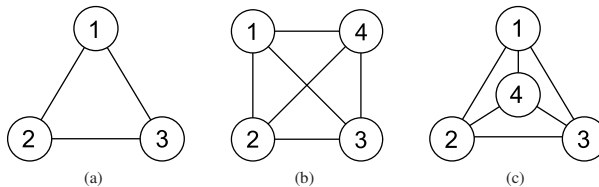


Figure 3.9.: Mental maps in graph drawing. (a) A simple graph drawing is shown using a force-directed layout. (b) When adding another node and running the layout algorithm again, the mental map is destroyed and leads to confusion. (c) In contrast, a consistent mental map supports the understanding of changes over time.

However, when the number of elements in the visualization and also the number or frequency of modifications increase, the problem of preserving the mental map becomes more critical. In this case, the user may be confused and misled. In the field of systems biology, preserving the mental map is crucial, but also a challenge.

3.3.4. Interactive Exploration

The support of interactivity is as important as the underlying visual representation of abstract data. The visual exploration can be used when the modeler has little or no knowledge about the data (e.g., the modeler is not familiar with a particular pathway), or the exploration goals are not clearly determined. Interactive exploration is a key element for managing the complexity of modular models. In particular, this involves the exploration of and interaction with the complex model structure. For instance, the modeler can interactively navigate in the model or zoom into different parts (e.g., modules) by focusing on a specific element. Thereby, each structural element can be used as an input target for controlling navigation and exploration tasks. Well-designed interactions preserve the mental map, help overcome

the difficulties of perception and comprehension, and facilitate reinspection by focusing on specific parts and switching of perspectives. Most of the interaction and exploration techniques can be improved by adding animated transitions.

3.3.4.1. Animated Transitions

Animation is necessary to create smooth transitions between different visual states. Hence, animated transitions preserve the perceptual continuity [159] and can be used as a visual cue to effectively express processes such as camera motions, zooming operations, changes of visual properties over time or alterations in graph layout. Animation is a very strong visual cue – physiologically more effective than color or shape [13]. It is also recognized as a preattentive feature [196, 203]. In general, animation can support the comprehension of complex visual systems [17].

A smooth transition from one visual state to the subsequent one is necessary in order to preserve the mental map [131], and thus reduce the cognitive effort and let the modeler focus on the changed entities. This also results in an improved understanding of changes in the modular model. In contrast, abrupt transitions would cause disorientation, confusion, misleading or annoyance of the modeler.

In the following, different transitions are summarized that can be supported by animations.

- *Navigation/exploration*: Visual transitions can be used to animate the navigation steps from node to node within a modular model. Hence, the user has the experience to zoom through the model while it is being explored. The zooming operation is achieved by (linear or non-linear) interpolation of the clipping rectangle of the view between the start and end node.
- *Time-dependent changes of visual properties*: For the interpretation of omics data (e. g., from time series) animated transitions are often used to depict the next condition after a predefined time interval [69]. This leads to a visual depiction of the overall dynamics of the model by animating several visual properties of the species nodes.
- *Graph layout*: Animations can guide the layout process by transitions of the node translations that allow the user to more easily follow the changes in layout. Some classes of layout algorithms (e. g., force-directed layout approaches) facilitate an intrinsic animation effect because the nodes are moved iteratively during the layout process.

Special properties of an animation are the *duration* and the *frequency*. In literature [17, 32, 33] usually a suitable time in the range of 0.3–1.0 seconds is proposed. Bartram et al. [12] stated that five frames per second are sufficient to produce a smooth transition. However, the appropriate duration of the transition depends on the application domain and the specific purpose. As stated in Cockburn et al. [39], sometimes “rich cues to maintain comprehension and orientation throughout the animation, demanding longer animated periods and more detailed representations throughout the transition” are needed.

Animated transition can be *uniform* and *non-uniform*. Uniform means for each time interval the same amount of changes is performed. In contrast, non-uniform animations can be divided into different patterns such as slow-in/slow-out, fast-in/slow-out or slow-in/fast-out. Typical uniform and non-uniform animation functions are depicted in Figure 3.10a. An optimized version⁸ of the general slow-in/slow-out animation function is shown in Figure 3.10b. There, the non-uniform animation function is approximated by splitting the function into two simpler subfunctions. This is important for performing animations in real-time.

⁸Optimized in terms of computational power.

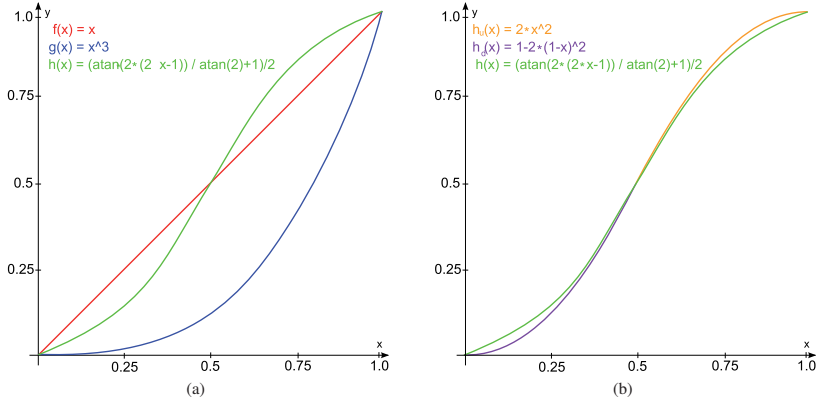


Figure 3.10.: Uniform, non-uniform and optimized functions for animated transitions. In (a) different uniform and non-uniform animation functions are depicted: standard uniform function (red), slow-in/fast-out (blue), and slow-in/slow-out (green). Additionally, in (b) an approximation of the given slow-in/slow-out function (green) is shown. Only the top (orange) and bottom (purple) parts of the function are used for animated transitions.

3.4. Visual Interface Schemes

Since modular models are highly complex, not all information can be displayed on the screen at the same time (e. g., display trade-offs such as size or resolution). In most cases, the displayed information is still too complex to be comprehensible to the user. Thus, only the essential information in a given context can be presented. Visual interfaces that contextualize information play an increasingly important role in how information is accessed, analyzed and understood. They distinguish between detailed information (focus) and overview information (context). Besides more general interface approaches described in Alves et al. [7], different specialized strategies exist that support the contextualization of focused information. Following the comprehensive review in Cockburn et al. [39], visual interfaces can be divided into *Overview+Detail* using a spatial separation, *zooming* using a temporal separation and *Focus+Context* using an integration of focus and context into a single view. In the following, these visual interfaces are introduced and described in more detail.

3.4.1. Overview+Detail Interfaces

Overview+Detail interfaces [150] are characterized by presenting the details together with an overview of the entire model. The information is separated in space – typically by multiple windows or views [15]. Thereby, focus and context information are shown simultaneously.

3.4.2. Zooming Interfaces

Zooming interfaces provide functionality in order to magnify (i. e., zoom into the focused information) or demagnify (i. e., zoom out for contextual information) the model. Thereby, transitions between different zoom states help to preserve the mental map of the user. Zooming is applied mostly in discrete steps, rarely in continuous, smooth animations. Zooming can be distinguished into two types – *geometric zooming* and *semantic zooming*. Using *geometric zooming* only the size of elements is altered, but the appearance of the elements remains unchanged. In contrast, applying semantic zooming the *Level Of Detail* (LOD) of elements (e. g., appearance, content or overall presence) is adjusted to the current zoom level. One important representative of zooming interfaces is the *Zoomable User Interface* (ZUI). In their simplest form ZUIs are interfaces with detail only. They allow zooming and panning [83]. ZUIs are based on the idea of an unlimited information space. Thereby, it is possible either to zoom in to or out from a single information object or the complete network or both, and to scale the information to be viewed in a very flexible way. ZUIs feature several methods known from information visualization such as semantic zooming, panning or smooth animations.

3.4.2.1. Space-Scale Diagrams

Space-scale diagrams [65] are the theoretical concept for the detailed description of ZUIs. The user has a fixed-sized viewing window that can be moved through the 3D space (see red-bordered rectangle in Figure 3.11a). In this zooming pyramid, a point in the drawing is transformed to a ray, a circular region is transformed to a cone, and a rectangular region becomes a pyramid, respectively. In this way, all possible sequences of Pan+Zoom can be achieved. When the user performs a zoom-out (i. e., move the viewing window on the z axis downwards) more elements can be represented. For a zoom-in, the viewing window has to be moved upwards, respectively. Moving the window on the x, y plane with constant view scale, a classical pan operation can be performed. As depicted in Figure 3.11b, different views are generated by Pan+Zoom operations of the viewing window.

3.4.2.2. Semantic Zooming

In contrast to geometric zooming, semantic zooming does not provide a scaled version of a visible object when zooming, but rather a different representation of the object. As depicted in Figure 3.12, an object can be completely changed by a zoom operation. Thus, semantic zooming defines varying information to be displayed based on the scale. The exact representation depends on the meaning in a specific application or domain; that is, semantic zooming is not generic.

Several reviews exist on the evaluation of ZUIs (e. g., Hornbæk et al. [83]). Most give controversial results on ZUIs, but they agreed that ZUIs can be effectively applied when the domain and the structure of the model data are considered.

3.4.3. Focus+Context Interfaces

The interface schemes described above provide focus⁹ with disjoint context, either spatial or temporal. In contrast, the *Focus+Context* interface integrates both, focus and context information, in a single visualization. The user is focusing on a local area of interest while an overview of important aspects of the context is still provided. In this way, elements can have different sizes and LOD (analogous to semantic zooming). A definition of Focus+Context is given in Card et al. [31]:

⁹A focus means a location in the structure of the visualized model that is of temporary importance for the user [171].

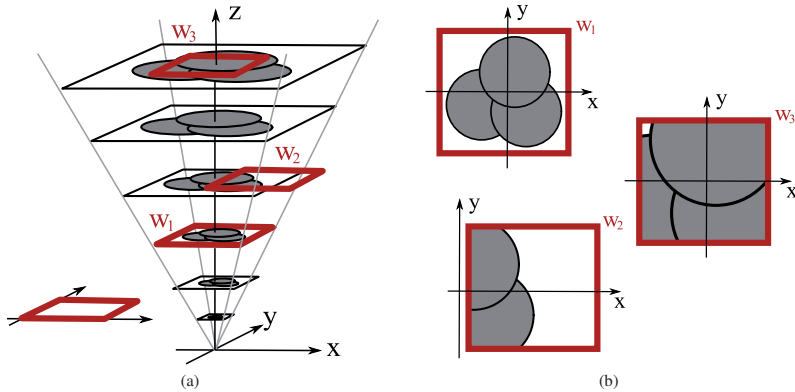


Figure 3.11.: Basic concept of space-scale diagrams. The viewing window with constant size is colored in red. (a) Zooming pyramid with different zooming levels. Only a zoom operation is necessary to transform the view from w_1 to w_3 ; to obtain w_2 only a pan operation must be performed; (b) different views generated by Pan+Zoom operations. Adapted from Furnas and Bederson [65].

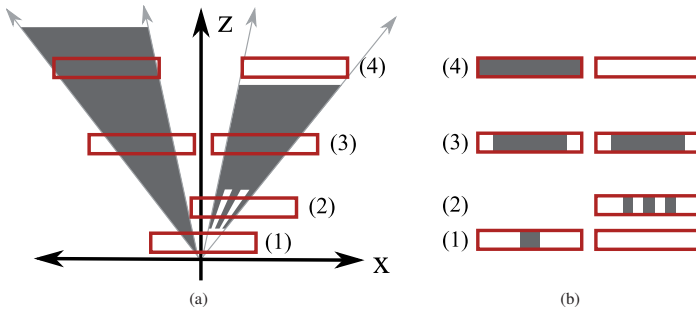


Figure 3.12.: The basic concept of semantic zooming applied to a space-scale diagram. The viewing window with constant size is colored red. (a) Zoom in a line, with standard geometric zooming (left), and with semantic zooming (right). (b) Different views are depicted: (1) too far, see nothing, (2) zoom in, see dashed line, (3) get closer, see solid line, and (4) get too close, see nothing. Adapted from Furnas and Bederson [65].

“Focus+context start from three premises: first, the user needs both overview (context) and detail information (focus) simultaneously. Second, information needed in the overview may be different from that needed in detail. Third, these two types of information can be combined within a single (dynamic) display, much as in human vision.”

Fisheye Views (FEV) [64] is the most popular Focus+Context technique¹⁰. According to Card et al. [31] and Carvalho et al. [34], FEV have the following main characteristics: (i) the modeling task requires both, detailed and overview information, (ii) information needed in overview and detail are different, and (iii) detailed and overview information are combined within a single visualization, analogous to the perception of the human vision system¹¹. Additionally, with FEV multiple foci can be supported. Robinson and Flores [160] emphasized that a Focus+Context technique is effective for understanding complex structures of biological data, for example, by supporting the user to maintain the orientation.

3.5. Networks and Modular Models

According to Chapter 2, Section 2.2.1, in systems biology researchers are faced with the acquisition of a tremendous amount of data. After acquisition, raw data must be compared, categorized, consolidated and summarized in order to provide comprehensible information about the underlying system. With increasing frequency large, interconnected and highly structured models that become hard to understand and navigate are the result of that processing step. Since models are growing in size and complexity it is evident that adequate visualization techniques are needed [69]. In the following, the visualization of networks and modular models is discussed.

3.5.1. Cellular Networks and their Visual Representation

In the context of systems biology, it is widely accepted to represent biochemical systems and their relational structure as *networks*. In order to analyze them, techniques from graph theory are applied. With that, the general graph in the form of a network is applied where nodes correspond to reactants like substances, co-substances, enzymes or signals, and directed edges (links) correspond to relations between them such as reaction, regulation, flux or transduction. The network can be added with the edges representing the direction of the reaction or undirected (edges without a direction) or mixed.

However, the semantics of nodes and edges depend on the application area and particular network type. Another possible constellation is a network with three different types of nodes. The nodes of the first type correspond to reactants in the classical sense as described above. The nodes of the second type represent events of functional regulation, chemical reactions or protein-protein interactions. They can have physical meaning or denote general associations. Nodes of the third type represent complex nodes such as complexes, functional units or other modular components.

Basic network types popular in systems biology research correspond to the important types of processes in cells: *metabolic networks*, *signaling networks* and *gene regulatory networks* (cf. Chapter 2, Section 2.1). An additional important network type is the *Protein-Protein-Interaction (PPI) network*. It is used to describe interactions of the level of proteins, for example, a binding between two proteins or a phosphorylation of the bound proteins.

In metabolic networks nodes represent chemicals (e. g., substrates, metabolites), in signal transduction and gene regulation nodes represent species such as genes, proteins (e. g., receptors, effectors, ligands or kinases). In metabolic networks edges represent biochemical reactions, in signal transduction and gene regulation edges often represent signal and information flow (e. g., through physical interactions such as binding, or activation for positive and inhibition for negative control). In PPI networks, proteins are represented by nodes and each interactions between a pair of proteins is denoted by an edge. Starting

¹⁰FEV based on the same idea as fisheye lenses in photography; that is, the objects in focus are enlarged while objects in the surrounding are reduced in size.

¹¹The attention of the user is naturally focused on objects in the foreground, but the user is still aware of the surrounding objects in the background.

from these basic network types, various subtypes are applied in research such as enzyme-enzyme relation, ligand-receptor relation, or gene expression interaction. Many of them describe binding and modification processes.

However, the strict division of cellular networks into the major types mentioned above becoming less important in future research because entities and related processes are tightly interconnected, for instance often the same molecule participates in signal transduction as well as in gene regulatory processes [80]. For visualizing cellular networks, different formalisms and graphical notations are used. Some important representatives are Petri nets [106], Molecular Interaction Maps [107], Process Diagram Notation [101] and more recently SBGN [114]. They represent networks using nodes and edges. Thereby, often groups of similar entities (e. g., kinase, phosphatase) are established.

So far, many *static representations* have been used [68, 107, 126, 137]. These schematic diagrams are usually of a high quality and reflect the diagrams often found in published books, but are expensive to create and update, and do not provide interactive exploration. This results in information overflow and a very limited view of the data. In contrast, *dynamic representations* facilitate interactive techniques such as multiple views, zooming or smooth animations. They solve most of the ‘static’ problems. However, they are frequently applied to simple, unstructured data.

Furthermore, heterogeneous data gathered by highly interdisciplinary working teams must be integrated and communicated in an appropriate and efficient way. For example, a working team integrates biological knowledge in the form of a complex model. This model has to be communicated to and finally explored by other teams. Here, visualization is even more appropriate because it defines a common language for direct communication and exchange, for example, between experimentalists and modelers. For instance, it can provide an understanding of complex models without the need for in-depth mathematical knowledge. Visualization also assists the interpretation of biological concepts and the solving of problems by presenting different ways and different view points. The visualization of complex cellular systems can assist the development of mental models that allow researchers to set up, access and analyze biochemical information in an efficient way and potentially in the end provide new insights.

3.5.2. Visualizing Modular Models

As already stated in Chapter 2, Section 2.3.3, modular models are characterized as having an inherent unitized and hierarchical structure. They describe model components at different levels of organization and abstraction. In order to visualize modular models of cellular systems and their rich structures, a compound graph as discussed in Section 3.1.1 can be utilized. The structure of the network and its associated information implies layout information (e. g., nesting relationships). This information can be used to efficiently visualize and analyze modular models and, thus, help to understand their properties. An example is depicted in Figure 5.1.

Regarding the visualization of modular models two aspects are important: (i) how to handle and characterize different views, and (ii) how to visually explore and navigate in modular models. Both aspects are discussed in the following sections.

3.5.2.1. Views of Modular Models

Modular models and their visualization are based on hierarchical graphs which can be used to visually describe adjacent and inclusion relationships. This graphical model can be characterized by different views (see Figure 3.13). For instance, abstract or specialized views can visually describe the model – either by representing different model parts, or different visual property sets mapped on the respective model (cf. ‘Mapping’ step in Section 3.3.2). Views are the final result of the visualization pipeline and

are directly presented to the modeler. Some of these views relevant for the thesis are discussed in the following.

View Level In graph theory the hierarchical level L_T is an important characteristic (cf. Section 3.1.2) but also its visual counterpart – the *view level*. The view level indicates the current hierarchical level of a particular module (node) that is being visualized. By definition, the module representing the model (the root node) is the first view level (top view level). In most cases this is the most abstract view level. Furthermore, detailed view levels are strongly associated with hierarchical levels. For an example, see Figure 3.13a. According to nodes, edges can also have a view level which is defined by the minimum of the view levels of its end nodes.

Global and local view A *global view* is a high-level representation of the entire model and provides information at a glance about the network. Hence, it is a view with the lowest level of detail. ZUIs naturally support global views by zooming out to the overall context (see Figure 3.13a). In contrast, a *local view* is used to show a subset of the model (i.e., concentrating on detailed information). Thereby, the context is often lost (see, for example, Figure 3.13d).

Multiple views and integrated view In many systems biology applications, *multiple views* exist to show the same model with different LOD, for example, Overview+Detail interfaces applying a context view and a detailed view. Multiple views can also be used to show only a portion of the model in each individual view. Then multiple detailed views are used. An example is discussed in Section 3.6.3. Multiple views need a lot of space on the display and must be linked by appropriate interaction techniques in order to relate the data in the separate views.

Multiple views can be aggregated to a single view – an *integrated view* [97]. Integrated views need less space compared to multiple views and improve the process of relating the data portions. By applying an integrated view the cognitive load can be reduced. However, the use of multiple and integrated view depends mainly on the type and structure of the model data.

Standard view *Standard views* are predefined sets of parameters or rules, which produce similar visualization results in the same semantic situations. They are useful for building a mental map and reducing the cognitive load of the user. The global view is the most prominent standard view. Other forms of standard views are discussed in Chapter 6, Section 6.2.5.

3.5.2.2. Strategies for Navigation and Exploration

Due to the increasing amount of data in the form of graph structures, techniques for the interactive interaction with the structure become more and more important [201]. Those representations effectively permit users to navigate and orient themselves in a biological model. Especially, the proper understanding of global and local structures is an essential aspect in many analysis tasks. In particular, this applies to modular models whose structure forms an inherent topological property. This property cannot only be used as a graphical model for representation purposes, but also for navigation and exploration. In the latter case, the hierarchical structure and explicit decomposition of the model facilitates rich cues for exploring the model and its subunits. However in the past, navigation and exploration has been facilitated through standard operations like panning, scaling, or scrolling. All of these techniques were originally developed for flat or quasi-flat network models and do not consider the special hierarchical structure.

A suitable solution is the ZUI that is dedicated to the navigation in hierarchically structured models. For that purpose it provides flexible zooming techniques, in particular semantic zooming (cf. Section 3.4.2).

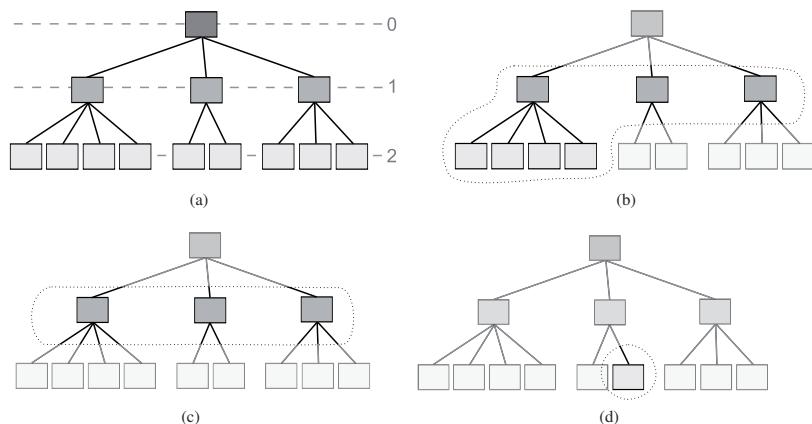


Figure 3.13.: Different views of the same modular model taken from Figure 3.2b. Nodes and their inclusion relations are represented as a tree structure. Nodes within the dotted area are visible in the specific view; transparent nodes (outside the dotted area) are not visible. (a) Global view where all nodes are visible. The hierarchical level L_T is annotated, (b)–(d) local views whereas (b) is a more detailed view of (c), and (d) is one of the most detailed views possible in the model.

However, the navigation and spatial orientation is difficult to control, for example, when applying zooming operations [65]. Thus, it is essential that operations such as navigation, exploration or zooming guide the user by reflecting the topology and the properties of the modular model. This assumes to know the current position of an element in the global context of the hierarchy, and change the focus accordingly. While navigating, the network is displayed by a sequence of different views that are determined by the *navigation path* or *focus path*. For instance, a seamless zooming operation leads to frequent changes in context. In extreme cases, each step is a completely new view [158]. Therefore, in order to preserve the mental map it is important to allow only slight changes from one to the subsequent view.

Furthermore, modular models provide the framework for viewing the data at various levels of detail. In combination with ZUIs, users can align the appropriate level of visual abstraction with the current modeling task. For visual abstraction the user can zoom out; for visual specialization the user can zoom in.

3.6. Tools for Visualization

There is a huge variety of successful visualization tools¹² – however, there are too many to be listed here. Instead, this section summarizes the most notable tools that have been developed for network visualization and exploration of complex datasets in systems biology research. Thereby, the focus is on tools which support either modular models, ZUIs, or both. First, relevant tools are divided into two main

¹²A comprehensive overview of visualization tools in systems biology can be found in recent literature [69, 128, 145, 170, 188].

categories – ZUI toolkits and visualization tools for systems biology – and then described in detail. Thereafter, the graphical user interface of PROMOT is introduced.

3.6.1. ZUI Toolkits

PAD++ [19] and its predecessor PAD [146] were, in fact, the first developments of a ZUI in the form of a general software framework. PAD implements the novel concept of *semantic zooming*. A further toolkit is JAZZ [20]. It is based on the main ideas of PAD++ and complements it with a 2D scene graph¹³ that clearly eases the development of complex ZUIs. PICCOLO [18] and its community-based approach PICCOLO2D [148] are based on the JAZZ toolkit; both use scene graphs as their basic structure. PICCOLO2D is a multi-purpose and very generic toolkit, and allows rich forms of semantic zooming to be implemented. Very similar to the basic concepts of PICCOLO2D is ZVTM [149]. These research projects were the fundamental basis for several applications using ZUIs like SHRIMP [185], CYTOSCAPE [174] or GUESS [1].

From an engineering point of view (e.g., software engineering), approaches such as Koth and Minas [108], Jacobs and Musial [87] or more recently Frisch et al. [60] use ZUIs including semantic zooming and Focus+Context techniques to navigate in abstract structures such as UML diagrams depicting packages and class structures. A similar approach is implemented in SHRIMP [185] where a ZUI for visualizing and explore hierarchical networks such as ontologies is implemented.

3.6.2. Visualization Tools for Systems Biology

Many tools designed for the visualization of biological data exist. A review and evaluation of general-purpose visualization tools for systems biology can be found (e.g., in Saraiya et al. [170] or Cline et al. [38]). Some popular software tools are CELLDISIGNER [63], CYTOSCAPE [174], EDINBURGH PATHWAY EDITOR [182], GSCOPE [195], OMIX [50], ONDEX [112], PATIKA [43], VANTED [91], or VISANT [85]. All of them can be used for visualizing biological data as graphical networks and facilitate visual setup and editing of models. Some are designed for specific pathways only, while others support general biological networks. In Table 3.1, features of these tools are compared with a focus on visualization methods relevant for this thesis. In the following, the tools mentioned above are described in more detail.

CellDesigner CELLDISIGNER is designed for drawing biochemical and gene-regulatory networks. It extensively supports quasi-standards such as SBML or SBGN [114]. By using some interfaces that integrate tools like COPASI [82], SBML ODE SOLVER [121] or SBW [21], CELLDISIGNER, is capable to perform simulation tasks on the defined networks.

Cytoscape CYTOSCAPE is a general-purpose visualization tool for complex networks and also allows the drawing and manipulation of cellular networks. Different attribute data can be overlaid with the network. For that, CYTOSCAPE provides a powerful mechanism for mapping from arbitrary data attributes to visual properties. Different standard layouts are supported. CYTOSCAPE provides an interface for plug-in development.

Edinburgh Pathway Editor This tool is dedicated for setup drawing and manipulation of biological networks such as signaling or metabolic pathways. The editor supports several notations including SBGN and CYTOSCAPE notations and is also capable of describing a logical formalism.

¹³Scene graphs provide a framework for the development of 2D structured graphics, in particular ZUIs.

GSCOPE GSCOPE facilitates the visualization of biological pathways. It is specialized in statistically analyzing microarray data and shows the results in a network context. Unfortunately, GSCOPE is not accessible on the internet anymore. However, it has interesting features including a fisheye viewer for Focus+Context visualization and a clipped viewer for local link structures (up-down cascades). GSCOPE also provides subgraphics such as heatmaps and line charts, for example, for the depicting of time-series of microarray data.

Omixon OMIX is an editor for drawing biochemical networks. Thereby, visualizations of network components and also their animation can be defined in a flexible manner using the scripting language OVL. Additionally, OMIX allows a 3D visualization of networks on different levels. A semi-automatic layout tool provides book-style drawings of metabolic pathways. Plug-ins for additional functionality can be integrated via an advanced interface.

Onindex The focus of the software ONDEX is on integration and visualization of diverse datasets from biology. For that, many formats and access methods for databases, and the composition of workflows are provided. Network graphs are used to integrate, visualize and analyze the integrated data. Onindex also supports standard layout methods. A plug-in mechanism is also provided.

Patika PATIKA is a pathway visualization and analysis framework. It consists of a comprehensive database and different tools (e.g., editors for the manipulation of pathways). PATIKA provides a sophisticated layout for hierarchically organized networks. This layout algorithm is mainly intended for signal transduction.

Vanted VANTED provides a graph-based editor capable of integrating diverse data gathered from research in systems biology. This data (time series, different environmental conditions) can be statistically analyzed in the context of the graph depicting the biological processes. Subgraphics can be used to depict multi-dimensional data. For analysis purposes, VANTED provides several layout methods. It also supports quasi-standard formats such as SBML and SBGN.

VisANT VISANT is a tool specifically built for integrative visual data-mining of biological pathways and networks. Different analysis methods are provided like comparison of two datasets or search for functional modules. Results of the various analyses and also integrated data (e.g., gene expression data) are used to highlight portions of the network accordingly. In order to illustrate functional modules, VISANT uses metagraphs to structure the network.

As seen in the previous list and in Table 3.1, the tools differ in their visualization features which is motivated in their different purposes. Mostly all of them do not provide a modular approach for model setup and do not deal with hierarchical structures, for example, navigating in hierarchical networks. They rely on flat or quasi-flat network models, i.e., the underlying model is defined in a flat manner. Sometimes the modeler can arrange elements hierarchically in the diagram but these cues are not used to support structured navigation in the diagram. Often there is one single flat view for all elements in the model at a time.

As depicted in Table 3.1, Pan+Zoom operations, for example, magnify and move the graphical network is a widely used technique. There, the zoom operation means standard geometric zooming and is often limited to discrete steps [158]. Seamless zooming in the sense of full flexibility in scaling and zooming (including semantic zooming and animations) is rarely used. Moreover, the used zooming techniques are limited to flat networks and do not consider the structural properties of the model. One of the tools, GSCOPE, provides Focus+Context techniques to zoom into details while keeping the less detailed context intact. This is facilitated by a fisheye view projected on a parabolic surface. However, no hierarchical network is supported.

<i>Feature/Tool</i>	<i>CD</i>	<i>CY</i>	<i>EP</i>	<i>GS</i>	<i>OM</i>	<i>ON</i>	<i>PA</i>	<i>PR</i>	<i>VA</i>	<i>VT</i>
Versioning & Availability										
Version	4.2	2.8.2	2.0	2.0	1.4.24	0.4	2.1	0.8.4	2.0	3.91
License	©	LGPL	GPL	©	©	GPL	GPL	GPL	GPL	©
Graphical User Interface										
Text Editor	-	-	-	-	-	-	-	✓	-	-
Diagrammatic	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Information Visualization										
Pan+Zoom	✓	✓	✓	✓	✓	✓	✓	(✓)	✓	✓
Focus+Context	-	-	-	✓	-	-	-	(✓)	-	-
Seamless Zooming	-	-	-	-	✓	✓	-	(✓)	✓	-
Semantic Zooming	-	-	-	-	-	-	-	(✓)	-	-
Levels Of Detail	✓	-	-	-	✓	✓	-	(✓)	-	-
Overview+Detail	✓	✓	✓	✓	✓	✓	-	(✓)	-	-
Integrated View	-	-	-	-	-	-	-	(✓)	-	-
Modularity										
Hierarchy	-	✓	-	-	-	Metagraph	✓	✓	ClusterID	Metagraph
Nested Graphs	-	-	Parenting	-	-	-	✓	(✓)	-	Metagraph
Layout and Styles										
Visual Styles	SBGN	✓	✓	-	✓	✓	-	(✓)	✓	-
Graph Layout	✓	✓	-	✓	✓	✓	✓	(✓)	✓	✓

Table 3.1.: A comparison of visualization features of tools for systems biology. (CD) CELLDESIGNER, (CY) CYTOSCAPE, (EP) EDINBURGH PATHWAY EDITOR, (GS) GSCOPE, (OM) OMIX, (ON) ONDEX, (PA) PATIKA, (PR) PROMOT, (VA) VANTED, (VT) VISANT. ‘✓’ – feature is available; ‘-’ – feature is not available; ‘(✓)’ – feature is provided in VISUAL EXPLORER; © – software is available under non-commercial license; LGPL, GPL – software is available under the given open-source license. Add-ons/plugin-ins are not considered.

In most tools visual mapping is restricted to some visual properties (e.g., color, line thickness) and not very flexible. Often, the visual mapping is hard-coded and cannot be easily changed by the user. Two exceptions are CYTOSCAPE and OMIX whereas OMIX requires programming skills for defining customized representations. The majority of the tools provide a variety of automatic layouts which results in different options to lay out cellular networks. Some of them offer specialized layout methods. For instance, in PATIKA layout methods are applied to hierarchical networks.

Although these tools provide a large number of methods for network visualization, the majority do not support hierarchical decomposable diagrams which is a prerequisite for visualizing modular models. Instead, they provide support for flat views of the model, sometimes in combination with overviews, trees that represent a view of the structure, or navigation scrollbars.

3.6.3. Visualization with ProMoT

As already mentioned in Chapter 2, Section 2.4.2, modeling in ProMoT is supported by two different editor types – a text editor for editing of MDL syntax, and a graphical editor, called VISUAL EDITOR, for editing the model in the network context. VISUAL EDITOR facilitates the graphical setup and manipulation of a modular model in a strongly structured manner. It uses a compound graph with ports (cf. Section 3.1.2) where nodes represent elementary modules, composite modules and terminals, and edges represent links. Submodules are always in the bounds of their parent module. Terminals are located close to the border of the module as depicted in Figure 3.14b.

VISUAL EDITOR represents each module in a separate window. This simplifies the editing of small and local changes because only a few windows (with modules) need to be opened. However, it is difficult to perform more complex editing operations or to show the function of a set of modules in relation to the overall structure. In this case, many different windows need to be opened. In the VISUAL EDITOR, this results, on the one hand, in higher visual complexity because of multiple, decoupled windows and on the other, in a higher cognitive load of the user to relate all pieces.

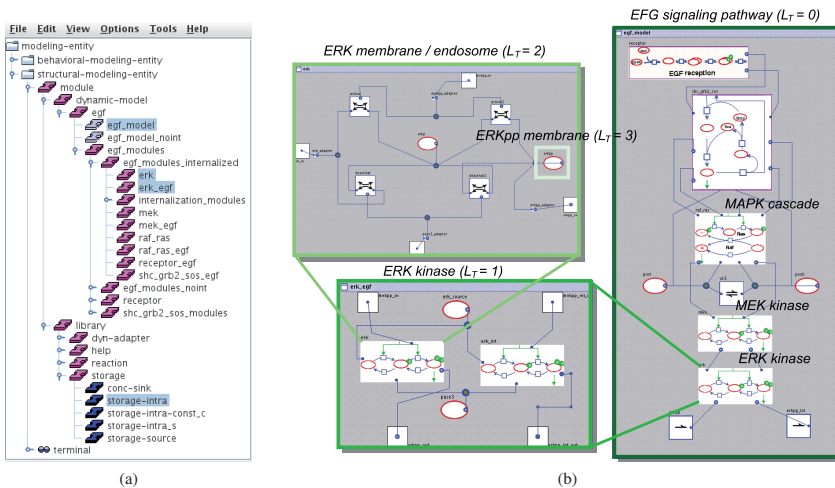


Figure 3.14.: The modular EGF model and its visualization in ProMoT. (a) PROMOT CLASS BROWSER with loaded library. (b) Highlighted modules of (a) are opened as disjunct views in VISUAL EDITOR. Composite modules (e. g., MAPK cascade) are decorated with icons depicting a rough sketch of the interior.

3.7. Conclusions

Graphs are an important concept for describing biological entities and their complex relationships. They facilitate the integration of various data that are produced in different groups and formats in an intuitive

way. Furthermore, graphs support the process of creating mental models of the system which leads to an improved understanding of the intrinsic data structure and solving the investigated problem. In particular, graphs with a carefully chosen layout and visual properties depicting signaling processes help users read and comprehend such complex systems.

For the purpose of visualizing complex relational data structures the graph drawing domain has a focus on optimized layouts for large and static graph representations. However, the domain of information visualization provides functionality to interactively visualize and explore such relational structures. This includes topics on interactivity, multiple views, different levels of detail, and visual appearance of data elements.

Visualization is a well-accepted method for analysis and interpretation, and translates the data into meaningful models and hypotheses. However, the potential of information visualization methods has so far not been sufficiently exploited by the systems biology community. With the recent transfer of systems biology to a data-driven discipline, the visualization of massive data is becoming more prominent. Recent work on visualizing data in systems biology has been limited to representing mostly flat data structures, but only little effort has been devoted to visualizing hierarchical data structures. Although several successful tools for network and data exploration in systems biology such as CYTOSCAPE, GSCOPE or OMIX exist, the majority lack in dealing with hierarchical structures or even modular models.

As mentioned in the previous section, VISUAL EDITOR of PROMOT provides disjunct submodels and its associated views. This introduces problems such as no visual context, occlusions of different windows and exactly only one detailed view. For the editing in VISUAL EDITOR only a single module description is used. In summary, there is a lack of a recursive or ‘deep’ view of modular models that depicts the various relations among modules and between different hierarchical levels. This can be implemented by an integrated view where the complete model can be visualized by generating different perspectives on the data and by utilizing Focus+Context techniques. These considerations led to the development of the VISUAL EXPLORER.

In the next chapters the actual work and results (i. e., VISUAL EXPLORER and its methods) are presented. It is shown how the combination of two different domains, graph drawing and information visualization, can lead to synergistic effects and can provide strategies to solve problems dealing with complex data structures. This is exemplified by application to real modeling projects at the end of each chapter.

3. Graphs and Visualization in Systems Biology – Fundamentals

Data is not information. Data must be presented in a usable form before it becomes information, and the choice of representation affects the usability. The representation system, and thus the graphical object model, plays a central role in converting data into information.

Thomas R. G. Green and M. Petre [76]

CHAPTER 4

Visual Representation of Dynamic and Logical Modular Models

As already discussed in the previous chapter, homogeneous views can help modelers become familiarized with the graphical representation of modeling entities and the biological processes behind it. This is in particular true for modular models where the representation of the structure is essential for understanding the model. Homogeneous views need some degree of formalization, in order to be ‘homogeneous’. This can be achieved, for example, by standard views. Another essential approach is to use formalized representations for modeling entities and the model structure.

This chapter describes the specification of two visual representations. The first representation visually encodes dynamic models; the second representation depicts logical models. The focus of both representations is on signal transduction systems used within PROMOT.

First the research problem is discussed. Then before introducing the specific modeling entities and their graphical representations, it is described how the models can be visualized in a single integrated view.

4.1. Research Problem

In textbooks, biological network models are usually represented as informal graphical representations. Often this results in ambiguous knowledge or long texts describing the depicted biological processes. In the past, in order to encourage the formal graphical representation for dynamic and logical models, several formalisms and notations have been presented. Important examples are Molecular Interaction Maps (MIM), the Process Diagram Notation or the notation used in the EDINBURGH PATHWAY EDITOR [182] (cf. Chapter 3, Section 3.5.1). Many of them provided a powerful and rich visual representation and also defined grammar. However, the focus of these languages is not on modular structures, and they are not capable of the full expressiveness of models defined in MDL (e.g., terminals).

In the following, some requirements for visual representations in PROMOT are presented:

- *Comply with MDL:* According to Chapter 3, Section 3.6.3, a model in PROMOT is defined by the modeling language MDL. Modeling entities such as modules, terminals and links are given in MDL syntax. Hence, a visual representation should be defined on the basis of components provided by MDL.
- *Consider modular structure:* Since the aim is the visualization of modular models, also typical modeling concepts such as composition (e.g., composite modules), abstraction (e.g., hierarchy),

4. Visual Representation of Dynamic and Logical Modular Models

and separation (e.g., flexible number of in- and outputs represented by terminals) must be considered. In this sense, a new visual representation, of course, must reflect the inherent structure including modules and hierarchy.

- *Adaptation and application to CellNetAnalyzer:* For logical modeling an additional research motivation and requirement is the compliance with CELLNETANALYZER. Such a formalized visual representation should be aligned closely to the requirements of CELLNETANALYZER since both tools should be tightly integrated, aiming at an efficient workflow.

None of the mentioned graphical notations can cover all requirements. Hence, there was a need for the development of formalized visual representations for dynamic and logical models in PROMOT. For this reason two specific visual representations were developed. The challenge was that such representations should be both, biologically intuitive using formal symbols, and suitable for modular models using an engineering approach. These visual representations were developed in close cooperation with modelers which have background knowledge in molecular biology. Shortly thereafter, a new notation, SBGN, was developed by the international community. The motivation for the development of SBGN was similar to the visual representations developed in this thesis. Today, SBGN is strongly supported and frequently used in the systems biology community and known as the quasi-standard notation.

In the following, first, it is described how the different model parts defined in MDL are composed to a single integrated view. Then the two specific visual representations for dynamic and logical models are presented and their main modeling entities are described in detail.

4.2. Towards a Single Integrated View

On first glance, the consideration of the modular structure seems to be straightforward but it reveals a challenging task. This is also highlighted by two recent publications [57, 77]. Many recent tools visualize their model parts in separate, often disjunct views. Multiple views in different windows might have a few advantages in the editing process, for example, by concentrating only on a model part of interest. However, for the exploration and analysis of the entire model this is cumbersome. As already mentioned in Chapter 3, Section 3.6.3, it is difficult to associate and relate elements from different model parts and views. Moreover, using different windows hosting these views can cause several problems such as occluded information by overlapping windows, significant time for managing such windows or disruptive visual context when switching between them. Hence, instead of a collection of disjunct views, an integrated view of a modular model should be generated; that is, the structured model should be represented in a single visualization. The procedure is described in the following.

For a single integrated view all model parts (e.g., modules, terminals, links) are considered. They are traversed in hierarchical order and aggregated to a multilevel structure. Therefore, a terminal cluster graph G_{TCL} (cf. Chapter 3, Section 3.1.2) is used. The resulting inclusion relations E_T are mapped to the nested hierarchy of G_{TCL} . Thereby, a submodule is added as subgraph G_S to the graph G_{TCL} . All nodes of the inclusion tree are represented as explicit nodes; that is, inclusion relations are part of the graph G_{TCL} (cf. Chapter 3, Figure 3.2d). Hence, the inherent modular structure is preserved. A terminal is added as a port p which is an interface of G_S to the exterior.

In most cases, model parts in separate views are visualized with the same scale factor (e.g., $s = 1.0$). Hence, it is possible that submodules are displayed bigger in size than their related parent modules. An example is given in Figure 4.1a. In order to prevent this, each subelement is scaled down by a default scale factor s_d before it is inserted into the parent element within the graph G_{TCL} . In this way, it is assured that each subelement is within the bounds of its parent element. As depicted in Figure 4.1b,

the subelement is then located on a certain hierarchical level L_T . The default scale factor of a parent-child relation is $s_d = 0.2$. Hence, the overall scale factor s of an element is computed by the following equation:

$$s = s_d^{L_T}. \quad (4.1)$$

For example, the overall scale factor of an elementary module m located on the third hierarchical level with $L_T = 2$ and $s_d = 0.2$ is $s_m = 0.2^2 = 0.04$. In other words, the size (length or width) of the elementary module m in the network is 25 times smaller than the size of the module on the first hierarchical level ($L_T = 0$), and 125 times smaller in respect to the area on the screen. As described later, the ‘scaled’ geometry has, of course, an impact on visualization, navigation and layout. For instance, elements in very deep hierarchies are nearly invisible because of their small size on the screen (in most cases for $L_T > 3$).

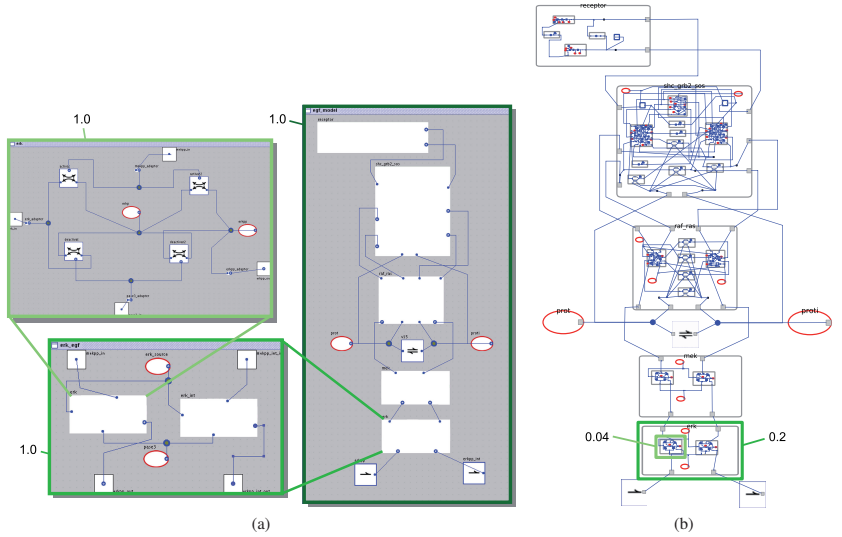


Figure 4.1.: Comparison of the visualization of the modular EGF model and its hierarchical structure in (a) separate, disjunct views and (b) an integrated view. Same color around modules in both subfigures corresponds to the same hierarchical level. Additionally, the global scale factors for certain modules are given (e.g., $s = 1.0$, $s = 0.2$, or $s = 0.04$). Note, in (a) all modules have the same global scale factor ($s = 1.0$). A magnified version of (b) is given in Figure 4.13.

From that single integrated model, according to the visualization pipeline, different views can be generated by drawing the graph G using its topological and visual properties. It is obvious that such a compact, single view of the modular model needs appropriate mechanisms for navigation and explo-

4. Visual Representation of Dynamic and Logical Modular Models

ration, for instance, to visually analyze elements in deep hierarchies. This is discussed in more detail in Chapter 5.

Having such an integrated view and appropriate exploration schemes it is possible to define high-level views and if necessary refer always to detailed processes of the individual modules [30]. In this way, the modeler is able to relate effects across different levels of scale and abstractions, for instance, small changes in a molecular interaction have effects at physiological levels [170].

In the following, the general procedure of visual transformation of dynamic and logical models is given in more detail. The entire process of the visual setup of models using the logical modeling approach is described in detail in Saez-Rodriguez et al. [165].

4.3. Entities for Dynamic Modeling and their Visualization

Dynamic models are characterized by a kinetic description of the underlying processes. This very detailed knowledge needs to be visually represented in a systematic manner. In this section, modeling entities for dynamic modeling and their visual representation are given. Thereby, the focus is on describing and depicting signaling systems. First the basic elements and their visualizations are described. Then the visualization of composite modules are discussed. Finally, in order to illustrate their usage an example with typical components is introduced. For a comprehensive overview, a ‘symbol reference card’ comprising relevant elements can be found in Appendix B, Figure B.1.

4.3.1. Basic Elements

For the convenient visual setup of dynamic models, different modeling entities are essential. They are provided by the standard library for dynamic modeling. The basic elements are *storage*, *reaction*, *help* and *adapter*. All of them are elementary modules and are derived from the class `module`. Additionally, different types of *links* are provided.

Storage Storages are used to model, for instance, proteins, genes, or species with constant concentration (e. g., ATP pool or constant enzymes). Specialized storages are, for example, `conc-sink` describing a sink of a signal (signal output) or `storage-intra-const` encoding an intra-cellular storage with constant concentration of a signal.

Reaction Reaction elements represent biochemical reactions including catalytic reactions like different enzymatic reactions (e. g., Michaelis-Menten), formal reactions like mass-action kinetics, or reactions for some special cases. Furthermore, reactions can be divided into reversible and irreversible reactions.

Help A help element represents either a `constant-rate` or a `sum`. A `constant-rate` element is used, for instance, for a constitutive protein synthesis. `Sum` elements are needed to add or split signals, for example, the product of two signals with different effects (one positive and one negative).

Adapter Adapters allow elements to connect to terminals of the module. For instance, this is useful when a reaction is modeled across different hierarchical levels.

Link A link connects different terminals that are owned by elementary or composite modules. Hence, a particular edge in a link is the connection between two terminals either of the same module or different modules, or a terminal and a link node. Terminals are needed as interfaces to the exterior.

In the next sections, important basic modeling elements and their particular visualizations are introduced.

4.3.1.1. Storages

Storages are represented by ellipses with a red border (Figure 4.2a). They can be visually specialized using bitmap icons, for example, a phosphorylated protein as shown in Figure 4.2b. Storages cannot be connected directly; they must be linked by a reaction element.

4.3.1.2. Reactions

Reactions are encoded by rectangles with a blue border (Figure 4.2c). Similar to storages their visual representation can be specialized by using bitmap icons (Figure 4.2d). A reaction element can connect two or more storages depending on the number of its terminals.

4.3.1.3. Links

Links are defined and visually represented by edges and link nodes (Figure 4.3). A link connects two or more terminals (owned by submodules). In the latter case, link nodes with adjacent edges within the same hierarchy are used. Link nodes are used to model the sum of the signaling information to different storages (information can also be split up). By that, link nodes reduce the visual complexity by aggregating all incoming and outgoing links into a single node. The direction of a link (e. g., direction of signals) is not indicated by an arrow head. Instead, the mass or signal flow is defined by the inner stoichiometry and kinetic information of the reaction, and can be visually indicated using a specialized bitmap icon (Figure 4.2d).

As already mentioned, links connect terminals owned by modules (e. g., storage or reaction elements). All terminals have to be connected; open terminals result in an invalid model. Terminals are always placed at the inner border of the module. They are represented by squares with a dark-gray border and light-gray background (Figure 4.3).

For each link, a color can be defined that is applied to the corresponding edges and link nodes. In MDL models blue, green, red and yellow links are used (Figure 4.3). Blue links indicate a concentration and reaction flux (bi-directional). Green links are used to denote signaling relationships (uni-directional relation), whereas red links indicate concentration flux with signaling feedback (bi-directional). In modular models derived from SBML models, yellow links are used to represent a concentration in a volume.

4.3.2. Visualizing a Composite Module

Composite modules of dynamic models provide terminals for the whole module. They are represented by a rounded rectangle with a white background and black border. As depicted in Figure 4.4, the interior and exterior can be visualized in different ways depending on the modeling task.

The most obvious way is to open the entire model and zoom to the particular composite module of interest. With this local view, only the module and its subelements are displayed. Edges to and from the exterior are shown and provide some contextual information. In Figure 4.4a a composite module describing a double phosphorylation of a protein is depicted. The storages `activ` and `deactiv` are connected to external terminals. Of course, a composite module can also be visualized as a 'closed' module, separated from the exterior context of the model. In this case, the composite module is the root of the inclusion tree. The focus is on the content of the composite module since all inner submodules are considered in the visualization. An example of that 'deep view' is shown in Figure 4.4b.

Similar to storages and reactions, composite modules can be decorated with bitmap icons depicting a rough sketch of the interior. This allows a customized representation of the underlying process, for example, a more intuitive reaction scheme of the double phosphorylation (Figure 4.4c). As shown in Figure 4.4d, the composite module can also be depicted with contextual edges to the exterior, but without

4. Visual Representation of Dynamic and Logical Modular Models

content. This option is useful, for example, in order to present a higher abstraction of the model with less detail. In this case, the visual complexity is decreased.



Figure 4.2.: Visualization of storages and reactions. (a) Standard representation of a storage using an ellipse with a red border, (b) storage with a bitmap icon indicating a modification state, (c) standard representation of a reaction using a rectangle with a blue border, and (d) reaction with a bitmap icon indicating a reversible reaction.

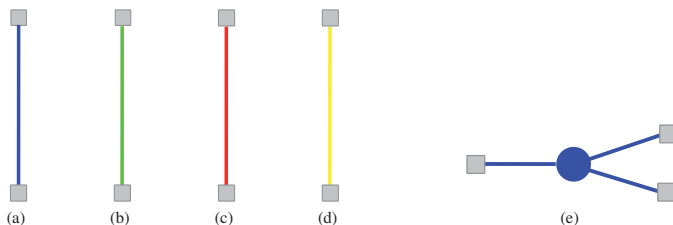


Figure 4.3.: Visualization of links and their related terminals. The link color can be defined to encode the type of connection established by the link: (a) Link encoding a flux, (b) link representing (signal) concentration, (c) link with both, concentration and flux, (d) link representing a concentration in a volume, and (e) link with link node.

4.3.3. ‘Phosphorylation’-Example

In Figure 4.5b, a simple dynamic model using typical components is presented that illustrates the visual representation. This model is defined in MDL in a modular fashion illustrated by Figure 4.5a. For an improved visualization the two different windows (showing the model `whole_model` and the composite module `cycle`) are integrated into a single view (Figure 4.5b). The model describes the phosphorylation of a protein. Thereby, in the module `cycle`, three storage elements `activ`, `deactiv` and `phosphatase` are used. They encode the two states of the protein (i.e., active and inactive) and the phosphatase. These storages are connected via a reversible reaction element `rev-reaction` describing a Michaelis-Menten kinetic. Two effectors control the reaction element – one from the exterior input (connected by the terminal) and the other from the storage `phosphatase`. The composite module is connected via terminals to the exterior modules `input` and `output` at the highest organization level.

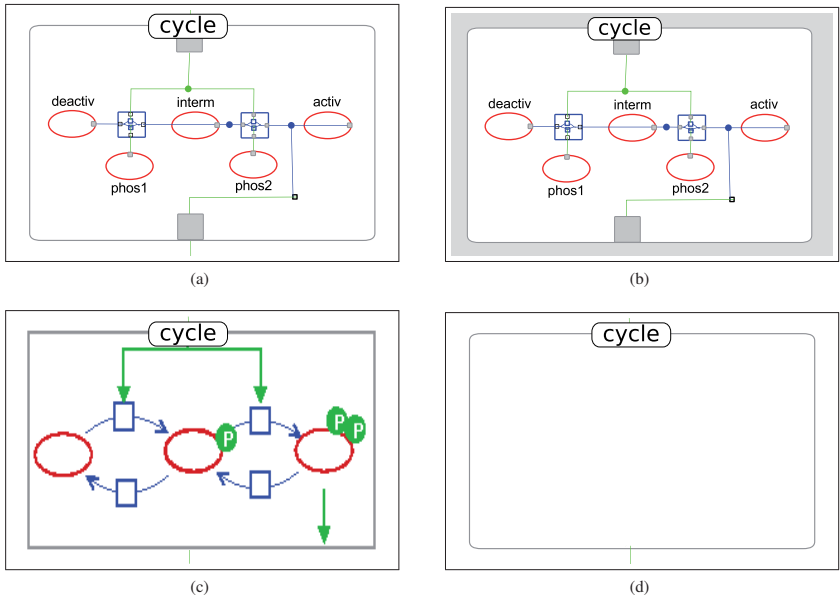


Figure 4.4.: Different ways of visualizing a composite module of a dynamic model. (a) Zoomed-in sub-module (with content and adjacent edges), (b) separately opened submodule (with content, no links to the exterior), (c) zoomed-in submodule (with bitmap icon), and (d) zoomed-in submodule (content is hidden, adjacent edges). The composite module is taken from Chapter 2, Figure 2.5.

4.4. Entities for Logical Modeling and their Visualization

Logical modeling results in a qualitative description of the model. Typically, logical models are characterized by many components resulting in large networks. Hence, visualization aspects are of great importance [170].

How qualitative model properties should be graphically represented is not yet standardized. For this purpose a draft is provided, which defines a special visual representation of modeling entities and the model structure.

Logical modeling in PROMOT is supported by the logical library with specialized classes. In order to represent the network in a biologically intuitive way, all modeling entities defined in MDL are inspected and then aggregated in a single integrated view. Nevertheless, the inherent modular structure of the model is still preserved. This step is analogous to the process described in Section 4.2. All elements necessary for the mathematical description of the logical formalism, but without a clear biological interpretation, are removed by editing the underlying graph structure. Particular steps are the simplification of complex structures and the removal of elements that are not needed in the later visualization. Once the terminal

4. Visual Representation of Dynamic and Logical Modular Models

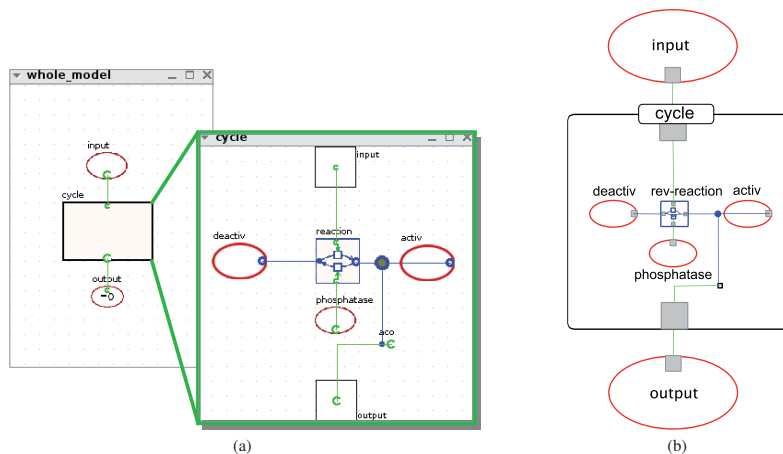


Figure 4.5.: A dynamic model using typical components. A phosphorylation step is encapsulated in the composite module `cycle`. This is a simplified version of the model given in Chapter 2, Figure 2.5.

cluster graph G_{TCL} is defined, the ‘Mapping’ step is applied. The mapping of data to topological and visual properties is the most critical step. Here, the visual properties of certain elements are modified to achieve a biologically more intuitive representation. In summary, the visual representation of logical models from MDL to the VISUAL EXPLORER implicates the following sequenced steps:

1. aggregating all modules and hierarchical relationships in a single visualization,
2. removing all information necessary for the mathematical description, but without a clear biological interpretation, and
3. altering the visual properties of certain elements using predefined mapping functions to achieve a biologically more intuitive representation.

Hence, any logical model, specified in MDL, independent of size and complexity, can be automatically converted to the formalized visual representation used by VISUAL EXPLORER. The following sections show in detail how such a representation can be reached.

4.4.1. Basic Elements

For the convenient visual setup of a logical model, different modeling classes are provided by the standard library for logical modeling. The basic classes are `compound` and `gate`. Similar to the dynamic modeling approach, they are elementary modules and are derived from the class `module`. Using the object-oriented modeling concept, these basic classes can be specialized (using sub-classing) in a straightforward manner. To each of these classes or subclasses several properties can be added. In addition, different types of *links* can be defined.

Compound Compounds are used to model logical states. A compound contains some common molecular classifications of certain species frequently used in signal transduction networks. For instance, subclasses of compound are `receptor`, `kinase`, `phosphatase`, `phenotype`, `ligand` or `drug target`. The subclass `other` serves as a generic compound that represents an unspecific molecular species. Additionally, the subclass `reservoir` representing a pool of species and the subclass `dummy` representing a generic entity for modeling different characteristics are provided. Most of the subclasses are mathematically equivalent to the class `compound`. The information of the specific subclass is used in the latter visualization [165].

Gate Gates include the logical connection of network elements for describing certain interactions. They are based on a common Boolean logic and its operators. Basic classes of logical gates are `and` (represents an ‘and’-connection of compounds), `or` (represents an ‘or’-connection of compounds), and `somehow` (can be used when the logic is not clear). Additionally, the class `boundary` refers to gates. It represents either an ‘input’ or ‘output’ of a compound. The class `input` describes an incoming signal, for example, an initial trigger; the class `output` denotes an outgoing signal of the model.

Link In the context of logical models, a link represents a relationship and connects gates to compounds or gates to gates.

In the next sections, important subclasses of `compound` and `gate`, and their special visualizations are introduced. A comprehensive overview including a ‘symbol reference card’ can be found in Appendix B, Figure B.2.

4.4.1.1. Compounds

As already mentioned, compounds can be specialized into different subclasses (e.g., `kinase`, `phosphatase` or `adapter-protein`). In order to distinguish different subclasses, they vary in their visual representation (see Figure B.2).

An important subclass is `reservoir`. It is used to denote a pool of entities (e.g., molecular species) relevant for signal transduction processes. For instance, a single reservoir can be used for the activation of two species. In general, this allows the modeling of simultaneous interventions (e.g., knock-outs) of multiple entities. A reservoir can be used from any reaction in the network independently from its localization. In recent tools, two different approaches exist to depict reservoir-like elements. First, multiple instances of a `reservoir` entity are each rendered near its adjacent compound. From a visual point of view, this has both, advantages and disadvantages: The advantage is having short edges between a `reservoir` entity and the adjacent compound. The drawback is that probably the connectivity does not match the actual topology of the network [26]. In the second approach only a single `reservoir` entity is drawn independent of the number of adjacent compounds. This has the advantage of having only one instance of a `reservoir`. The latter approach is integrated in the formalized visual representation presented in this chapter.

The modeler is able to customize the visual representations of `reservoir` elements. According to Figure 4.6, three different options can be chosen: In Figure 4.6a, the standard representation is shown where the `reservoir` is represented with a high level of detail. A more basic version is depicted in Figure 4.6b where the `reservoir` construct is simplified by removing the `input` element. In contrast to Figure 4.6a, influences of the `reservoir` on adjacent compounds are represented as dashed lines. A further simplification is shown in Figure 4.6c. Here, both the `reservoir` and the `input` element are removed. The influenced compounds are linked directly by a dashed line. It should be mentioned that the latter option can only be applied when the `reservoir` influences exactly two compounds. The

4. Visual Representation of Dynamic and Logical Modular Models

options in Figure 4.6b and Figure 4.6c reduce the visual clutter, especially in large networks and aim at a clear visual representation.

In contrast to dynamic models, compounds can be directly connected (if no gate is in-between). However, the situation appears different for gates. They cannot be directly linked; at least one element of the class `compound` has to be placed between two gates. Additionally, all compounds that have no upstream or downstream stimulating reaction must be connected to an `input` element or `output` element, respectively.

The class `dummy` is used as a generic compound for modeling specific characteristics of the system, for example, time delays. In many cases, these characteristics are encoded in the mathematical description and has no appropriate counterpart in the visual representation. Thus, depending on the purpose and semantics, the user needs either to visualize a `dummy` or not. Consequently, two different ways for the representation of the `dummy` element exist. As depicted in Figure 4.7, `dummy` elements can be visualized in or removed from the network. Figure 4.7a depicts a `dummy` element with an upstream activation and a downstream inhibition. When the `dummy` is removed the new overall reaction becomes an inhibition (Figure 4.7b). Another use case for `dummy` elements is a compact representation of the logic. This is exemplified in Figure 4.7c-4.7d. Figure 4.7c shows a network with complex logic using `and` gates and many (partly overlapped) reaction edges. This visual representation of the network can be defined in a more compact way by using `dummy` elements. Both figures show exactly the same input-output behavior, but Figure 4.7d is much more compact. This approach can reduce the visual clutter.

4.4.1.2. Gates

As already described, subclasses of `gate` are the logical operators `and`, `or` and `somehow`, and the boundary elements `input` and `output`. An `and` gate facilitates the combination of two or more distinct compounds into one logical construct. Thereby, all input species are ‘and’-connected. The `and` element is explicitly rendered using a light-blue circle with a dark-blue border. An ‘or’-logic is usually modeled implicitly (cf. Figure 4.8b). Sometimes the modeler is not sure about the logical behavior, especially between ‘and’-logic and ‘or’-logic. In this case, the ‘or’-logic can be defined by an explicit `or` gate (cf. Figure 4.8c). It is depicted by a light-blue circle with a magenta border. In this way, the ‘or’-logic can be switched easily to ‘and’-logic by simply replacing the `or` gate by an `and` gate. In most cases, an ‘or’-logic is represented implicitly. This leads to a more familiar and less cluttered visualization. Instead of having three visualizations of different gates in `VISUAL EXPLORER`, only two gates (`and`, `somehow`) must be visually distinguished. This results in an improved understanding of the network.

Logical models simplify the continuous behavior. During the transformation of continuous to discrete behavior, often the type of the logical connection is unclear or ambiguous; that is, several species are combined with an `and` gate or an `or` gate. The `somehow` gate can be used for these cases because it allows the logical connection to be modeled using an incomplete truth table¹. The `somehow` element is explicitly represented by a light-blue circle with a dark-red border (see Figure 4.8d). Once the logic is clear, the `somehow` gate can be replaced by an `and` gate or `or` gate.

4.4.1.3. Links

Similar to the representation in dynamic models (cf. Section 4.3.1.3), links are represented by edges. An edge connects two compounds, or a compound and a gate. All compounds and gates have to be connected. In MDL a certain type of interaction (e. g., stimulatory or inhibitory influence) can be defined. In the visualization of logical models, a positive influence on a biological entity (e. g., activating effect) is

¹For more details on incomplete truth tables and their usage, see Klamt et al. [104].

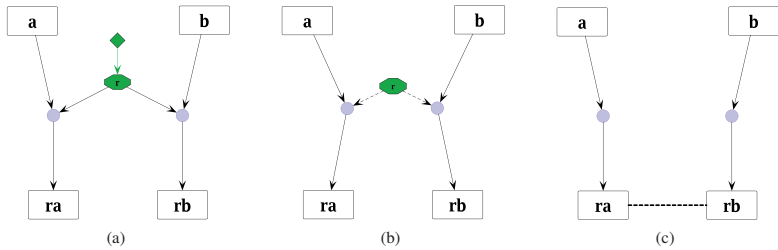


Figure 4.6.: Options for the visual representation of a `reservoir` element: (a) Standard visual representation with full detail. (b) For the sake of clarity the input element can be hidden and in (c) additionally, the `reservoir` can be hidden. In (b) and (c), influences of the `reservoir` on adjacent compounds are encoded by dashed lines.

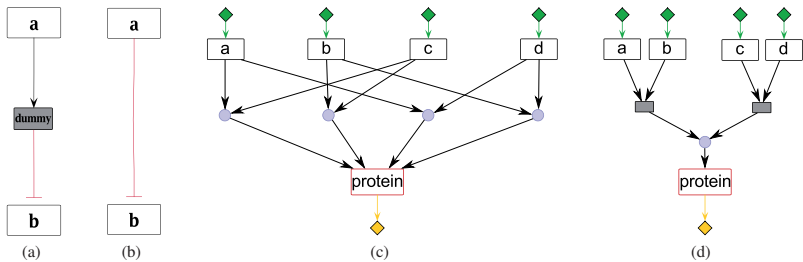


Figure 4.7.: Visual representation and usage of a `dummy` element: (a) The `dummy` element is shown; (b) the `dummy` element is removed. Usage of `dummy` for compact representation. (c) Common visual representation with explicit use of gates. (d) More compact visual representation using `dummy` elements.

indicated by a black edge (Figure 4.9a). In contrast, a negative influence on a biological entity (e. g., inhibitory effect) is encoded by a red edge (Figure 4.9b). Both representations have special target arrows – different for positive and negative effects, i. e., activation is encoded by a black target arrow, and inhibition by a red target ‘T-end’, respectively. The direction of signal flow is depicted using the target arrows mentioned before. This is a widely used visual representation in research of molecular biology (e. g., in textbooks). There are also special types of interactions such as a `reservoir` reaction (Figures 4.6b, 4.6c or 4.9c), an input reaction (Figure 4.9d), and an output reaction (Figure 4.9e). An undefined interaction is represented using a gray line with no arrow heads (Figure 4.9f).

By default, no terminals are rendered because in biological representations they are not common. However, an option for showing terminal nodes in logical models exists. Similar to dynamic models, terminals are placed at the inner border of the module. They are represented by squares with a dark-gray border and light-gray background.

4. Visual Representation of Dynamic and Logical Modular Models

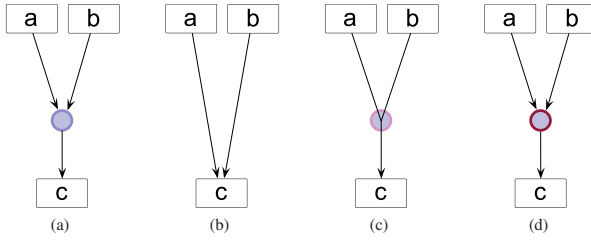


Figure 4.8.: Visualization of logical gates: (a) an *and* gate, (b) an *or* gate using an implicit representation, (c) an *or* gate using an explicit representation, and (d) a *somehow* gate. Gate elements can be visually distinguished by different border colors.

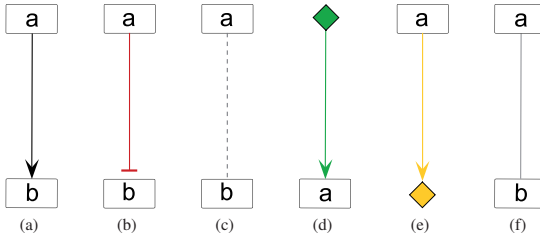


Figure 4.9.: Visualization of different interaction types in logical models: (a) Activation is encoded by a black edge with a target arrow; (b) inhibition is represented by a red edge with a target ‘T-end’; (c) a reservoir reaction is encoded by a gray, dashed edge; (d) an input reaction is displayed using a green edge with target arrow; (e) an output reaction is defined by a yellow edge with target arrow; (f) an undefined reaction is represented using a gray edge with no arrow heads.

4.4.2. Visualizing a Composite Module

Using the logical modeling formalisms, composite modules can be visualized in different ways. Figure 4.10 provides an overview. A first option is to open the entire model and zoom into the particular composite module. Then, having a local view, only the module and its subelements are displayed. Possible reaction edges from the outside are shown in part and provide some contextual information. In Figure 4.10a a composite module with subelements *d*, *e*, *f* and *and* is depicted. Subelements *d* and *e* are connected to incoming edges, *f* to an outgoing edge, respectively. As shown in Figure 4.10d, the composite module can also be depicted with contextual edges, but without content. This results in a higher abstraction with less detail, and therefore reduces the visual complexity. Composite modules can also be opened explicitly, independent from their parent module, but considering all submodules. An example is shown in Figure 4.10b. In this case, the composite module is the root of the inclusion tree. The composite module may contain terminals that indicate coupling to the exterior modules. These

terminal nodes are enhanced with labels for contextual information. Moreover, for this option, incoming and outgoing edges are drawn between terminal nodes and subelements in order to correlate both. The composite module can also be visualized as a ‘closed’ module; separated from the model context. For instance no context is given (e. g., edges or terminals to the outside). In this case, the focus is only on the content of the composite module. An example is shown in Figure 4.10c. There is an additional option to use a bitmap icon (similar to the usage in dynamic models described in Section 4.3.2).

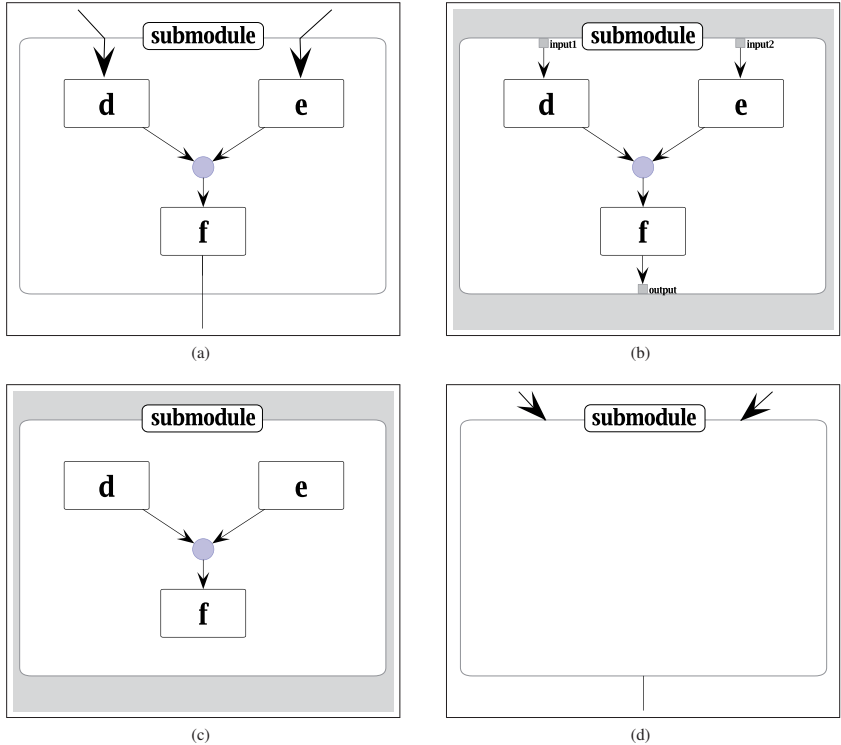


Figure 4.10.: Different ways of visualizing a composite module of a logical model. (a) Zoomed-in submodule (content is visible), (b) separately opened submodule (shows both content, and terminal nodes), (c) separately opened submodule (visualized as closed submodule, shows content but without edges to the exterior), and (d) zoomed-in submodule (content is hidden).

4.4.3. Visual Handling of Parameters

As mentioned in Section 4.4.1 different properties of modeling elements, in form of parameters, can be easily added. These parameters can be used, for instance, to define a time-scale of the logical functions, or to define the confidence of the encoded biological knowledge. Other parameters are necessary to encode options for the (visual) handling of logical models such as `visibility` or `map-number`. Most are dedicated to the further use in CELLNETANALYZER.

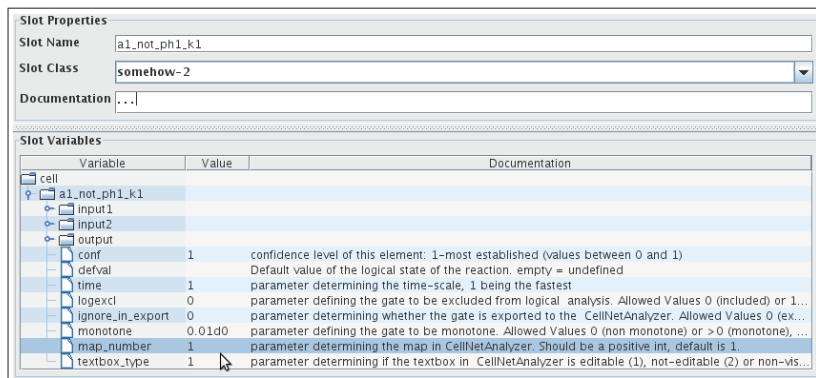


Figure 4.11.: VARIABLE EDITOR for a somehow gate in the Toy model example. The parameter *map_number* (`map-number`) is highlighted. Other parameters shown are, for example, *time* (`time-scale`), *logexcl* (`logical-excluded`) and *ignore_in_export* (`ignore-in-export`). Additionally, using this dialog, the name, the class and the documentation of the particular element can be changed.

Parameters of logical models can be set or modified through the VARIABLE EDITOR, a GUI component that was developed and implemented as part of this thesis (Figure 4.11). The general purpose of that component is the visual editing of properties for modules in the model. Additionally, it facilitates the modification of the name, class and documentation of the particular modeling element.

In the following, two parameters that are especially useful for the visualization of logical models are discussed. In Appendix B (Section B.3) further parameters are listed. Many are described in detail in Klamt et al. [104].

Visibility The parameter `visibility` is used for expressing that composite modules are not showing their contents. Hence, enabling this option, subelements of the respected composite module are not rendered. This property is important for adjusting the visual complexity and abstraction level of the model.

This can be performed by using the parameter `visibility`, example depicted in Figure 4.10d.

Map Number The parameter `map-number` can be used to define *multiple maps*. Such maps are comparable to local views – showing parts of the network. They are especially important for the integration with CELLNETANALYZER. The parameter `map-number` itself determines the specific map to which a certain network element belongs. Since the association is based on elementary and

not on composite modules, a map may comprise different hierarchical levels. However, often the modular structure of logical models is an appropriate starting point because within a module, elements have the same level of abstraction. Then all elements in a particular module have the same `map-number`. The application of the parameter `map-number` and its visual representation is described in more detail in Section 4.5.3.

In principle, all parameters mentioned before and listed in Appendix B can be used as metadata for the visual mapping to the elements in the network. For instance, all compounds in the network can be colored according to their specific `map-number`. In VISUAL EXPLORER the parameter `map-number` is needed for the convenient visualization and later export of multiple maps.

4.4.4. The Logical `Toy` Model

In this section, the visual representation of logical models is illustrated using the `Toy` model. As already mentioned, this process aims at designing a biologically motivated representation. The `Toy` model is defined in MDL in a modular fashion illustrated by Figure 4.12a. The model can be represented in a single integrated view (Figure 4.12b). Thereby, the local views separated into different windows (showing the model `whole_model`, and the composite modules `cell` and `nucleus`) are integrated in a single view. In Figure 4.12b, different modeling elements are shown. The information of a negative influence (inhibition) is encoded by a red edge of the particular reaction. A similar rule is applied to the positive influence (activation) where a black edge is used. Additionally, the direction of signal flow is indicated by target arrows that are different for activation and inhibition. Furthermore, the reservoir `k3r` has been automatically removed, since it belongs to the class `reservoir` and is not necessary for a clear biological interpretation. Instead, an edge connecting the corresponding compounds supplied by the entity pools is drawn using a dashed style. For a more biologically motivated representation the composite modules, in particular `cell` and `nucleus`, are drawn as ellipses rather than strict rectangles. Additionally, the ellipses are rendered in yellow and dashed (to imitate a biological, semi-permeable membrane). Interactions are rendered as curved edges² instead of angular edges. Both representations, ellipses and curved edges, are more familiar for biologists. More visualization examples of the `Toy` model are presented in Appendix B, Section B.4.

4.5. Applications

In the following, different realistic large models and their visual representations are shown. The models including their biological backgrounds were described already in Chapter 2, Section 2.5. First, the dynamic EGF model is presented. Then the visualization of two logical models – the logical EGFR/Erbb model and the logical TCR model – are given. Finally, the logical modeling workflow including the analysis tool CELLNETANALYZER is presented exemplarily using *multiple maps* and the associated parameter `map-number`. The models presented here or particular parts of them serve as examples in the subsequent chapters.

4.5.1. The Dynamic EGF Model

The biological processes described by the dynamic EGF model were already mentioned in Chapter 2, Section 2.5.1. How this model is visually represented in VISUAL EXPLORER is shown in Figure 4.13. Here, different modeling elements for dynamic modeling are used (e.g., *storage*, *reaction* or

²Curved edges are applied using Bézier curves. They are used to draw multi-segment edges very smooth, independent of the zoom scale factor.

adapter). Since the model makes extensive use of hierarchy and modularity, many composite modules are integrated. Most of the elements are not visible since they are located on deep hierarchical levels. Two composite modules `raf_ras_int` and `mek_int` are magnified to indicate more details.

4.5.2. The Logical Models: EGFR/ErbB and TCR

The visual representation for logical models can also be applied to large models. This is illustrated in Figure 4.14 and Figure 4.15. In these figures the models described in Chapter 2, Sections 2.5.1 and 2.5.2 are depicted. The first example shows the EGFR/ErbB model (Figure 4.14). It can be seen that the visual complexity and clutter is reduced. Both, compounds and interactions are presented more clearly, for instance, the `reservoir` constructs are simplified and explicit `or` gates are not rendered. An exception is the composite module `egf_ligand` on top of the network. It describes the combinatorial logic of the ligand-receptor bindings; and this logic is explicitly shown. However, there is the option to hide the interior of a composite module from view (cf. Figure 4.10d). Detailed information on the visual representations of the EGFR/ErbB model in different steps of the logical modeling workflow are given in Appendix B (particularly in Figure B.5).

Another example showing the TCR network is depicted in Figure 4.15. It provides a visual summary of the qualitative knowledge gathered while studying signal processing in T Cell receptor activation. Standard modeling entities and their parameters are visually modified by applying visual representations (e.g., `kinase` or `phosphatase`). The border of the composite module representing a membrane is modified (yellow dashed edge) to distinguish intra-cellular and extracellular processes. Also curved edges are supported because they are prevalent in (traditional) graphics of molecular biology. At the end, a more biological, clearer representation can be achieved; and a map is provided that encodes all necessary information in a visual manner. With this at hand, the modeler can zoom into particular parts of the signaling network to explore the underlying biological knowledge. The adequate visual representation can also be used as a visual aid for further analysis in CELLNETANALYZER.

4.5.3. Integration with CELLNETANALYZER – Multiple Maps

The workflow of logical modeling is already described in Chapter 2, Section 2.4.2 and illustrated in Figure 2.6. In this section the focus is on visual issues regarding the workflow with special emphasis on multiple maps.

As mentioned before, VISUAL EXPLORER can be used to generate adequate visual representations for the logical modeling formalism (an example is given in Figure 4.12b) and tools supporting structural analysis, for example, CELLNETANALYZER. For that, the formalized visual representations are exported as bitmap graphics to CELLNETANALYZER. This also includes the layout information of the network elements. In CELLNETANALYZER these static visualizations are used as background images.

As described in Section 4.4.3, CELLNETANALYZER supports multiple maps. In MDL the information on multiple maps is encoded in the parameter `map-number` for each network element. The following procedure is used to generate multiple maps for CELLNETANALYZER in an automatic fashion. First, for each network element the parameter `map-number` is obtained. This information is used to detect the specific submaps defined in the model. Then, all network element having the same `map-number` are saved in a group. For each of the groups the geometric bounding box is computed. Then, for each group VISUAL EXPLORER automatically zooms to the specified bounding box. In this step, the view is focused on the components with the same map number. The layout information of all network elements with that

4. Visual Representation of Dynamic and Logical Modular Models

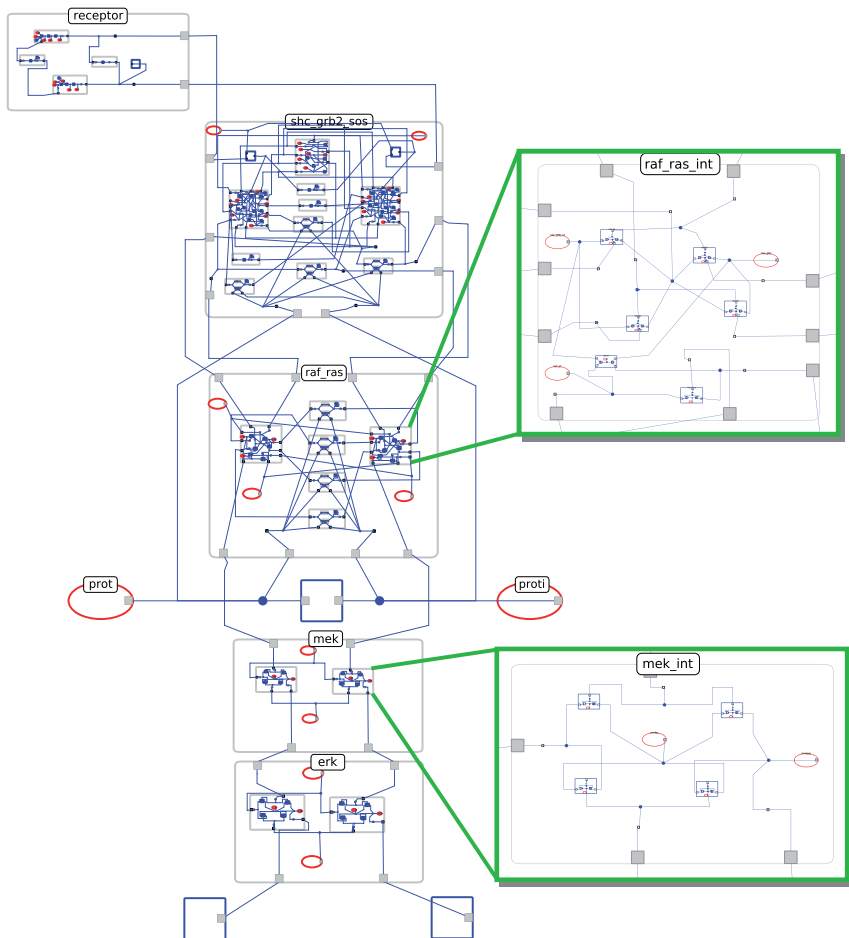


Figure 4.13.: Visual representation of the dynamic EGF model. Modules in very deep hierarchies are nearly invisible because of their small size on the screen. Two composite modules `raf_ras_int` and `mek_int` are magnified to provide more details.

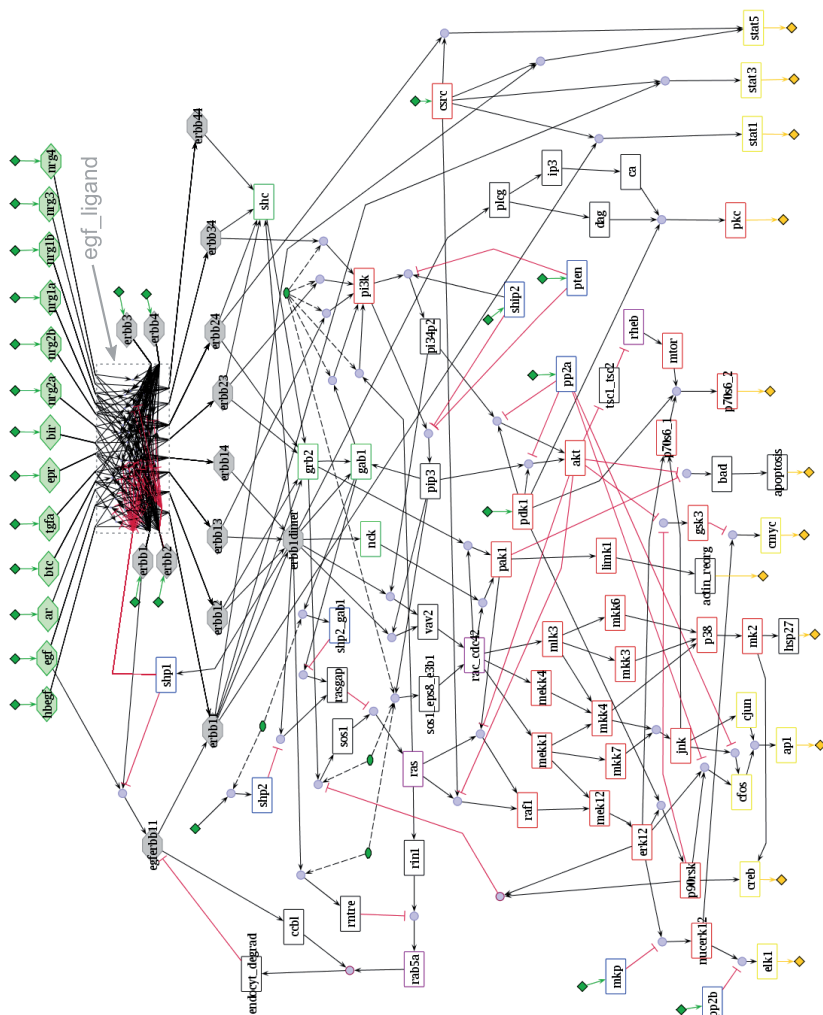


Figure 4.14.: Visual representation of the large EGFR/ErbB model. This shows a clear representation of compounds and interactions, except the composite module `egf_ligand` on top of the network. The visual complexity of this module can be reduced by hiding the interior from view. Figure is taken from Samaga et al. [167].

particular `map-number` is saved³. Finally, the visual representation terminated by the bounding box is saved as a bitmap image for further processing in `CELLNETANALYZER`.

In Figure 4.16, two states in this workflow are depicted. In Figure 4.16a, the parameter `map-number` is overlaid on the network of the `Toy` model using color coding. Three different submaps are defined and the maps are based on the predefined modules in the model, namely `whole_model`, `cell` and `nucleus`. In Figure 4.16b, these multiple maps are used to visually assist the structural analysis in `CELLNETANALYZER`.

4.6. Conclusions

In this chapter, first, it is shown how modular models specified in the object-oriented language MDL can be organized and managed in a single integrative view by utilizing a terminal cluster graph, a special compound graph. This, in combination with appropriate mechanisms for navigation and exploration, presents an interactive and very flexible way for visualizing different levels of scale and abstraction, and thus relates underlying biological concepts captured in MDL models.

In close cooperation with modelers, two formalized visual representations are developed which are specialized to the adequate representation of dynamic and logical models with modular structure. They are tailored to processes in signal transduction. For that, different graphical symbols for the individual modeling entities aggregating in a symbol set are provided. A general procedure was elaborated that combines the MDL syntax and the related graphical symbols to automatically generate the representations of MDL models. Hence, any dynamic or logical model specified in MDL, independent of size and complexity, can be automatically converted to the formalized visual representation used by `VISUAL EXPLORER`.

The graphical representation of dynamic and logical models aims to be, on the one hand, intuitive and, on the other, precise and descriptive while incorporating and depicting modular structures and different abstraction levels of knowledge. These graphical representations are intensively applied in several modeling projects that use realistic large models such as EGF or TCR (Sections 4.5.1 and 4.5.2).

Furthermore, such comprehensible, visual representations can be used to visually summarize the gathered knowledge about the system under investigation. In this way, the communication between different research groups can be facilitated. They are the basis for the interpretation of data generated in an external process, for example, analysis or simulation data that can be displayed in the context of the network model. The example in the last section shows the high level of integration between `PROMOT` and `CELLNETANALYZER`, which improves the structural analysis and interpretation of logical models.

³Note that not all network elements depicted on the submap must have the same `map-number` since the bounding box has to have a rectangular shape, whereas map numbers can be located arbitrarily in the network. Moreover, bounding boxes specified for different map numbers may overlap.

4. Visual Representation of Dynamic and Logical Modular Models

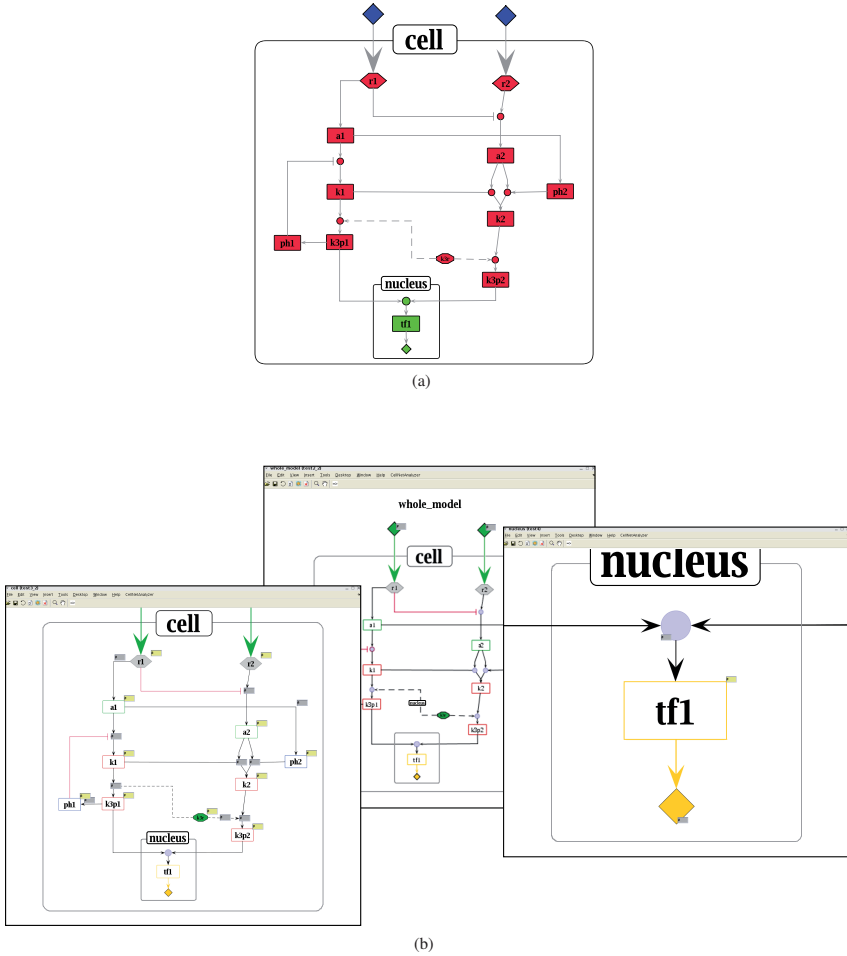


Figure 4.16.: The use of multiple maps for CELLNETANALYZER is illustrated using the Toy model. In this case, the definition of multiple maps is based on existing composite modules in the model. (a) Visualization of the parameter map-number in VISUAL EXPLORER. The parameter values for map-number are encoded by node color (whole_model with value '1' is encoded in blue; cell with value '2' is encoded in red; nucleus with value '3' is encoded in green). (b) CELLNETANALYZER screenshots from the three submaps specified in PROMOT and generated by VISUAL EXPLORER.

The best visualizations are not static images to be printed in books, but fluid, dynamic artifacts that respond to the need for a different view or more detailed information.

Colin Ware [203]

CHAPTER 5

A Visual Interface for Interactive Exploratory Model Analysis

In this chapter, different *interactive exploration schemes* for complex structures of modular models are presented. These schemes are combined in a visual interface for the interactive exploratory analysis of models. In the first part, the research problem is outlined. Then the interactive exploration schemes are introduced, followed by a comparison of them. Finally, some use cases illustrate their application.

5.1. Research Problem

As mentioned in Chapter 3, Section 3.6.2 most of the tools operate on flat models – even if they are large and complex. As the integration of models is pushed further, models become larger and larger. In many cases they do not fit onto the screen and the elements that are outside of the viewing window have to be explored by standard Pan+Zoom operations. Frequently used zoom operations are often strictly discrete. As a consequence, the mental map could be lost through the abrupt changes in the view. Additionally, standard zoom emphasizes only the detail of model parts but this results in a loss of context and does not enable different levels of information to be represented in the cellular network.

Even in modular models with many and deep hierarchical levels, the completion of a particular task by navigation and exploration can be confusing and disorientating without visual cues; the modeler is confronted with too much information. A lot of the information is not relevant or not interesting in the context of the specific task. In the context of this thesis common tasks for exploratory analysis are the following:

- examine the overall structure and relations of the intra-cellular network,
- recover specific entities deep in the hierarchical structure,
- find and view all biological entities that will be affected by a certain biological entity of interest (e. g., a protein),
- view a particular biological entity or a group of entities in the context of the entire network,
- elucidate properties of biological entities like concentration, signal, initial parameters or activity state in the network context of the entire system, and
- find functional units, duplicates, interdependencies, and inheritance properties of the overall network structure.

These tasks can be performed in an efficient manner when the structure of the modular model is fully integrated into the visualization and navigation capabilities of the respective tool [158]. It must be able to navigate within the hierarchical structure, for instance, to find the actual position of an element in the hierarchy. Standard Pan+Zoom operations are not sufficient; they do not consider structural properties. Based on the analysis of existing tools and several reviews from literature, both discussed in the introductory chapters, a set of requirements for the visualization and navigation of modular models has been identified:

- For the application of modular models the simple graph model (introduced in Chapter 3, Section 3.1.1) is too restrictive and, therefore, does not model the data encoded in a modular model adequately. In contrast to simple graph models, compound graphs models are able to model a nesting hierarchy; that is, the graph structure is used to produce a meaningful inclusion tree. Hence, an integrated single hierarchical model based on a terminal cluster graph, a specialized compound graph, should be used.
- Model parts need to be scaled depending on their level in the hierarchy. In other words, elements on deeper hierarchical levels must be scaled down by a constant factor. This saves space on the limited screen and can be seen as analogous to the organization of things in nature.
- The intrinsic hierarchy of the compound graph should be captured. Then it can be used as a paradigm for navigation and exploration of the model.
- Multiple interactive exploration schemes should be provided – also in combination – to flexibly support various modeling tasks in an intuitive and efficient manner.
- The focus should be on navigation and exploration, not on editing. That means, mainly several functionalities of a viewer including different model views, advanced navigation pattern and animations (cf. Chapter 3, Section 3.3.4.1) should be supported. A few basic editing functions can be provided, but are not mandatory.

The first two points are already mentioned in Chapter 4. The rest of the requirements are discussed in the chapter at hand. In the following sections, the interactive visual exploration of model parts in the context of the entire modular system is presented. Thereby, the application of proven concepts from the research of information visualization to the domain of systems biology is in focus.

5.2. Interactive Exploration Schemes for Modular Models

In order to facilitate the intuitive navigation and the flexible visualization several specific interactive exploration schemes are implemented in *VISUAL EXPLORER*. These implementations are based on the ZUI toolkit *PICCOLO2D*¹. The interactive exploration schemes can be used as navigation and exploration modes with the integrated view and are based on seamless zooming, semantic zooming, and content- and topology-aware navigation.

In the following, the individual schemes are introduced. In particular, these are *InteractiveZoom*, *ContextualZoom* and *HierarchicalZoom*. Furthermore, the integration of the interactive exploration schemes with complex structures of modular models is described in detail. They are strongly correlated to the steps ‘Mapping’ and ‘Rendering’, and to the specific user interactions of the visualization pipeline (cf. Chapter 3, Figure 3.6).

¹Piccolo2D is already discussed in detail in Chapter 3, Section 3.6.1.

5.2.1. InteractiveZoom

The InteractiveZoom facilitates geometric zooming by providing efficient Pan+Zoom operations. *Pan* in the InteractiveZoom is similar to standard pan operations in many other tools. The modeler can move the entire network with constant scale of the view. By zooming in, the area in focus is enlarged; that is, details on deeper hierarchical levels are depicted more clearly. In contrast, zooming out displays a more abstract view of the focused area. In this way, the signaling network can be interactively explored at different levels of abstraction, for example, a global view for high-level processes or several local views for processes described in detail.

As stated in Chapter 3, Section 3.4.2, the zooming functionality of the majority of tools is restricted to discrete steps. This can destroy the mental map of the modular model. For instance, the modeler can perform several zoom-in operations, and finally gets lost in the hierarchy. In contrast, InteractiveZoom features seamless zooming that enables smooth transitions between the subsequent views. In this way, the mental map can be preserved. An additional feature is that zooming performed by the InteractiveZoom is relative to the mouse pointer, and not defined by the center of the viewing window. This is an essential advantage because the direction of the zoom can be actively controlled by the modeler. For instance, the modeler can perform a zoom-in operation directly on an element located peripherally in the network without moving it out of the viewing window. This is more efficient in terms of the amount of navigation operations while exploring a modular model. Zooming, in the InteractiveZoom, is a non-context-preserving navigation technique; that is, the global context gets lost while zooming in.

5.2.2. ContextualZoom

The ContextualZoom provides an exploration scheme with interactive Focus+Context. Thus, the representation of the modular model is modified by selectively emphasizing elements (focus) and de-emphasizing elements (contextual information). Emphasizing and de-emphasizing can be performed by adjusting the scale of elements (i.e., size), and thus the level of detail they can present on the screen. The ContextualZoom can be compared to a ‘local’ zoom-in of a focused element (Figure 5.15a). In this way, detailed focus areas and less detailed context areas are established in the same view. An example is depicted in Figure 5.1. The ContextualZoom complies with the following requirements:

- R1. In order to follow the potential changes of the network, topological properties such as orthogonal ordering and proximity should be preserved. This also includes the prevention of node overlapping.
- R2. The overall model size should be fixed. This can be achieved by scaling all elements (zoom in focus element, zoom out all other elements). For instance, the size of the composite module *raf_ras* in both subfigures of Figure 5.1 is the same.
- R3. It should be able to propagate to upper hierarchical levels. Thus, if not enough space is available at the current hierarchical level then the next-higher level should be used.
- R4. For an easy comparison of elements located within different modules and at different hierarchical levels, it is of advantage to define an arbitrary number of concurrent foci. By applying multiple foci, it is possible to “display both short and long range correlations and patterns in data simultaneously and their possible inter-relationship”[160].
- R5. Bend points are an integral part of the overall appearance of the network. Thus, they should be considered during modification in order to maintain the mental map.
- R6. Since relations are as important as elements, ContextualZoom of edges should be considered; that is, focusing on an edge forces its end nodes to perform a ContextualZoom (multiple foci).

5. A Visual Interface for Interactive Exploratory Model Analysis

- R7. The sequence of transformation steps must be reversible and unambiguous in order to maintain the mental map of the modeler. The reverse step can also be used for de-emphasizing elements (reduce scale of elements).
- R8. The technique should be performed in real-time and changes should be animated in order to maintain visual context.

Many of the requirements are fulfilled by existing Focus+Context techniques, for example, R1-R3 [186], R4 [12, 172], R5 [171] or R7-R8 [158]. However, they still need to adapt to modular models and the domain of signal transduction. All requirements are implemented as part of the ContextualZoom. The ContextualZoom approach is based on the concept of manipulating intervals that splits the viewing window into different horizontal and vertical segments. Thus, basically, the ContextualZoom does not work on elements, but rather on intervals. Similar ideas are presented by Bartram et al. [12] and Storey et al. [186]. In the following paragraphs different relevant aspects of the ContextualZoom are described in more detail.

5.2.2.1. Consideration of Edges

Some existing approaches ignore the edges and their complex routes in the network. In modular models edges can relate to different hierarchical levels. Following edge routes and preserving the mental map gets even more complicated in signaling networks where each element often has more than one interaction partner resulting in tightly interlinked networks.

The computation of routes of straight lines is not complicated. For example, as depicted in Figure 5.2 (e. g., the two lines from the adapter protein `a1` to the kinase `k1`), the straight lines in the initial view are mapped to straight lines in the view generated by the ContextualZoom.

Polylines² must be handled differently. They are mapped from the initial to the final view by treating each bend point as a separate node. Thereby, the node has an interval of size 0. In this way, the mapping of polylines is reduced to the mapping of their line segments. Then each line segment is defined by a straight line that can be easily mapped. In general, parallelism between edges is not preserved except they are vertical and horizontal. The mapping of polylines was already described by Sarkar and Brown [171]. However, this approach is not based on intervals.

5.2.2.2. Multiple Foci

The ContextualZoom can handle multiple foci (independently zoomed elements). As depicted in Figure 5.3, three elements are in focus and, therefore, increased in scale. All other elements are de-emphasized by reducing the scale (i. e., size).

Often elements are located within different modules and at various hierarchical levels. By focusing on two or more areas in the network simultaneously, the emphasized modules can be visually correlated without losing context – even if they have a high spatial distance in the network layout. Hence, important structural features can be revealed such as network-wide dependencies. An example where multiple foci are used for visual correlation is discussed later in this chapter (Section 5.5.2).

5.2.2.3. Scale Factor

For balancing local detail and global context in the ContextualZoom the appropriate value of the zoom scale factor is essential. In Figure 5.4, it is shown how the zoom scale factor influences the final visualization result. The zoom scale factor depends on the modeling task, and thus is easy to configure by the

²A polyline is an edge with multiple bend points and line segments in between.

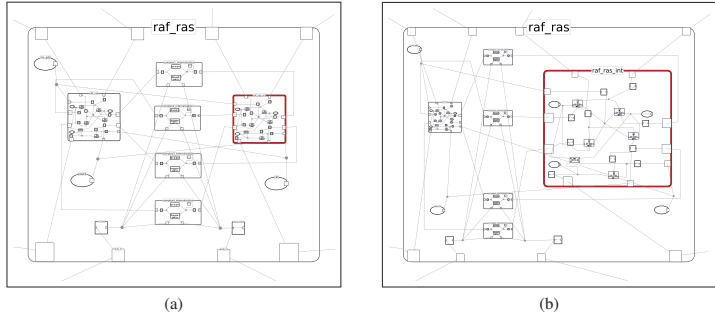


Figure 5.1.: The protein complex *raf_ras* encapsulated in a composite module as part of the EGF model displayed in VISUAL EXPLORER. (a) Initial visualization without a focus area. Altering the visualization from the initial state (a) to the final state (b) using the Contextual-Zoom preserves all elements and their relationships (contextual information), but gives the particular element highlighted in red approximately six times more space for presenting its content and additional information compared to the area in the initial state.

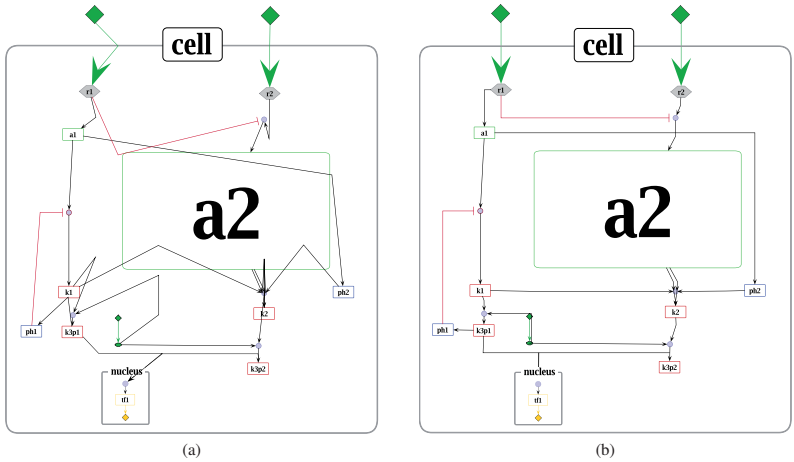


Figure 5.2.: Side-by-side comparison of ContextualZoom (a) with and (b) without consideration of edge bend points. The scale of the adapter protein *a2* is increased. In (b) the mental map is preserved. The original network layout is shown in Figure 5.4a.

user via a GUI component.

In summary, the ContextualZoom facilitates the simultaneous display of cellular information at different levels of abstraction. Moreover, it guides the attention of the modeler and helps to differentiate between important and non-relevant biological information regarding the current modeling task.

5.2.3. HierarchicalZoom

Many software tools provide generic exploration schemes such as Pan+Zoom or Overview+Detail as navigation facilities for different exploratory tasks. However, dealing with hierarchical structures of modular models, navigation is not efficiently supported. The navigation mechanisms should be closely linked to the content and its structure. Hence, it will be of great value for the modeler to use content- and topology-aware navigation techniques able to enhance the exploration process [134]. As stated in Chapter 3, Section 3.5.2, modular models consist of several units with varying organization and abstraction levels forming a nested hierarchy of modules. These topological features of the underlying compound graph can be used as a central paradigm for navigation and exploration of the modular model.

The HierarchicalZoom was developed to support such topology-aware navigation. A typical navigation process is explained by means of Figure 5.5 where the modular EGF model is explored. Initially, the focus is on a specific module using a local view. Focusing on a particular element results in an animated zoom sequence out of the hierarchy guided by the modular structure, for example, by considering the inclusion relations. Thereby, the view is centered on the next logical parent and, accordingly, is scaled to the size of the viewing window. Zoom-out operations are performed in several hierarchical steps until the root module is reached. This results in a global view showing the complete model. Finally, zoom-in operations are performed until a certain, very detailed module deep in the hierarchy is focused using a local view. Each zooming operation is described by a smooth, non-uniform animation. Alternatively, the HierarchicalZoom also provides an option to directly zoom to the focused node without considering hierarchical steps in between. This can improve the exploration time.

5.2.3.1. Edge Topology

Not only modules can benefit from such a technique. Another way is to use the topology of the edges between modules. In complex structures of modular models, edges can often be long, connecting elements on different hierarchical levels and can cross many other edges. Here, topology-aware navigation techniques such as ‘zoom to source’, ‘zoom to target’ or ‘zoom along edge’ can improve the exploration experience. These examples are discussed in more detail in Section 5.5.3. Another possibility is zooming to the entire edge. In the HierarchicalZoom this is computed by the union of the bounding boxes of the entire edge route and the two adjacent modules. Thereby, terminal nodes interfacing different modules are not considered. The edge and its related modules are displayed with maximal size on the screen.

5.2.3.2. User Interactions

Enabling the HierarchicalZoom, both, modules and edges are sensitive for user interactions. Usually, a focus is defined by clicking with the mouse within the bounds of the module or directly on an edge, respectively. For zooming in, the focus is lead on a particular module or edge of interest. For zooming out (navigating to the next logical parent), the periphery can be focused. The focus areas are sensitive to user interactions as depicted in Figure 5.6. The navigation using the HierarchicalZoom is illustrated in animation movies mentioned in Appendix A, Section A.1.

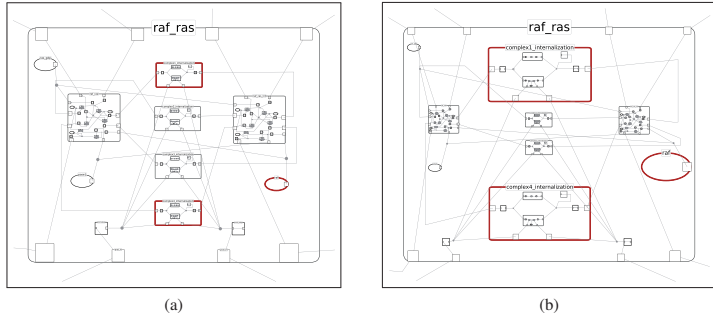


Figure 5.3.: The protein complex `raf_ras` as part of the EGF model displayed in VISUAL EXPLORER. Altering the visualization from the initial state (a) to the final state (b) preserves all elements and their relationships (contextual information) but gives the particular elements in focus (kinase `raf`, protein complex `complex1_internalization`, protein complex `complex4_internalization`, depicted in red) more space for presenting its content and additional information compared to context elements.

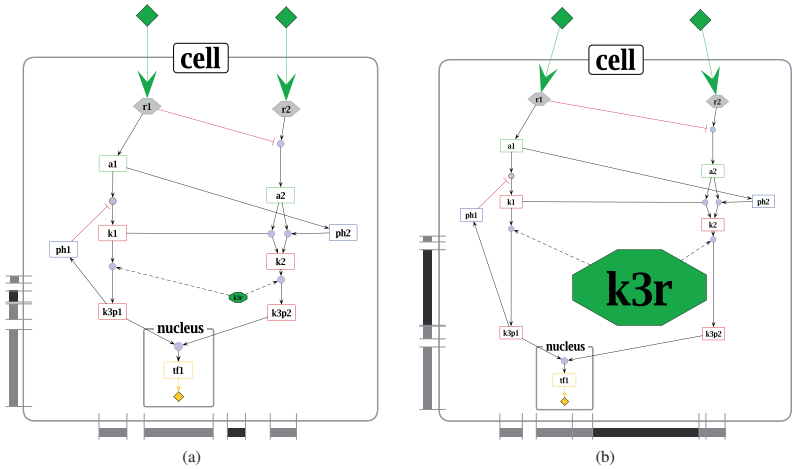


Figure 5.4.: Focusing on receptor `k3r` using ContextualZoom with different scale factor s . (a) $s = 1.0$ and (b) $s = 11.25$. Intervals in light gray depict shrinking segments; intervals in dark gray show enlarging segments. Figure is inspired by Schaffer et al. [172].

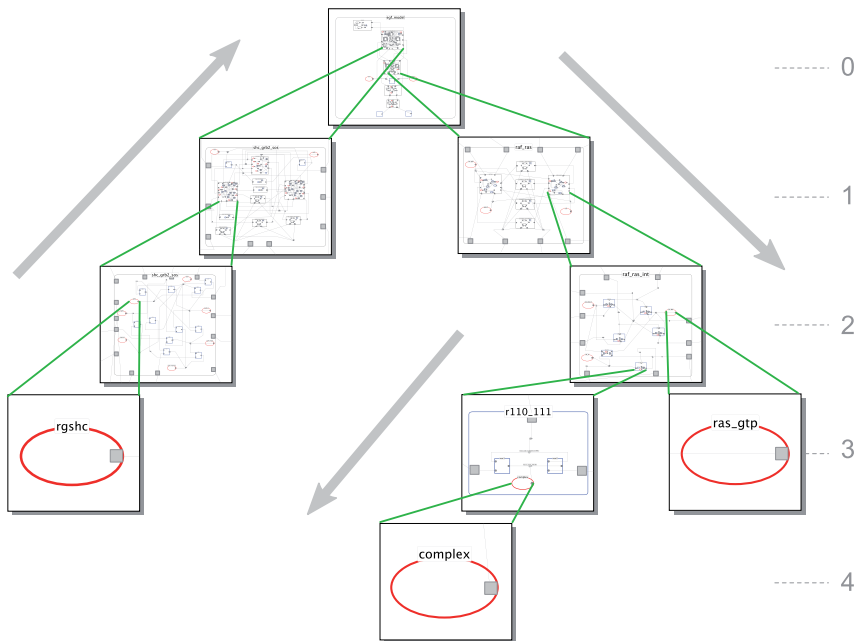


Figure 5.5.: HierarchicalZoom based on topological features of the modular EGF model. Several zoom-out and zoom-in operations from protein *rgshc* (left) over root element *egf_model* representing the most abstract view (middle) to protein *ras_gtp* (right) on efficient and comprehensible paths through the hierarchical structure. Numbers on the right side denote the hierarchical levels.

In summary, in the HierarchicalZoom intrinsic, topological features of the compound graph are used to navigate on efficient and comprehensible paths through the hierarchical structure. The navigation is based on user interactions, for example, simply by defining the area of interest. In this sense, the modeler is guided by rich navigation cues until the targeted element is reached. Moreover, topology-aware navigation cues ensure the user's orientation in complex structures. Hence, it is nearly impossible to get *lost in information space*, for example, by navigating to empty regions in the network.

5.3. Animations, Semantic Zooming and LOD

In the following section, different techniques relevant for InteractiveZoom, ContextualZoom as well as HierarchicalZoom are discussed. They are adapted to modular models and the domain of signal transduction, and thus are able to improve the exploration experience. In particular, these techniques are non-linear animation functions, semantic zooming with level of detail and efficient, topology-aware Pan+Zoom operations.

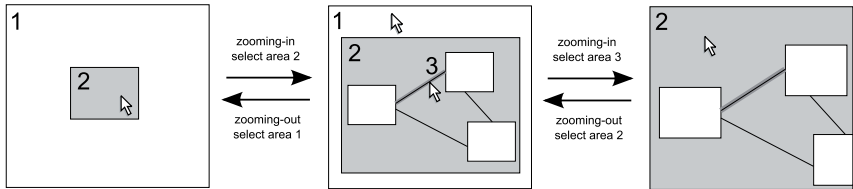


Figure 5.6.: HierarchicalZoom can be used with focus areas sensitive for interactions (encoded with gray shades). Zoom-in operations can be performed by selecting the element in focus (area 2 for submodule or area 3 for edge, respectively); zoom-out operations can be done by selecting the peripheral element (area 1 or area 2 for parent module). In most cases, this is the direct parent in the hierarchy.

5.3.1. Smooth (Non-)Uniform Animations

As mentioned in Chapter 3, Section 3.3.4.1, a fundamental characteristic of interactive ZUIs are animated transitions. They can be applied to diverse geometric transformations such as translation and scale of the view or a certain element. Important in this context are smooth, non-uniform animations. They provide immediate feedback and maintain the visual context. Non-uniform animations are typically used for camera and view transformations, and for some glyph animations such as translations.

InteractiveZoom, ContextualZoom and HierarchicalZoom are aided by animated transitions. For both, InteractiveZoom and ContextualZoom uniform animations are applied. This is applicable because the individual animation steps are too short to have more complex (non-uniform) animations. However, the InteractiveZoom overall type of animation depends on the specific user interactions. Hence, the modeler decides which animation behavior is applied. For instance, when the modeler performs a non-uniform translation with the mouse, also the overall animation can be understood in terms of a non-uniform transition.

For HierarchicalZoom a uniform animation is not appropriate. It rather employs the optimized version of the non-uniform animation function as described in Chapter 3, Section 3.3.4.1. The animation speed of this function slowly increases in the beginning and slowly decreases in the end of the animation sequence (i. e., slow-in/slow-out). Hence, the modules where the animation starts and finally ends are emphasized. By applying smooth, non-uniform and optimized animation functions, the interactive exploration schemes facilitate efficient transitions from one to another visual state. As already mentioned, this is an important component for preserving the mental map.

5.3.2. Semantic Zooming and LOD

VISUAL EXPLORER also supports semantic zooming, which is useful to graphically abstract the modular model. In many cases, the modeler is confronted with too much information – often not relevant or not interesting in the context of the task. Semantic zooming is able to show only information that is important for a particular modeling task. Some example tasks are given in the beginning of this chapter.

As discussed in Chapter 3, Section 3.4.2 an important property of semantic zooming is LOD. Different visual states can be defined for an element. By applying semantic zooming, an element can then switch between the different states. In general, the LOD is assigned through user interactions. As depicted in Figure 5.7, elements and their semantic information are altered according to the zoom level of the view of the modular model. For instance, starting with a zoom-in operation using the InteractiveZoom,

only molecules are displayed (Figure 5.7a). When zooming in further, the logical gates and names of molecules become visible (Figure 5.7b). Finally, at a higher zoom level relationships between molecules are displayed (Figure 5.7c).

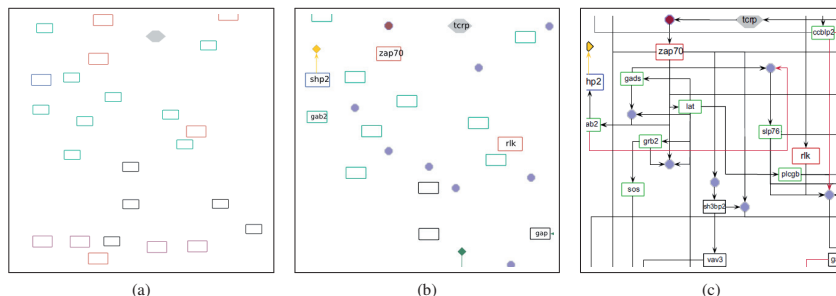


Figure 5.7.: Three consecutive screenshots of the logical TCR model captured when performing a zoom-in operation in VISUAL EXPLORER. Thereby, semantic zooming is applied. Elements and their semantic information are altered according to the zoom scale level of the network. (a) Only molecules distinguishable by border color are displayed ($s = 1.0$). (b) Molecules, their names, and logical gates are shown ($s = 1.3$). (c) All elements and their relationships become visible ($s = 1.5$).

The LOD defined by semantic zooming is a domain-specific issue (cf. Chapter 3, Section 3.4.2.2). For that, the modeler can select the threshold of LOD to influence the visual states in a modular model (e.g., the amount of visible objects). As depicted in Figure 5.8, this can result in a reduction of visual complexity and density by hiding unnecessary details and uninteresting parts of the structure.

5.3.2.1. LOD of Elementary and Composite Modules

For each elementary module four different visual states relevant for semantic zooming are defined – *full detail*, *without content*, *without label* and *invisible*. The first three visual states are depicted in Figure 5.9. In VISUAL EXPLORER, the LOD function of each node in the network including elementary modules, composite modules, text labels and terminal nodes depends on the displayed size and is a binary operator. When the size of a particular element is under a certain threshold this element becomes invisible. When the size is above the threshold the element is rendered according to its specification. The threshold can be adjusted using a GUI component. Composite modules allow a hierarchical view of all submodules. The LOD of a composite module is based on the LOD (i.e., size) of its inner elementary modules and terminal nodes. In other words, the LOD states of all elementary modules and terminal nodes results in the visual state of the related composite module. This can vary among different zoom levels of the view or different scales of the composite module, respectively.

The visibility of a bitmap icon is not controlled by semantic zooming; it can be manually switched on or off by an additional option in the GUI. An example of an elementary module with rendered icon depicting the reaction scheme is shown in Figure 5.9d.

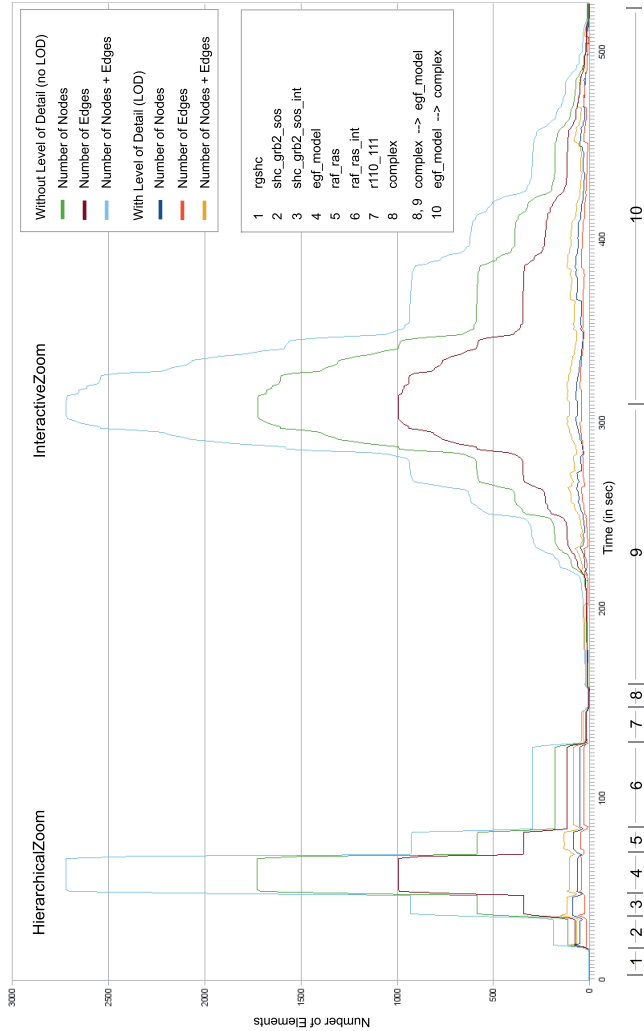


Figure 5.8.: LOD applied to the modular EGF model. A typical zoom sequence according to Figure 5.5 (left) with and (right) without module-based navigation. Module borders are steps in the plot; plateaus are pauses in the zoom sequence. Pauses are needed for visual inspection of the model. The zoom steps are performed either with or without LOD. As depicted with enabled LOD the number of nodes, edges and overall elements are at a low constant level while zooming in and out the hierarchical structure. Thus, the essential structural properties are displayed – and at the same time the visual density and complexity can be reduced. (Left) HierarchicalZoom and (right) InteractiveZoom.

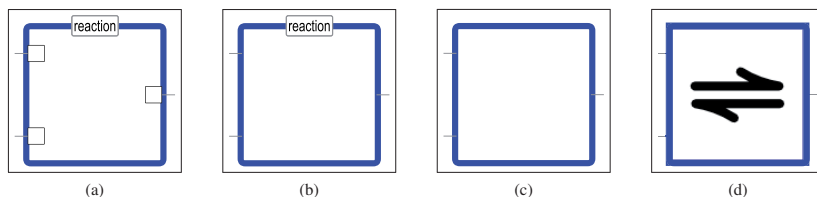


Figure 5.9.: Visual states using the example of an enzymatic reaction – (a) full detail, (b) without content, (c) without label, and (d) using an icon representation. The visual state using a bitmap icon is not controlled by semantic zooming. It can be manually switched on or off in addition to the LOD states in (a)-(c).

5.3.2.2. LOD of Edges

The LOD of an edge depends on the visual state of its adjacent modules and terminal nodes. When all adjacent modules and terminals are rendered (i.e., they are above a certain LOD threshold), also the edge is visible. This is obvious because a relation between two interaction partners (e.g., proteins) only makes sense when both partners are visible.

Figure 5.8 shows the number of visible edges (and also nodes) during a typical zoom sequence in VISUAL EXPLORER. The zooming pattern is based on the sequence from Figure 5.5 and is performed with different exploration schemes, and either enabled or disabled LOD. On the left, different zooming operations from the elementary module `rgshc` over the high-level module `egf_model` to the elementary module `complex` is performed using the HierarchicalZoom. On the right, zooming is performed from the elementary module `complex` to the high-level module `egf_model` and back again to the elementary module `complex` using the InteractiveZoom. As depicted in Figure 5.8, the overall number of elements with enabled LOD mode is nearly constant (around 100 visible elements) in order to maintain a consistent level of complexity, whereas the number of elements with disabled LOD increases to over 2700 and decreases again. An advantage of an enabled LOD is the decreased number of elements to draw and the resulting higher performance.

5.3.2.3. Options for Adjustments of LOD

As described in a previous section, the high-level representation of a particular module is obtained by hiding the subelements (e.g., elementary modules or terminal nodes) and their related edges.

In order to adjust the LOD to the application domain and individual preferences, the thresholds of LOD for nodes, edges and text labels can be configured and fine-tuned through a GUI options dialog.

In addition, to the LOD, driven by semantic zooming, in some cases it is necessary to manually adjust the visual states of elementary or composite modules such as with/without content, label or icon. This can be performed by different options for each individual element.

In summary, it can be shown that in different models such as EGF model (cf. Figure 5.8) or `Ecoli` model (cf. Figure C.2 in Appendix C) the enabled LOD can reduce the overall number of elements by a factor of 3 to 25 depending on the zoom level. This leads to the visual representation of essential structural properties and at the same time the reduction of visual density and complexity.

5.3.3. Efficient Pan+Zoom Operations

In modular models often users navigate with operations separated in pan and zoom. For example, the modeler is focusing on a biological entity using a local view and is interested in a peripheral entity on the same abstraction level. As depicted in Figure 5.10a the modeler first performs several pan operations, but perhaps gets lost in the hierarchical structure since the global context is not always shown. In order to orient, the modeler performs a zoom-out operation and after locating the entity of interest the modeler, again, performs a zoom-in operation.

Performing separate Pan+Zoom operations is a common task, but is not an efficient navigation in the sense of the number of navigational operations. A well-known example for an efficient navigation is given by the web mapping service GOOGLE MAPS [75]. There the modeler is supported by animated ‘flights’ – from the start location to the destination. First the modeler zooms out applying a high-level or global view. On that scale, the modeler performs a single pan operation to see the destination in context. Then the modeler zooms in, using a local view focusing on the targeted location (see Figure 5.10b). This represents an efficient strategy because, in most cases, this is the shortest path between the two specific places [65].

However, this strategy does not use the structural properties of the model. In VISUAL EXPLORER the model structure is used while zooming. The modeler is able to zoom out in hierarchical steps³ until the common parent of both, the source and the target, is visible in the view. Now both locations can be related to each other and understood at a glance. Then the modeler can zoom in hierarchical steps until the target is reached. Thus, the pan operation on the high-level view is not necessary. The navigation in hierarchical steps can also be performed in an automatic fashion when the source and target are known, for example, when performing a search operation (see example in Section 5.5.1). In this way, navigating is guided by the modular structure and enables the modeler to experience the structure (Figure 5.10c).

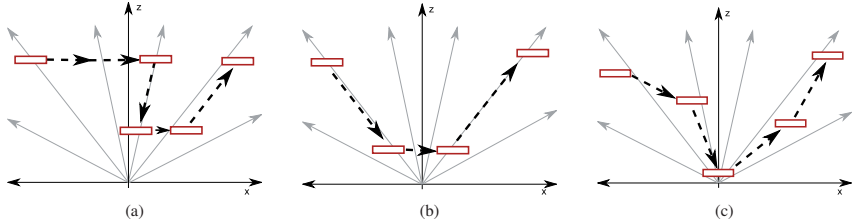


Figure 5.10.: Different strategies for Pan+Zoom operations applied to space-scale diagrams: (a) subsequent separate Pan+Zoom operations, (b) efficient operations like in GOOGLE MAPS, and (c) subsequent operations with combined Pan+Zoom considering the modular structure.

5.4. Combining and Comparing Interactive Exploration Schemes

VISUAL EXPLORER also provides ways to combine the different interactive exploration schemes. This is helpful in order to improve the navigation and exploration process of modular models. For instance,

³Zooming in or zooming out in hierarchical steps means the following: in each step a combined Pan+Zoom operation to the parent or particular child is performed.

the ContextualZoom can be performed first to show details in the model within their related context. Then, a HierarchicalZoom can be used to have a quick view of further details without context (zoom into a particular module and zoom out again). Another example is combining InteractiveZoom and ContextualZoom. When working on large models, often modelers prefer to use InteractiveZoom to zoom into a certain part of the model, and work with the ContextualZoom within this area.

Each interactive exploration scheme or a combination of them can be suitable for a specific modeling task. Hence, providing all schemes and their flexible combination is essential in order to get an efficient support for each individual modeling task. Especially a Focus+Context approach, such as the ContextualZoom, can be combined with other schemes (e. g., Pan+Zoom) in an effective way.

5.4.1. Overview+Detail

In addition to the interactive exploration schemes mentioned in the previous sections, an Overview+Detail scheme is provided. It facilitates the exploration of the modular model by using both, an overview and a detailed view. In this way, a view of the overall model structure, and focused information about specific submodules and details can be displayed simultaneously. Figure 5.15b shows the approach using a space-scale diagram whereas Figure 5.11 provides an example. These views and their different abstraction levels of the model are linked. For instance, when the modeler is interactively focusing on an element of interest in the local view, the same element is emphasized in the overview. In this way the modeler is able to contextualize the element in focus.

Overview+Detail also provides navigation through the network on a constant zoom level. As depicted in Figure 5.11, in VISUAL EXPLORER an overview window of the entire modular model appears in the lower-right corner and can be switched on or off. In the overview window a drag-and-drop box can be used to navigate the network in the detailed view.

The smooth integration of overview and detailed view is realized by using two cameras⁴. Each camera belongs to an independent layer which is superimposed, for example, by a transparent background (see Figure 5.12).

5.4.2. Comparison

In literature, there have several evaluations of the different interactive exploration schemes. Results have been revealed that the efficiency of each approach depends on several factors, in particular on the specific task and individual preferences. None of these approaches is suitable for all tasks, but rather for a specific group of task. For a recent discussion of advantages and disadvantages see the review of Cockburn et al. [39].

In VISUAL EXPLORER three interactive exploration schemes are provided. By using different exploration schemes, the provided contextual information varied widely when focusing on the same element or focus path. This is shown in Figure 5.13. Another example emphasizing a focus path is given in Appendix C, Figure C.1. The different exploration schemes can also be combined in a straightforward manner. For instance, the Overview+Detail scheme can be enhanced by features of the zooming interface. In this way, the modeler is able to decide which exploration scheme is most suitable for the individual task. Table 5.1 summarizes the features of the exploration schemes discussed in this chapter. Geometric transformations of Focus+Context interfaces as well as the semantic zooming of zooming interfaces is performed by the ‘Mapping’ step of the visualization pipeline. Geometric zooming and panning can be assigned to the ‘Rendering’ step (cf. Figure 3.6). All interactive exploration schemes provide the information visualization mantra: “Overview first, then zoom and filter, then details on demand” [176]. They support views with different granularity.

⁴PICCOLO2D provides a flexible model for using multiple cameras.

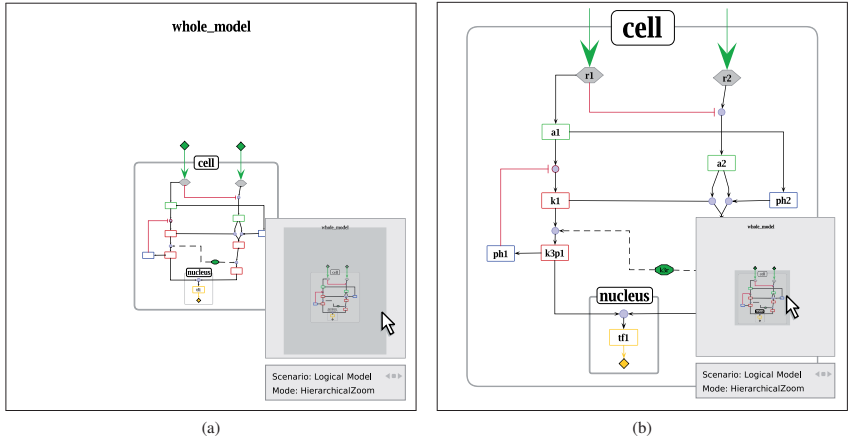


Figure 5.11.: Logical Toy model with Overview+Detail: (a) global view; (b) local view. By moving the transparent gray box in the overview, the network in the local view can be navigated. In this way, the hierarchical level of the view remains constant and can be used to navigate the network similar to a pan operation, but with additional contextual information.

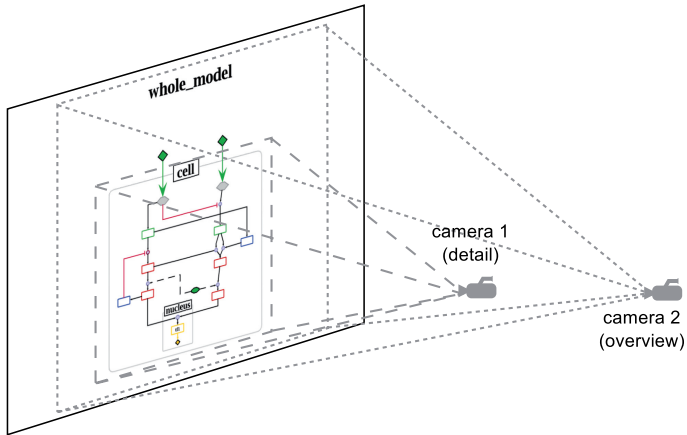


Figure 5.12.: Overview+Detail using two cameras. Camera 1 captures the detailed view (dashed line); camera 2 relates to the overview (dotted line). The rendered picture is shown in Figure 5.11b where the detailed view captures the full screen and the overview is provided in the lower-right corner.

5. A Visual Interface for Interactive Exploratory Model Analysis

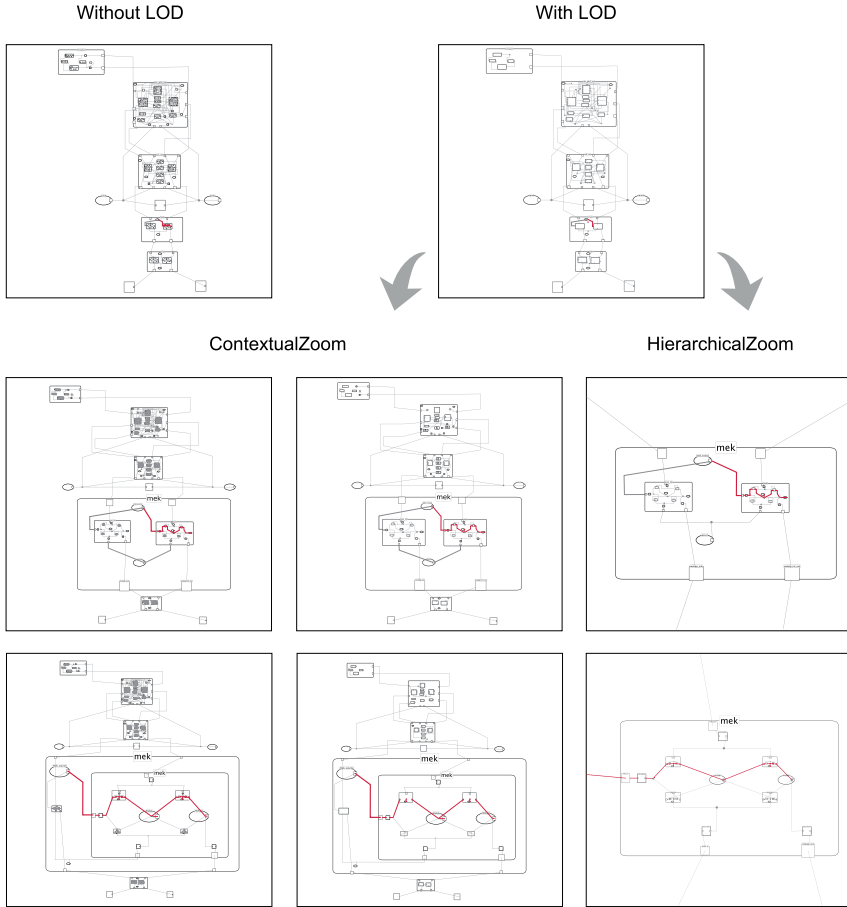


Figure 5.13.: HierarchicalZoom versus ContextualZoom applied to the modular EGF model. A focus path for visual correlation of distinct elements is used in order to emphasize double phosphorylation of the protein kinase mek. Columns show different schemes and options – ContextualZoom with disabled LOD (left), ContextualZoom with enabled LOD (middle), and HierarchicalZoom with enabled LOD (right). When using ContextualZoom the context is preserved, while when applying HierarchicalZoom it is not preserved. The left column has a higher visual complexity than the middle and right columns. Rows depict different zoom levels – from global view (top) to local view (bottom). The figure is inspired by Schaffer et al. [172].

<i>Feature/Scheme</i>	<i>InteractiveZoom</i>	<i>ContextualZoom</i>	<i>HierarchicalZoom</i>	<i>Overview+Detail</i>
Scale view	✓	.	✓	✓
Scale elements	.	✓	.	.
Content-aware	.	✓	✓	.
Topology-aware	.	✓	✓	.
Context-preserving	.	✓	.	.
Level of Detail	✓	✓	✓	✓
Multiple foci	.	✓	.	.
Uniform animations	✓	✓	.	✓
Non-uniform animations	✓	.	✓	✓

Table 5.1.: Features of the different interactive exploration schemes used in VISUAL EXPLORER. ‘✓’ – feature is supported; ‘.’ – feature is not supported.

5.5. Applications

In this section three applications are presented that use one or more of the interactive exploration schemes. First, it is shown how the modular model can be quickly and efficiently searched. Then the interactive exploration schemes are used to explore and visually correlate elements of interest in the context.

5.5.1. Search the Modular Model

In the first section of this chapter common tasks are listed. One of them is to search for a particular biological element or a group of entities (proteins, complexes, etc.) in the network. Usually, this is performed by searching for all entities (partially) matching a given property (e. g., represented by a string), and then listing or highlighting the result.

In modular models search operations and the representation of results need to be adapted to their complex structures. In VISUAL EXPLORER, for example, different properties of nodes and edges such as ‘hierarchical level’ or ‘type of element’ are searchable. Hence, requests like: “Give me all signaling processes that describing a specific level of detail.” or “Give me all (complex) entities of a specific type or instance.” can be answered. Additionally, the search process is aided by the visualization and navigation capabilities of VISUAL EXPLORER. As depicted in Figure 5.16, the modeler can show the results by highlighting them in the network context, by listing them in a separate window, or both. A mechanism for highlighting independent of the zoom level is especially useful to indicate results that are too small to be realized at the current zoom level.

In parallel, the HierarchicalZoom can be chosen to automatically zoom to the search result (e. g., navigation in hierarchical steps). Applying this scheme, the model structure can be experienced while zooming to the specific result using a smooth animation. In cases of multiple results, the list can be used to select a particular result and to focus on it using the HierarchicalZoom. In this way, all results can be explored in a convenient manner.

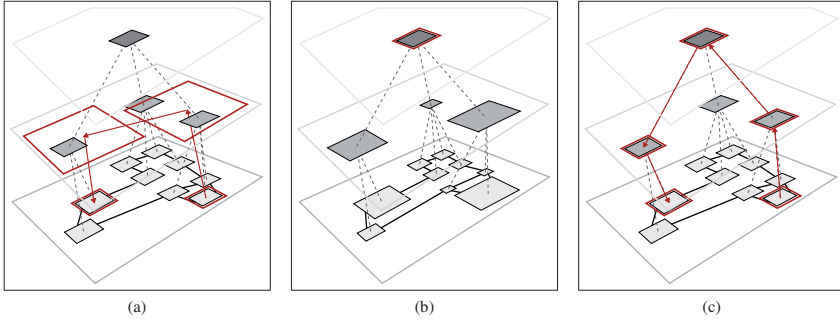


Figure 5.14.: Comparison of views of different exploration schemes using a 3D perspective. The original graph can be found in Figure 3.2e. (a) InteractiveZoom – only viewing window changes, separate Pan+Zoom, manual by the modeler – zoom out, pan, zoom in; (b) ContextualZoom – no changes of the viewing window, changes in size of elements; (c) HierarchicalZoom – only viewing window changes, Pan+Zoom simultaneously, semi-automatic, abstraction until both are in view (same parent), then specialization until target node is reached.

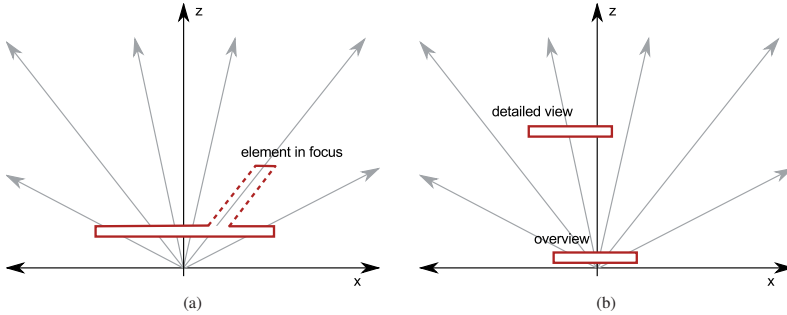


Figure 5.15.: Exploration schemes applied to space-scale diagrams: (a) ContextualZoom, (b) Overview+Detail. InteractiveZoom and HierarchicalZoom are depicted in Figure 5.10a and Figure 5.10c, respectively.

The advanced search tool of VISUAL EXPLORER uses rich visual cues for highlighting the results and, furthermore, applies the navigation and exploration capabilities of the tool. Thus, based on a comprehensive modular model, questions can be answered more efficiently.

5.5.2. Visual Correlation Using Multiple Foci and Contextual Visualization

For certain tasks the visual correlation of distinct biological entities will be of great value for modelers. In modular models, often these entities are located at different hierarchical levels and with a high spatial distance in the network. The direct visual correlation of entities can be achieved by the ContextualZoom with multiple foci. As depicted in Figure 5.17, the proteins *rgshc* (Shc) and *ras_gtp* (Ras) are located at different locations in the hierarchy. However, by applying the ContextualZoom both modules can be simultaneously displayed, and thus, visually correlated without losing much context. The proteins *rgshc* and *ras_gtp* are also the elements of the local views in the zoom sequence in Figure 5.5.

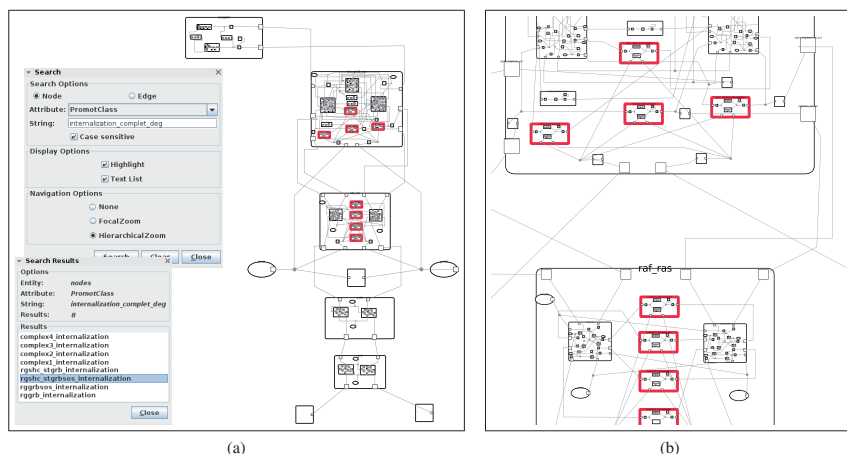


Figure 5.16.: Search the modular EGF model using highlighting and interactive exploration schemes. The current display options are 'Highlight' and 'Text List'; the interactive exploration scheme is 'HierarchicalZoom'. (a) Search dialog and results list, and search results in the network context. (b) Detailed view of the search results.

5.5.3. Topology-Aware Exploring of the Local Context

Sometimes it is very difficult to follow an edge through the hierarchical structure because the biological entities belonging to this edge are at different hierarchical levels or have a high spatial distance in the network layout. Some techniques exist, e.g., as described in Moscovich et al. [134]. In VISUAL EXPLORER different strategies are provided to explore the local context of an edge. The different interaction mechanisms are using the HierarchicalZoom and are described in the following list. For an illustration see also Figure 5.18.

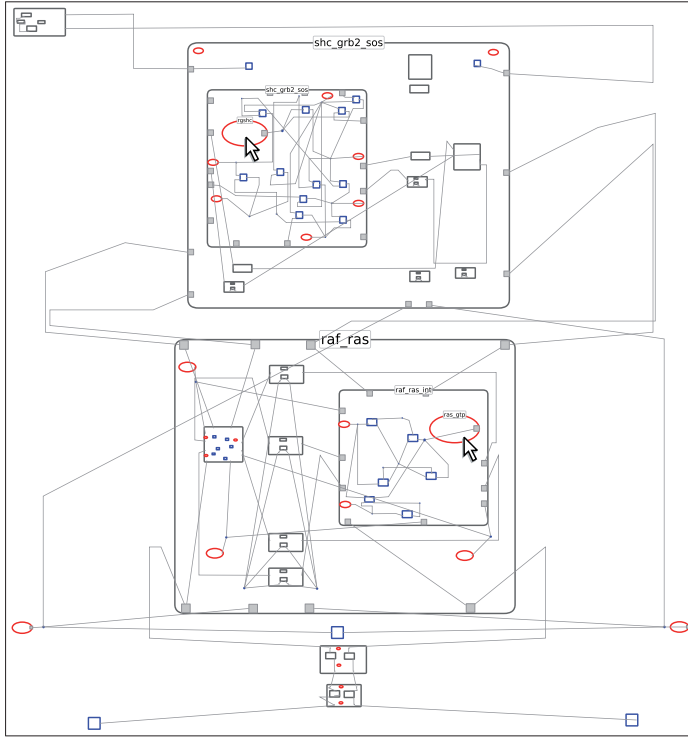


Figure 5.17.: ContextualZoom using the Focus+Context approach. Protein `rgshc` (Shc) and protein `ras_gtp` (Ras) are emphasized by increasing their size. All remaining elements are de-emphasized (decrease in size or hidden).

Zoom to Source. By clicking on a reaction edge defined between entities, this strategy automatically zooms from an arbitrary viewing window W_A to the source entity focused by a local view W_S . The source entity is displayed enlarged.

Zoom to Target. By clicking on a reaction edge defined between entities, this strategy automatically zooms from an arbitrary viewing window W_A to the target entity focused by a local view W_T . The target entity is displayed enlarged.

Zoom along Edge. By clicking on a reaction edge defined between entities, this strategy automatically zooms from an arbitrary viewing window W_A to the source entity rendered by W_S . This follows the movement of the viewing window along the path in edge direction until the target entity, rendered by W_T , is reached. The HierarchicalZoom introduces some intermediate viewing windows W_I .

For this purpose, animated ‘flights’ (moving the viewing window along the path) are applied. They also consider bend points of polylines. These methods do not preserve the context but are topology-aware by using the HierarchicalZoom and the route of the edge. They are performed in real-time and use the optimized version of the non-linear animation function already mentioned in Chapter 3, Section 3.3.4.1.

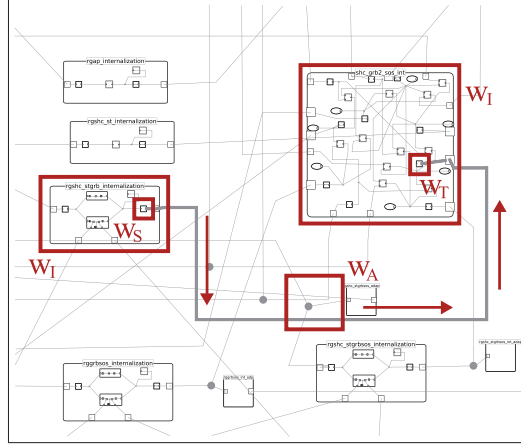


Figure 5.18.: Exploring the local context of an edge using the HierarchicalZoom with different options: zoom to source destination (from W_A over W_I to W_S), zoom to target destination (from W_A over W_I to W_T), and zoom along an edge (from W_S over W_I to W_T , based on the edge geometry including bend points).

5.6. Conclusions

Appropriate interaction mechanisms are an essential foundation for complex visual representations and their exploration. In this chapter different interactive exploration schemes are presented. Most of them are based on well-known navigation techniques used in other tools such as the Overview+Detail approach. But all of them are adapted to handle hierarchical structures – they facilitate interactive visual exploration of modular models specified in MDL. In this way, model parts can be visually presented in the context of the entire modular system. This can be provided by either spatial separation (Overview+Detail), temporal separation (InteractiveZoom, HierarchicalZoom) or integration (ContextualZoom). Moreover, the interactive exploration schemes can be combined to improve the navigation and exploration process and provide flexibility for multifaceted tasks in systems biology. For instance, an ContextualZoom can be applied to remain the related context. If the context is not necessary anymore, a HierarchicalZoom can be performed to have a quick view of the further details.

Applying the HierarchicalZoom, structural features of the modular model can be used to navigate on comprehensible paths through the hierarchical structure. This helps users to build up a mental map and, finally, ensures their orientation in complex structures of signaling models. A particular focus is on the

5. *A Visual Interface for Interactive Exploratory Model Analysis*

edges of the network. In signal transduction, they represent relations (e. g., interactions) and are often much more complex than in other domains.

All interactive exploration schemes are elaborated to work with complex structures of modular models and are integrated in VISUAL EXPLORER. Shortcuts and mouse controls for the provided exploration schemes are compiled in Appendix C, Table C.1.

All communication between the readers of an image and the makers of an image must now take place on a two-dimensional surface. Escaping this flatland is the essential task of envisioning information for all the interesting worlds (physical, biological, imaginary, human) that we seek to understand are inevitably and happily multivariate in nature. Not flatlands.

Edward R. Tufte [197]

CHAPTER 6

Visual Scenarios

The interactive exploration schemes described in the previous section modify either the scale of the view (i.e., zoom level) or the scale of individual elements. This results in different views and varying sizes of the geometric objects within these views. However, the interactive exploration schemes do not primarily define how geometric objects are rendered, for example, the color of the object. Nevertheless, the visual appearance of elements is certainly an essential prerequisite for encoding and interpreting different properties of the model. Thereby, in systems biology, visual properties often have pre-associated meanings, for instance, red or green is used to indicate under- and over-expressed elements in microarray data analysis.

According to the visualization pipeline (cf. Chapter 3, Figure 3.6), there are several essential steps for obtaining a desired visual representation: In the first, data must be analyzed, filtered and focused. These operations are often necessary to prepare the data for adequate visualization. The most important and critical step in this process is the mapping from *arbitrary datasets* (e.g., in the form of numerical values) to *visual properties* of the element such as node shape, node color or line width. After mapping, often an appropriate layout must be found. In practice, this can be performed, for example, by adjusting an existing layout. Finally, different user interactions must be defined. They describe in which way the modeler can interact with the data and its visual representation.

In this chapter *Visual Scenarios*¹ are introduced – a mechanism to encapsulate different visual representations, their mappings, adequate filters, the overall layout strategy and appropriate exploration schemes. First, the research problem is outlined. Then the general idea of visual scenarios is introduced. For illustrating this idea, two types of visual scenarios – predefined and user-defined – are introduced, and then several visual scenarios are described in more detail. Finally, some applications using different visual scenarios are presented.

6.1. Research Problem

A primary aim in visualization is to define an adequate visual representation for a particular modeling task. Some modeling tasks need a common or standardized representation such as in textbooks or as provided by SBGN. For instance, when a model has to be exchanged between different scientific groups,

¹ A scenario is a use case with a special intention, goal or set of tasks. This work deals with visual representations of models. Accordingly, visual use cases are called *Visual Scenarios*.

6. Visual Scenarios

standardized visualizations are essential for providing a common basis for maintaining the mental map and supporting a small cognitive load (cf. Chapter 3, Section 3.5.2.1).

Besides the trend towards standardized representations [114], visualizations must also be highly flexible because data characteristics and modeling tasks widely differ. Often the meaning of ‘adequate’ depends on the modeler and the scientific environment. Hence, visualization of network elements needs to be customizable to individual requirements and preferences [111]. For instance, some modelers want a particular color scheme applied to the network elements for highlighting a certain biological phenomena. Others desire a specific set of shapes for the different molecules depicted. Furthermore, modeling and analysis tasks change frequently. For instance, initially the modeler wants to emphasize nodes that meet a certain condition, for example, over-expressed elements in microarray data analysis. In a later step, the modeler becomes aware of some interesting elements and wants to explore the local network of one of these elements in context of the whole network. Therefore, complex visual settings for different modeling tasks should be switched between easily and with a direct visual feedback.

Typically, in the context of systems biology, visualization rules are defined as hard coded (i. e., in a programmatic manner). However, the definition of an adequate visual representation cannot be performed by the software engineer because only the modeler knows the appropriate visualization in the current context. However, the modeler has no or only a few options for direct interaction and manipulation. Further, most of the visualizations are designed for a restricted set of data using very specialized methods. It provides less visual flexibility regarding changes in the modeling task. Hence, a high flexibility in visualization is needed to allow the integration of new data coming from in-vivo, in-vitro and in-silico experiments.

In summary, the following list gives relevant requirements for a flexible visual representation of modular models (some are taken from Droste et al. [50]):

- VS1. *Simplicity*: Convenient and intuitive customization of visualization using GUI components.
- VS2. *Completeness*: Support of the (mostly) complete ‘visual property set’ provided by the particular tool. Additionally, other components affecting the visual representation such as filter options, layout and exploration strategies should be provided.
- VS3. *Flexibility*: Support of different mapping schemes for different types of data, for example, continuous, ordered, categorical (cf. Chapter 3, Section 3.3.3.2).
- VS4. *Abstraction*: Support of modular models and usage of different abstraction levels in order to hide complexity or show different aspects of datasets.
- VS5. *Responsiveness*: Simple and immediate switching between different visual settings in order to deal with frequently changing requirements.
- VS6. *Scope*: Support of pre-defined and user-defined visual settings.

Recently, two different approaches are presented that allow the modeler to access and directly manipulate the visual representation of elements in an interactive manner². This is realized either using a GUI or a scripting language.

The first approach is called VIZMAPPER, a feature of the CYTOSCAPE tool [174]. Here, the visualizations can be customized by several GUI components. The VIZMAPPER is a visual mapping system that controls the visual properties of network elements according to different datasets.

²The necessity of an interactive tool was already pointed out by Kumar et al. [111].

The second approach is introduced by Droste et al. [50, 51]. For the tool OMIX, a scripting language, the OMIX VISUALIZATION LANGUAGE (OVL) was developed in order to give the modeler highly flexible access to visual properties. An OVL script can be created by using a development environment that supports the modeler in scripting appropriate visualizations.

Both approaches allow the exchange of visual styles between different models. For that, the VIZMAPPER uses a special file with the definition of all visual styles; OMIX uses different OVL scripts. However, in spite of the great flexibility of a scripting language, it would be preferable to use a GUI-based approach. Usually, modelers are not well trained in scripting visual styles using a programming language, and there is no direct visual feedback for the visualizations to be created.

6.2. Concept

As stated in the previous section, one of the main motivations for this work is the fast and convenient switch between different visual configurations or settings. When the modeler is working on a modeling or analysis task very intensively over a long period, they usually create complex visual settings that support the efficient processing of the task. These settings are lost when the modeler changes the modeling task. Then this happens, the modeler has to create all of the visual settings again when continuing with the previous modeling task. An improved way would be to encapsulate all of the settings needed to reproduce the customized visual representation in one configuration and to call a single action to generate it.

Based on the approach of the VIZMAPPER, a new software component is developed within VISUAL EXPLORER. It introduces *Visual Scenarios (VSs)*, a mechanism for encapsulating visual configurations or settings for a customized visual representation. VSs are a way towards a more transparent definition of specific visual settings that can be modified by the modeler. A central component of the VS approach is a graphical interface for the flexible and intuitive handling of visual properties. Moreover, according to the visualization pipeline described in Chapter 3, Section 3.3.2, the visual appearance of a network can be influenced by additional components such as different filter options, a layout strategy or several exploration schemes applied by the modeler (Sections 6.2.2 and 6.2.3). Thus, a customized, visual representation may depend on several components. This can be seen as an extension to the VIZMAPPER approach. Moreover, the VS approach supports modular models, for example, by special handling of important visual properties like *scale* or *width* (Section 6.2.4), and two different scenario types which reflect the trade-off between standardization and customization (Section 6.2.5).

In this sense, VSs are an attempt to combine the different components in a convenient form. The central component of the VS approach is the flexible and intuitive handling of visual properties which is described next. Thereafter, further components of VSs are introduced.

6.2.1. Visual Mappings Using Transfer Functions

One of the most important components is the visual mapping of metadata to visual properties of the network. It refers to the ‘Mapping’ step in the visualization pipeline (cf. Chapter 3, Figure 3.6).

As illustrated in Figure 6.1, arbitrary metadata of the model (e.g., expression data, reaction type or compartment location) is mapped by specific *transfer functions*³ to visual properties of the graph element. For instance, the modeler needs to differentiate certain molecule types which are defined in the metadata. By using a *discrete transfer function* the modeler can define an appropriate border color for each type of molecule. In another example the modeler performs some analysis using the underlying graph and

³They are not related to the transfer functions used for distortions in graphical fisheye views or bifocal lenses described in Leung and Apperley [116].

6. Visual Scenarios

computes the degree (i. e., connectedness) for each species in the network. Using a *continuous transfer function* the modeler is able to map these connectedness values to the fill color of the respective species. The visual mapping component \mathbb{M} of a visual scenario is defined by a finite set of mappings M . A specific mapping $m \in M = \{P_D, tf, P_V\}$ consists of a set of metadata properties P_D ; a transfer functions tf with $tf \in TF = \{direct, discrete, continuous\}$, and a set of visual properties P_V . The mapping process from the metadata to visual properties is achieved by specific transfer functions. They are described in the following list (adapted from the VIZMAPPER):

Direct With the direct transfer function tf_{direct} the metadata are directly passed to the visual property of the graph element. This is useful for labeling nodes and edges. For example, the name or identifier of a molecule is directly visible as the text label of the related node (Figure 6.1a).

Discrete Using the discrete transfer function $tf_{discrete}$, the metadata are mapped to discrete values of the visual property. This works on the basis of a lookup table where for each piece of metadata g_i , a value v_i is stored. The related function is $v_i = f(g_i)$. For example, different molecule types form a set of groups. Each group is mapped to a discrete color of the node borders (Figure 6.1b).

Continuous Applying a continuous transfer function $tf_{continuous}$, numerical metadata can be related to visual properties that are continuous in nature such as a color gradient. The interpolation does not necessarily have to be uniform – it can also be non-uniform using interpolation points. For instance, the connectedness value can be mapped to node fill color using a red-blue gradient. (Figure 6.1c).

VISUAL EXPLORER supports the mapping to a wide variety of visual properties, which improves the flexibility and efficiency in encoding (requirements VS2 and VS3). For a comprehensive overview of supported visual properties of network elements, their related data types and feasible transfer functions, see Appendix B, Table B.2. For the relative effectiveness and efficiency of different visual properties see Chapter 3, Figure 3.8.

The visual settings are configurable and can be fine-tuned by the modeler via several GUI components. These components facilitate the selection of desired metadata, and the adjustment of the transfer functions and visual properties in an interactive and convenient way (requirements VS1 and VS5). So initially, a visual scenario (VS) is defined by $vs \in VS = \{\mathbb{M}\}$ with a visual mapping component \mathbb{M} . When the component \mathbb{M} is empty, a default mapping component is used.

6.2.2. Filter and Analysis

As described before, a VS includes the definition of a set of transfer functions for changing visual properties of the network. Along with these visual mappings also an appropriate filter mechanism, and necessary analysis on top of that filtered dataset can be defined. Both components, filter and analysis refer to the ‘Analysis’ and ‘Filtering’ steps in the visualization pipeline (cf. Chapter 3, Figure 3.6). They are performed on data operators which change the raw data and generate a new dataset [35]. Filter operations change the topology of the network, for example, through addition, removal or replacement of elements (Figure 6.2a). For instance, terminal nodes can be removed in order to have a clearer visual representation.

VSs can also have a component for analysis purposes where specific properties are computed before the actual visual mapping (using transfer functions) takes place. These necessary computations are based on the filtered data (Figure 6.2b). For instance, the connectedness value for each node in the network is computed and stored for further processing. Of course, this computation is required whenever the topology of the network is changed. Another example is the computation of the hierarchical level L_T

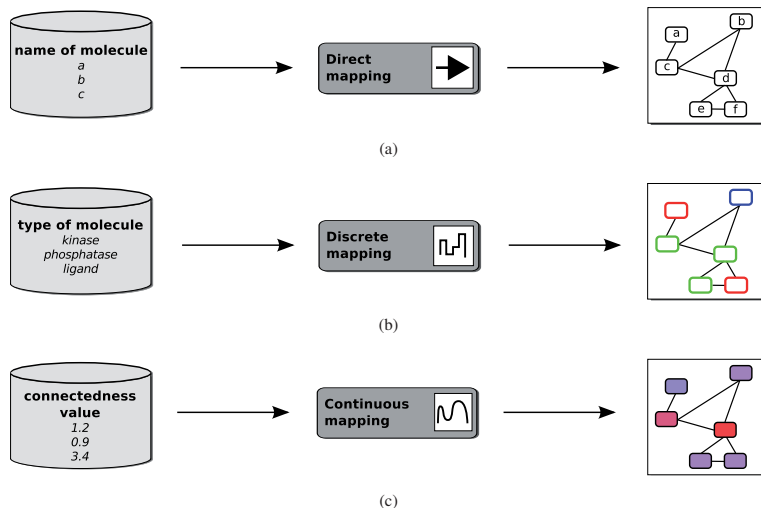


Figure 6.1.: Arbitrary metadata of the model (e. g., type of molecule) is mapped by a transfer function to the visual properties of the element (e. g., node border color). Thereby, different transfer functions can be used for several data types: (a) direct, (b) discrete, and (c) continuous transfer function.

of elements. Thereby, the values are computed only once because the structure does not change and the calculation is independent of user interactions. In contrast, the calculation of the distance of each element in the network to the current focus depends on user interactions. Hence a computation of the distance values is performed every time, a new element is focused. According to that, the former definition of a visual scenario (VS) can be extended by a filter component \mathbb{F} and an analysis component \mathbb{A} . This results in the following definition: $vs \in VS = \{\mathbb{F}, \mathbb{A}, \mathbb{M}\}$. \mathbb{F} and \mathbb{A} can be empty. These additional components fulfill the requirement VS2.

6.2.3. Layout and Exploration Strategy

In addition to the filter and analysis components, a layout strategy and a particular exploration scheme can be defined in a VS. Both components operate on the view data; that is, they do not change the underlying dataset [35]. For the layout component \mathbb{L} , in principle, all provided layout strategies can be used such as orthogonal layout or flat layout (Figure 6.2c). Furthermore, it is possible to define an interactive exploration scheme for a specific VS. This is handled using a separate component \mathbb{E} (Figure 6.2d). The modeler can choose from different interactive exploration schemes such as InteractiveZoom, ContextualZoom or HierarchicalZoom. Finally, a visual scenario (VS) is defined as a tuple of five components:

$$vs \in VS = \{\mathbb{F}, \mathbb{A}, \mathbb{M}, \mathbb{L}, \mathbb{E}\} \quad (6.1)$$

6. Visual Scenarios

The VS includes a filter component \mathbb{F} , an analysis component \mathbb{A} , a visual mapping component \mathbb{M} , a layout component \mathbb{L} , and a component for exploration themes \mathbb{E} . Additional to \mathbb{F} and \mathbb{A} , also \mathbb{L} and \mathbb{E} could be empty. For instance, a layout could take a long time depending on the network complexity and, therefore, the layout component in the VS definition can be defined as empty.

The components are ordered and can be applied according to the visualization pipeline; that is, first, filter and analysis operations, then changing of visual properties, and finally layout methods. Of course, all provided layout methods and exploration schemes (e. g., described in Chapter 5, Section 5.2) can be used separately via a GUI component.

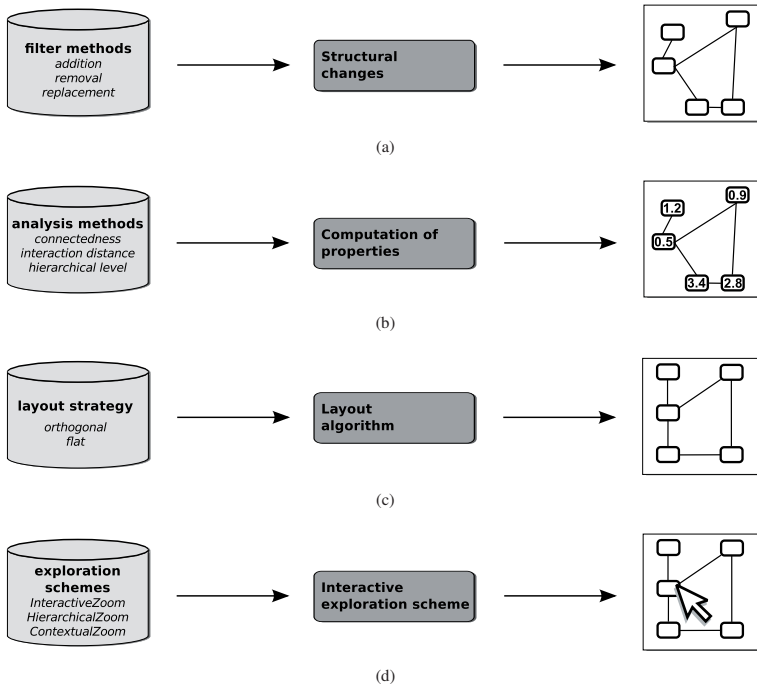


Figure 6.2.: Different components are used to visually alter the network. (a) Filter operations can be applied resulting in a different network topology. (b) Specific properties can be computed before the actual visual mapping, using transfer functions takes place. (c) Provided layout methods can be used to alter the positions of network elements. (d) Exploration schemes such as ContextualZoom or HierarchicalZoom can be chosen. (a) and (b) operate on the dataset whereas (c) and (d) perform on the view.

6.2.4. Visual Mapping in Modular Models

In the context of modular models the mapping of metadata to visual properties is of high importance because network elements and hence their visual properties are located in a hierarchical and nested structure. In the following, some visual properties and their hierarchical handling are described (requirement VS4).

Scale In VISUAL EXPLORER scaling is a central method. As mentioned in Chapter 4, Section 4.2, nodes and edges are scaled in relation to their location in the hierarchy. For that, the same, default scale factor for each level is used. This represents the inherent scaling within the view of the modular model. A new visual property ‘scale’ is introduced that can be accessed and managed by VSs. Hence, the size of an element in the hierarchy can be modified by that property. The scale property is applied recursively. Thus, it defines not only the size of a certain element but also the size of all its subelements. When mapping a dataset to the dimension of elements, it is advantageous to use the property ‘scale’ instead of the property ‘size’ because the property ‘scale’ is independent of the hierarchical level of an element.

Width According to the scale of the complete element, also the thickness of edges or node borders is scaled in order to prevent unwanted occlusions. However, in hierarchical views this is not adequate because lines with a broad line width located on a high level in the hierarchy can occlude elements completely on lower hierarchical levels. See Figure 6.4a for an example. In order to ensure the visibility of all entities without occlusion, lines are always rendered with a default constant width, in all hierarchical levels and independent of the zoom level⁴. The general concept is illustrated in Figure 6.3. An example with fixed line width is given in Figure 6.4b. To emphasize different quantities encoded in the visual property ‘line width’, VISUAL EXPLORER provides multiples of the default constant line width.

Transparency There are different ways for handling transparency in hierarchical structures. In VISUAL EXPLORER transparency of an element is propagated to all of its submodules. That means the transparency of an element cannot be set by an absolute value, but is rather a combination of all transparency values of its parent modules – it is accumulated throughout the hierarchical levels. Such handling simplifies the setting of transparency in hierarchical structures.

6.2.5. Pre- and User-Defined Visual Scenarios

In order to cope with the trade-off between standardization and customization, *predefined* and *user-defined* visual scenarios are introduced. Predefined VSs, hereafter referred to as *global VSs*, are standardized configurations or settings that produce similar visualizations in similar semantic situations. They ensure homogeneous views for common modeling and analysis tasks and also provide assistance for the inexperienced modeler. As described in Chapter 3, Section 3.5.2.1, they can be used to have nearly the same visual starting point for the same task. From a particular, standardized view the modeler can perform user-specific tasks. They are useful to build a mental map. Using global VSs the modeler can save time and effort required to become familiarized with the visualization. Thus, the modeler has more time to decipher the information of interest. In order to ensure availability and integrity, global VSs are predefined in VISUAL EXPLORER and are not editable by the modeler.

However, for some modeling or analysis tasks the predefined or default visualization is not applicable or not appropriate. For these cases a global VS can be further refined. Since global VSs are not editable,

⁴Working with fixed line width is also common practice in software for computer-aided design.

6. Visual Scenarios

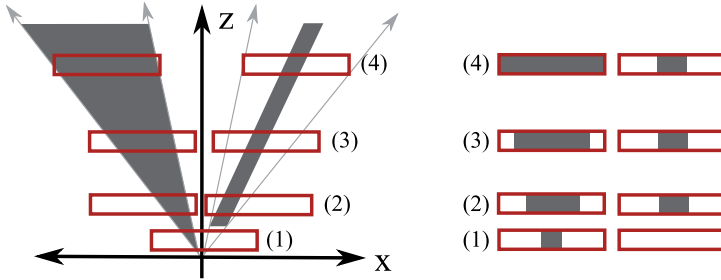


Figure 6.3.: In VISUAL EXPLORER line width and border width are independent of the zooming level of the viewing window (cf. Chapter 3, Figure 3.12). While zooming in, the width of lines and borders remain constant. The viewing window is colored in red.

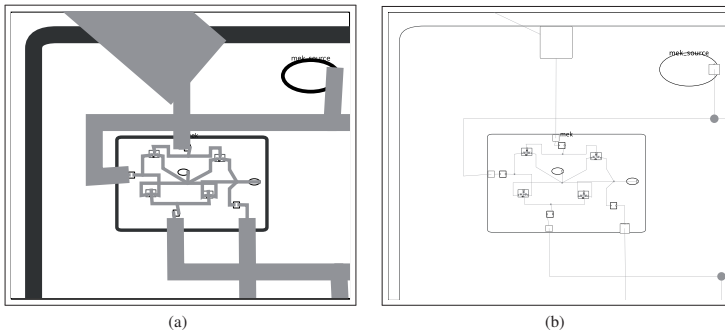


Figure 6.4.: Handling of line and border width in modular models. (a) Lines are scaled the same way as modules in the hierarchy. Hence, the line width is not constant and occlusions may occur. (b) In VISUAL EXPLORER the line width is kept fixed throughout the hierarchical levels.

the refinement can only be performed by duplicating the VS and modifying the instance as desired. Scenarios that are instantiated from global VS are called user-defined VS, hereafter referred to as *local* VSs. In contrast to global VSs, local VSs can be defined and modified by the modeler. For instance, each of the predefined transfer functions can be changed in order to customize the visual representation to the current modeling or analysis task. For that, all three transfer functions (direct, discrete and continuous) and the complete set of visual properties can be used. The separation into global and local VS fulfills the requirement VS6.

6.3. Specific Visual Scenarios

This section is concerned with specific global VSs developed in this thesis and adjusted for use with modular models. First, VSs for logical and dynamic models are covered. Then, three different examples for depicting structural properties of the modular model are introduced. Furthermore, how local VSs can be defined, is discussed.

6.3.1. Visual Scenarios ‘Dynamic Model’ and ‘Logical Model’

The aim of the VS ‘Dynamic Model’ and the VS ‘Logical Model’ is to apply an adequate visual representation to dynamic and logical models, respectively.

The VS ‘Dynamic Model’ is dedicated to dynamic models. The main visual features of this VS are described in Chapter 4, Section 4.3. It is defined by $vs \in VS = \{\mathbb{M}, \mathbb{E}\}$ whereas component \mathbb{M} is described in Chapter 4, Sections 4.3.1 and 4.3.2, and component \mathbb{E} is defined as HierarchicalZoom. Components \mathbb{F} , \mathbb{A} and \mathbb{L} are not defined.

As mentioned in Chapter 4, no standard for the visual representation of logical models exists. Hence, an intensively used global VS is the VS ‘Logical Model’. It serves as a common starting point when modeling with the logical formalism. The main features of this VS are described in Chapter 4, Section 4.4. By encapsulating all necessary transformation steps and visual mappings, it ensures the formalized visual representation of logical models in VISUAL EXPLORER. Its visual representation is defined by $vs \in VS = \{\mathbb{F}, \mathbb{M}, \mathbb{E}, \mathbb{L}\}$ whereas the component \mathbb{F} is described in this section, the component \mathbb{M} is described in Chapter 4, Sections 4.4.1 and 4.4.2, the component \mathbb{L} is defined as orthogonal layout (optional), and the component \mathbb{E} is set to the HierarchicalZoom. The component \mathbb{A} is not defined. In the current implementation, the VS ‘Logical Model’ is the only VS that uses a filter component \mathbb{F} .

It is important to note that the VS ‘Dynamic Model’ and the VS ‘Logical Model’ are independent of a specific model. They only depend on the model type ‘Dynamic Model’ and ‘Logical Model’, respectively.

6.3.2. Visual Scenario ‘Hierarchical Level’

When using modular models the hierarchical level of a certain module is an important property. It describes the hierarchical distance – the depth of the element in the hierarchical structure⁵. In systems biology, the hierarchical level can be used to characterize the level of abstraction of the biological knowledge. As described in Frisch et al. [60], when navigating through a modular model, it is often difficult to determine which is the hierarchical level currently depicted⁶. A clear allocation of network elements according to their hierarchical level is missing. This can be solved by visual cues for depth in a 2D network. Thereby, the hierarchical levels of such network elements are encoded by different colors. As shown in Figure 6.5a, clear allocation of elements can be established by encoding them with a unique background color according to their hierarchical level. By interpolating between colors (i. e., by creating a color gradient) hierarchical levels can be distinguished very easily. This can imply depth cueing and is beneficial for allowing a quick navigation to higher or lower levels. This kind of visual feedback is of special importance in modular models. In these models often the module borders are clipped, just showing a portion of the network, for instance, a local view generated by an InteractiveZoom operation (Figure 6.5b). Then the inclusion relations between different modules are not clear. With color cod-

⁵As described in Chapter 3, Section 3.1.2 and Chapter 4, Section 4.2 the hierarchical level L_T of a module is defined by the length of the shortest path between the root module and the module itself.

⁶Module size is not necessarily an indicator for the hierarchical level.

6. Visual Scenarios

ing it is apparent how these modules relate to each other, for example, two reactions are on the same hierarchical level or have a parent-child relation.

In some use cases also the hierarchical level of edges is important. For instance, in Figure 6.5b edges are clipped; that is, their end nodes are not visible. Then without having a visual indication it is nearly impossible to determine or correlate the hierarchical level of these edges. Within the modular structure the end nodes of an edge may lie on different hierarchical levels. Therefore, first the conjoint parent of the end nodes is computed. Hereafter, the hierarchical level of this parent is used to define the hierarchical level of the corresponding edge⁷. Then these values are mapped to the color gradient in order to indicate the hierarchical level of edges.

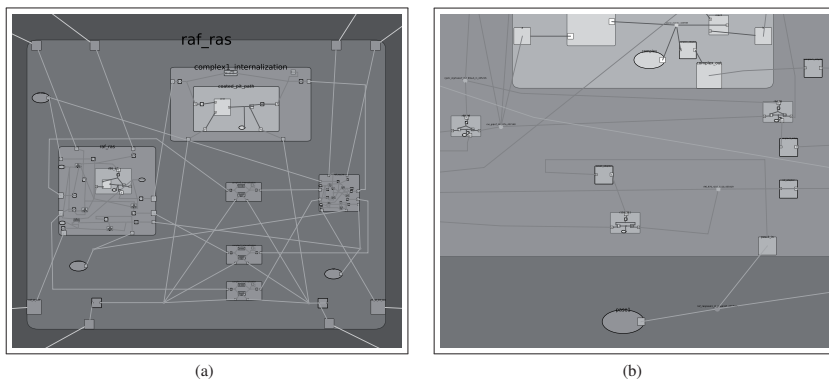


Figure 6.5.: The hierarchical levels in the network are encoded using a gray color gradient. For modules low saturation means low hierarchical level, high saturation means high hierarchical level; for edges the gradient is reversed. (a) A view of the composite module `raf_ras`. (b) A local view of (a). Although the modules and links are clipped, the association to a specific hierarchical level is straightforward.

Visual encoding of hierarchical levels is successfully used also in other applications. These applications include TREEMAPS [175] or diagramming software such as TOMSAWYER [193] or YED [205, 211]. However, none of the applications is dedicated to the domain of systems biology and visually encodes the hierarchical level of edges. However, the encoding of hierarchical levels should be displayed temporally in order to be oriented within the hierarchical structure; it includes a lot of different color cues and reduces the overall clarity of the visualization.

All the functionality described above is defined in the VS ‘Hierarchical Level’. It defines a component \mathbb{A} by calculating the value of the hierarchical level of each network element, a component \mathbb{M} as described above, and a component \mathbb{E} as HierarchicalZoom. The components \mathbb{F} and \mathbb{L} are not defined.

⁷Note, the conjoint parent must not necessarily be the parent of one of the end nodes.

6.3.3. Visual Scenario ‘Connectivity’

A classical property in graph theory is the *degree* of a network element, which was already discussed in Chapter 3, Section 3.1.2. In cellular networks, this property is important and can be used to characterize the *degree of connection* or the *connectedness* of biological entities. Highly connected entities can be biologically interpreted as *hubs*. In the majority of cases, these hubs belong to the core functions and are essential for the survival of the cell [58, 90].

Each node is ranked by a *connectedness value* k . In dynamical models the directionality information is not given in an explicit form. In a trivial case, k is equal to the number of adjacent nodes. In logical models with directionality information, a trivial solution for the computation of k would be to define it as the sum of the number of adjacent nodes that are source nodes k_{in} and target nodes k_{out} , as stated in Equation 6.2.

$$k = k_{in} + k_{out} \quad (6.2)$$

However, this generalized form is not appropriate for modular models, because in modular models, defined by MDL, several nodes require special treatment – they have to be filtered. For instance, in dynamic models specific network elements such as of the class `reaction`, `adapter`, `terminal` should not be considered in the computation. A more sophisticated computation should also apply to logical models. Network elements such as of the class `terminal` should not be considered.

Moreover, link nodes as part of links in dynamic models and gates (e.g., `and` or `or`) in logical models, can split or aggregate ‘reactions’ and are not source or target nodes in a biological sense. Often, these structures result in complex paths across different hierarchical levels. Thus, they have to be evaluated accordingly. In this way, connectedness is defined as a special variant of the property ‘degree’.

The value for the connectedness of a network element is computed as follows: Starting from the particular element, all adjacent edges are computed. For each edge the adjacent node is computed. If the adjacent node is an invalid element, this procedure is repeated until a valid network element is found. The overall algorithm is repeated for all elements in the network that need a connectedness value k . Network elements not considered in the computation (invalid network elements) are listed in Appendix B, Table B.5.

After computation⁸, the values of the connectedness are mapped to a color gradient. By default, the most highly connected modules (with a high connectedness value) are encoded in red; unconnected modules are encoded in blue. An example is shown in Figure 6.12c. From a perceptual point of view, the highly connected modules in red are more emphasized than modules in blue.

The computation and highlighting of the connectedness values is performed by the VS ‘Connectivity’. It is defined by the component `A` calculating the connectivity value of each network element, the component `M` as described above and the component `E` defined as `HierarchicalZoom`. The components `F` and `L` are not defined.

6.3.4. Visual Scenario ‘Interaction Distance’

Similar to the two previous visual scenarios, the VS ‘Interaction Distance’ aims at emphasizing structural properties of the network or its elements. It focuses on the neighborhood and interdependencies of a specific element. The local network around an element of interest is emphasized and at the same time this information is embedded in the context of the complete network. In this way, the modeler only pays attention to the neighborhood of a specific network hub [200]. This property was already mentioned in Chapter 3, Section 3.1.2.

⁸The computation is performed only once because the structure of the modular model does not change.

Sometimes this type of distance is also referred to as *semantic distance* since it does not represent the geometric distance between two elements, but rather the semantic relation based on their proximity. For instance, two elements can have a high geometric or spatial distance in the network layout, but have a low interaction distance because both elements are directly connected. From a semantic point of view, they are very close.

The computation of the distance values, and handling of visual properties and interaction patterns is defined by the VS ‘Interaction Distance’. The VS is defined by the component \mathbb{A} that computes the distance values for the network elements as described in Section 6.3.4.1, and the component \mathbb{M} as described in Section 6.3.4.2. The components \mathbb{F} , \mathbb{L} and \mathbb{E} are not defined.

In the next subsections, the VS ‘Interaction Distance’ is discussed, in particular with regard to the computation of the values for the interaction distance, the visualization and issues related to modular models.

6.3.4.1. Computation of the Values Defining the Interaction Distance

Computing a local network and visualizing it is straightforward and provided by some tools, for instance, described in Toyoda et al. [195]. However, when dealing with modular models, the algorithm must consider the modular and hierarchical structure.

The computation of the local network of relationships is based on two input parameters: the focus module m_{focus} which is the element of interest and the interaction distance d_{max} defining the n -th degree of the neighborhood. Both parameters can be adjusted by the modeler, whereas m_{focus} can be selected interactively. The output is a local network (neighborhood) with the given distance d_{max} . The parameter d is used to assign the current computed interaction distances to each element.

As already mentioned, for the computation of the local network, the modular and hierarchical structure as well as domain-specific characteristics must be considered. For dynamic models, for example, link nodes, and elements of the classes *reaction*, *adapter*, *help* and *terminal* are considered as not valid for computing the interaction distance. This also applies to logical models where gates (e. g., *and* or *somehow*) are not required for measuring the interaction distance of elements. These aforementioned elements and further elements, compiled in Appendix B, Table B.5, are not relevant in the context of signaling networks.

By definition, m_{focus} has an interaction distance of $d = 0$. Each node that is directly connected to m_{focus} and considered as valid, gets a value for the interaction distance of $d = 1$ assigned; that is, the interaction distance will be increased by one. The same applies to edges: for adjacent edges of m_{focus} an interaction distance value of $d = 1$ is assigned. Then each of the adjacent nodes is analyzed and the interaction distance d of its direct edges and nodes is set. The network structure is recursively traversed until a valid elementary module is reached.

The procedure is illustrated in Figure 6.6 applying the interaction distance of $d_{max} = 2$. The focused node m_{focus} has an interaction distance of $d = 0$. All adjacent modules are computed from that node. Thereby, the hierarchical structure (in particular composite modules and terminal nodes) is considered. The computation results in a reaction module with interaction distance of $d = 1$ within a composite module. Now, the next adjacent modules are computed. The adjacent element is a link node (depicted by a circle). This type is not considered in the computation. Thus, the next adjacent modules are determined. Since the link node has two adjacent edges, both modules are considered – one located at the same hierarchical level; another located one hierarchical level below, within a composite module. Both modules have an interaction distance of $d = 2$ assigned. All remaining modules with $d > d_{max}$ (in this case: $d > 2$) are treated equally and are given a value of $d = -1$. Then the procedure is finished.

The general assignment is given by the piecewise function in Equation 6.3 where $d \in \mathbb{N}_0 \cup \{-1\}$ and $d_{max} \in \mathbb{N}$.

$$f(d) = \begin{cases} d & \text{if } 0 \leq d \leq d_{max} \\ -1 & \text{if } d > d_{max} \end{cases} \quad (6.3)$$

In Figure 6.7, the TCR network with different values for the interaction distance is presented. For instance, the modeler is interested in a protein and its interdependencies and adjacent interaction partners. As clearly seen in Figure 6.7a, the adjacent proteins are not in the geometric neighborhood, but often have a high spatial distance in the network. In Figure 6.7b, the activation of kinase `jnk` and its link to a direct output is shown.

Allowing the modeler to select an additional element provides additional options, for example, to remove old subnetworks from the view or to add all neighbors to the current view. Another option would be to calculate the shortest path⁹ between the former and current focus element and add all elements on that path (and their local neighborhood) to the current view.

6.3.4.2. Visualizing the Neighborhood

The resulting local network is highlighted by a color; that is, the interval $[0..d_{max}]$ is mapped to a color gradient using a continuous transfer function. By default, the color gradient ranges from dark blue for $d = 0$ to light blue for $d = d_{max}$. In other words, the saturation of the color is reduced (faded out) with every iteration. In order to improve the visual appearance, additional visual properties are modified within the VS. In particular, these are *edge width*, *node border width* and *transparency*. Network elements with a low value (i.e., strong relation) are emphasized by a high width and a low transparency. All elements with $d > d_{max}$ (i.e., the network context) are de-emphasized by rendering in light-gray. Additionally, they are rendered with small line width and in a semi-transparent fashion. An example is presented in Figure 6.8.

From a perceptual point of view, the combination of these different mappings makes sense in order to use stronger visual cues for emphasizing elements of interest and the less effective cues for de-emphasizing elements out of scope. In this way, by selecting a certain module, the local interaction network is depicted very prominently in the context of the complete network.

6.3.4.3. Considering Hierarchy

When focusing on a specific element in the network, adjacent elements must not be located at the same hierarchical level; they can be placed in another module linked through terminal interfaces. However, these relationships should be depicted according to the interaction distance. Therefore, the modeler can choose from different options: (i) show only path elements defined either in the same module or in parent modules, (ii) show only path elements defined either in the same module or in submodules, or (iii) show all path elements independently of any inclusion relation.

In Figure 6.8a, the first option is depicted where all relationships are considered except those in parent modules. In contrast, in Figure 6.8b the third option is used to visualize structures independent of submodules and parent modules.

⁹For this scenario the graph property ‘path’ discussed in Chapter 3, Section 3.1.2 is used.

6. Visual Scenarios

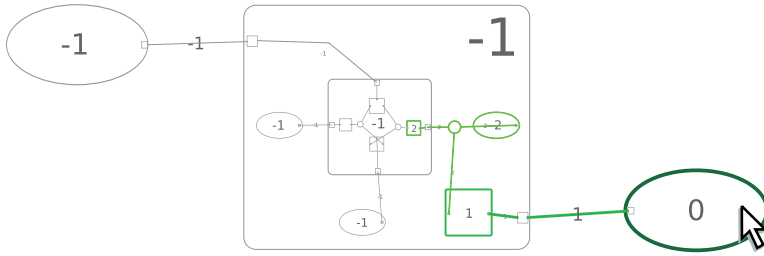


Figure 6.6.: Computation of the interaction distance $d_{max} = 2$ with all intermediate steps. The process is illustrated using the network example from Figure 2.5. For the sake of clarity, the interaction distance of each node and edge is given as label text. A negative 1 (-1) shown in the label means $d > 2$. A green gradient is used for emphasizing the local network.

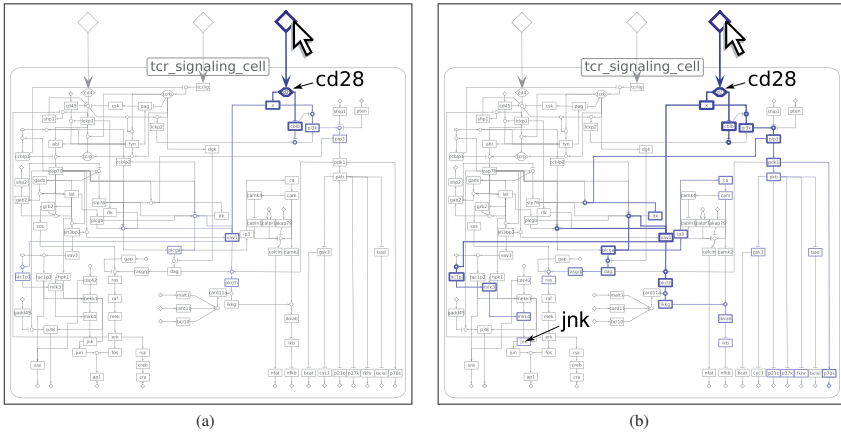


Figure 6.7.: Different values for the interaction distance d_{max} applied to the TCR model. The input element of receptor molecule `cd28` is selected as the entity of interest by the modeler (element under the mouse cursor). (a) $d_{max} = 4$, (b) $d_{max} = 7$. The figure shows how the local network is evolved.

6.3.4.4. Consider Directionality of Interactions

There also exist the possibility to consider the directionality of interactions in the computation of the interaction distance¹⁰. In this way, the logical and functional order of cellular processes, for instance, upstream or downstream processes in cell signaling, can be depicted. An example is given in Figure 6.9.

¹⁰This is based on directed edges and only applicable for logical models.

Considering the directionality of interactions, the highlighted network can be biologically interpreted as a flow of reaction information downstream; it shows the effects of an element in focus on other elements in the network.

6.3.4.5. Further Options

In Figure 6.10, different further options for handling hierarchical relations are illustrated. One option is handling multiple foci. This can be performed by defining additional elements as focused nodes. For each element the local network with the current interaction distance is calculated. As shown in Figure 6.10a this gives separated or connected local networks of as many foci as defined by the modeler. In this way, a local network can be expanded by defining, for example, a peripheral node as an additional focus node in order to analyze network-wide interdependencies.

Another option is to use the shortest path between elements in the network. As shown in Figure 6.10b the modeler can define two elements and the shortest path between them can be computed automatically¹¹. All nodes within this shortest path are used as focused elements. From each of the focused elements a local network is computed using the defined interaction distance d_{max} . In this way, several local networks can be emphasized within the entire network and the structural neighborhood can be explored effectively and efficiently. All parameters for the different options can be configured by the modeler through a GUI component. Related shortcuts are compiled in Appendix C, Table C.3.

6.3.5. Defining Local Visual Scenarios

Local VSs can be derived either from global VSs or from other existing local VSs, or can be defined from scratch (using a default set of settings). In order to customize a visual representation, typically, only a subset of the settings of a global VSs is altered. For this, the global VS is copied, the particular mapping is changed and saved as a new local VS. For instance, in Figure 6.12d, a global VS depicts the interaction distance using a blue gradient. In contrast, in Figure 6.11b, a local VS derived from the global VS ‘Interaction Distance’ is shown. For this purpose, the global VS is copied, the gradient is changed to a red color and then saved as a new local VS.

Another example is given in Chapter 4, Figure 4.15 where the global VS ‘Logical Model’ for the representation of logical models is modified towards a more biological meaning. In particular, the visual property ‘edge line type’ is used to change the edge appearance from polylines to curved lines. Although the global VS ‘Logical Model’ represents a single, very specialized visualization method defined as a global VS, it can be instantiated as a local VS and further modified. In this way, the visual properties of the elements can be easily altered using the approach introduced. Thus, the generation of alternative representations matching different requirements or individual preferences is straightforward.

Several examples for customized local VSs can be found in Section 6.4 or in Appendix B, Figures B.6 and B.7. As described already in Section 6.2, the process of customization is supported by comprehensible and easy-to-use GUI components.

6.4. Applications

For the illustration of the applicability of VSs, in Figure 6.11, two visual scenarios of the EGF model and, in Figure 6.12, several scenarios of the TCR model are shown. In particular, in Figure 6.11b and in Figure 6.12d several VSs emphasizing the interaction distance are depicted. In Figure 6.12, a possible use

¹¹The shortest path between two network elements is computed using Dijkstra’s algorithm [47]. Depending on the model type the network is considered either as a directed or undirected graph.

6. Visual Scenarios

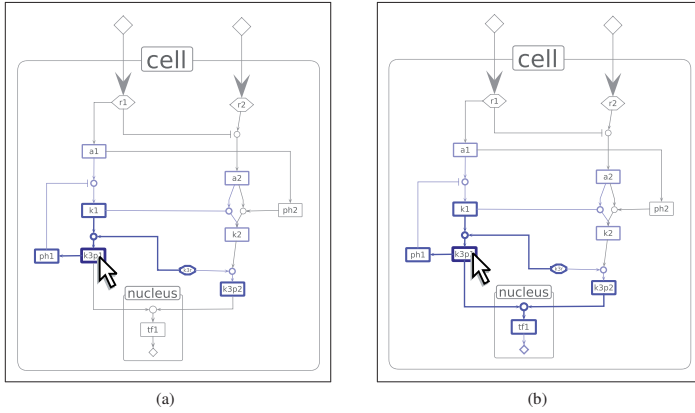


Figure 6.8.: Interaction distance with $d_{max} = 2$ and focus on $k3p1$. Different options for handling hierarchical relations are shown. (a) Only relationships in the same module or in parent modules are considered. (b) All relationships are considered, also those in submodules.

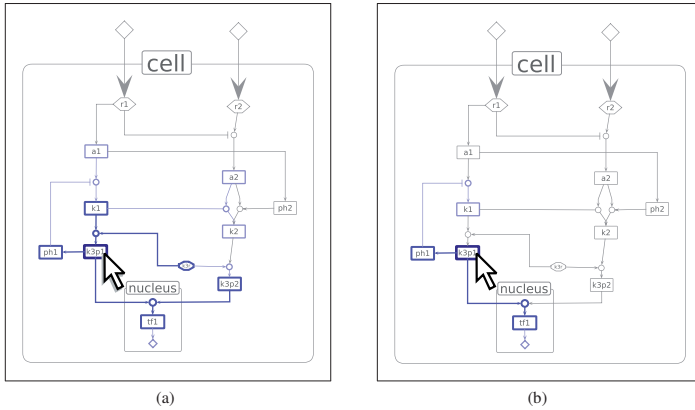


Figure 6.9.: Interaction distance with $d_{max} = 2$ with focus on kinase $k3p1$. There are different options for directionality of interactions: (a) direction is not considered, (b) direction is considered; for example, the upstream event to the adapter protein $a1$ is not part of the emphasized network.

case is shown where the visual representation of the logical TCR network is adapted to different modeling and analysis tasks. First, the network is visualized using metadata such as type of protein or interaction, both are encoded by color (Figure 6.12b). After studying the topology of the network, the modeler may be interested in the connectedness of all proteins in the network in order to find hubs. For that, the modeler switches to the VS ‘Connectivity’. Hubs are immediately emphasized in red (Figure 6.12c). Now, the modeler is interested in the local network of a specific highly connected entity. For that, the modeler switches to the VS ‘Interaction Distance’ and selects the hub entity of interest. All adjacent molecules within a given interaction radius are emphasized (Figure 6.12d). For such a diverse modeling procedure, fast switching between VSs is essential.

Another use case is shown in Figure 6.11 where different VSs of the EGF model are used to encode different distance metrics; in this case, hierarchical level and interaction distance.

6.4.1. Overlay Data in Context of Networks

As mentioned in Chapter 2, Section 2.2.1, new data can be generated from in-vivo and in-vitro experiments (e. g., expression data, protein identifier, subcellular localization, function from ontologies), or from in-silico experiments (e. g., structural, functional or dynamic properties) in order to validate hypotheses and predictions.

In the beginning, the model must be enriched with external data to describe properties of the model. In most cases, the interpretation and analysis of these datasets is not trivial. Hence, an adequate visualization of externally generated data in their related context is essential. A potential strategy is to overlay the data on top of the complex network for their interpretation because the modeler is often familiar with the network. In Saraiya et al. [169] it was evaluated that modelers have problems to derive meaningful insights from the datasets without the network context. Thus, the overlay data should be interactively adjusted in order to obtain an adequate visual feedback. This may support the generation of new hypotheses and knowledge.

Several research projects were published where the mapping of omics data or time-series data is discussed in general [26, 168]. There are different options for overlaying data in the context of the network. Some approaches exist, which annotate network elements with numbers (e. g., in CELLNET-ANALYZER), or subgraphics like complex glyphs or small charts for multi-dimensional data (e. g., in VANTED, GSCOPE, CYTOSCAPE, OMIX or VISANT). The benefits are obvious: the original network element can be preserved and the additional space for each element can be used for multi-dimensional data. However, this approach has some drawbacks: Due to the amount of information being visualized simultaneously the visual density is increased. This can also lead to poorer results in the interpretation of topology-related features of the model [168]. Furthermore, the network element as well as the complex subgraphics must be integrated. Thus, interrelations between the element and the subgraphics are difficult to identify [50].

Consequently, the generated data must be integrated directly with the properties of elements (which preserve the mental map of the modeler). In order to achieve this, a VS of choice is instantiated to a local VS and then combined with external analysis data, for example, visual integration of a logical model with external data. In this way, the presentation of analysis results will be more intuitive and flexible. For instance, it is feasible to visually group identical or similar data (e. g., to depict possible correlations between proteins or functional groups) to emphasize non-trivial or unexpected data (e. g., activation of a certain protein that was not expected) and to de-emphasize data that are not relevant in the context of a specific analysis (e. g., hide proteins that are not involved in a certain pathway). For instance, the state of each protein (e. g., active or not active) after a certain simulation experiment can be mapped to the color of the nodes in the network. In the following sections different use cases are presented.

6. Visual Scenarios

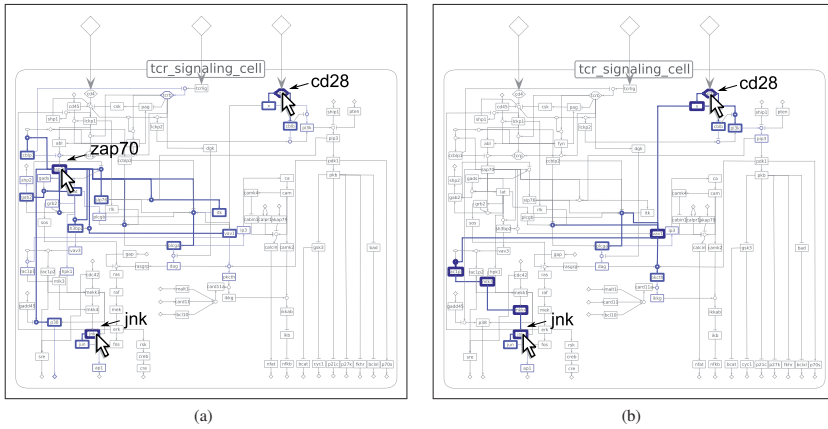


Figure 6.10.: Interaction distance with $d_{max} = 2$ showing different options for handling multiple foci. (a) Multiple foci on receptor molecule *cd28*, protein *zap70* and protein *jnk*. (b) Shortest path from the receptor molecule *cd28* to the kinase *jnk*.

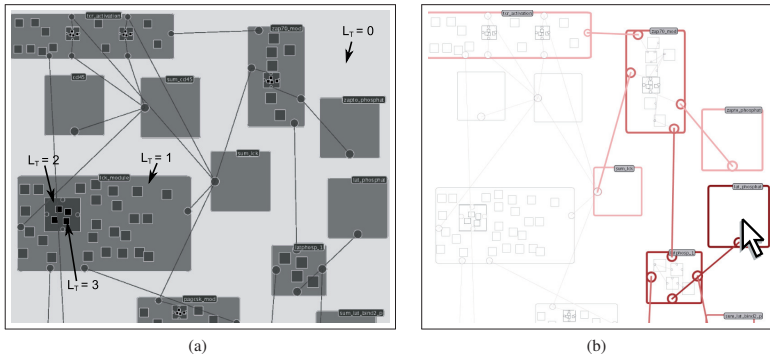


Figure 6.11.: VSs of the EGF model encoding different distance metrics. (a) Hierarchical level L_T . The distance is encoded using a gray gradient; complete TCR signaling pathway ($L_T = 0$), lymphocyte-specific protein tyrosine kinase *Lck* ($L_T = 1$), and more detailed information of *Lck* ($L_T = 2$ and $L_T = 3$). (b) Interaction distance with $d_{max} = 3$; graph-theoretical distance of each element to the current element in focus; distance is encoded using a red gradient. The focus is on the protein *LAT*.

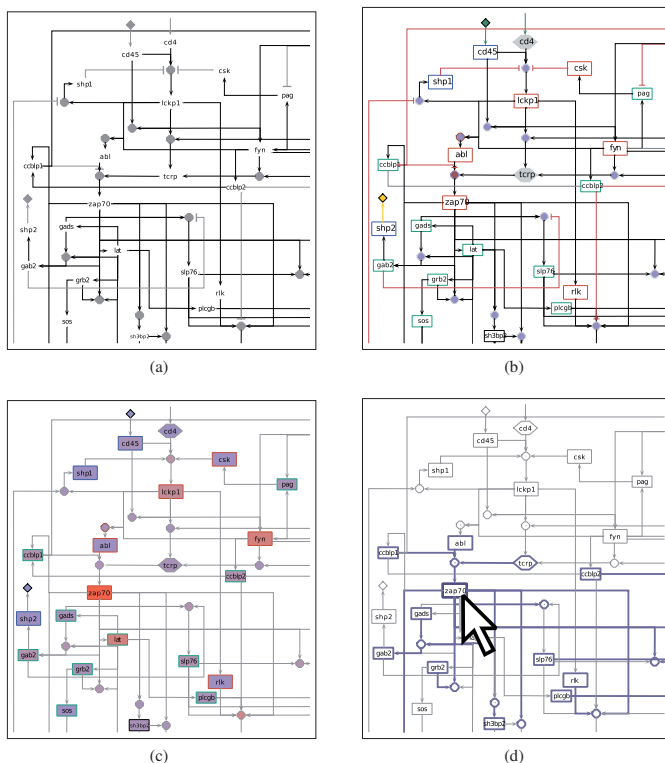


Figure 6.12.: Different VSs applied to the logical TCR model tailored to the modeling task and individual preferences. Only part of the model is shown. (a) Local VS that represents the network in textbook style showing only proteins, their names and relationships. (b) Global VS 'Logical Model' that shows a biologically intended visualization using metadata such as type of protein or interaction. (c) Global VS 'Connectivity' depicting the connectedness of proteins and other biological entities in the network. (d) Global VS 'Interaction Distance' by clicking on a certain protein (e.g., zap70), the adjacent molecules within a given interaction distance are emphasized.

6.4.2. Visual Analysis of the Logical TCR Model

In this section the application of VSs to the logical TCR model is presented. Biological aspects of the TCR model are described in Chapter 2, Sections 2.1.2 and 2.5.2.

In Figure 6.13, different results of structural analyses of the TCR model generated by the external analysis tool CELLNETANALYZER are overlaid on top of the network. These resulting structural network

properties like network-wide relations are highlighted by applying different visual scenarios. Figure 4.15 in Chapter 4 shows an intuitive representation of signaling paths in the logical model using the global VS described in Chapter 4, Section 4.4. Figure 6.13 shows the activation of receptor molecule `cd28` and the adequate representation of analysis results. Thereby, the active network (green and red nodes) is depicted in the context of the entire network. In this way, obvious results can be depicted such as the inhibition of the extracellular signal-regulated kinase ERK. However, even non-trivial results can be highlighted, for example, the activation of kinase `jun` was not expected, which is elucidated in the network context. In Figure 6.13b, initial values and analysis results are presented simultaneously by encoding those using different visual properties. Active elements are rendered opaque whereas inactive elements are rendered semi-transparent. External data are also mapped to the edges since this is a powerful visual cue for understanding which substructures of the network are active [26]. Another example is given in Appendix B by Figures B.6 and B.7.

As stated above, an interesting feature is the integrative representation of heterogeneous results or different datasets within a single illustration encoded by different visual properties. They often reveal correlations or other relations, which are not obvious in separate visualizations per se. For instance, for a certain protein, the initial value and the value obtained from the analysis are mapped to the node color and the node label, respectively. In this way, multiple visual properties are used to convey the underlying data and more information can be presented in the visualization without using complex glyphs. Hence, only the information density but not the visual density of the representation is increased. Hence, different facets of the cellular system and its processes can be integrated.

6.4.3. Visual Analysis of the Logical EGFR/Erbb Model

In this section the application of VSs to the logical EGFR/Erbb model is given. The biological model is described in the Chapter 2, Sections 2.1.2 and 2.5.1. In Figure 6.14, visual scenarios of the EGFR/Erbb model are given. Results of structural analyses are shown in the network context and highlighted using different visual scenarios. Figures 4.14 and 4.15 in Chapter 4 provide intuitive representations of the logical model using the global VS described in Section 4.4. The protein type is encoded by node border color whereas the entity type (e.g., receptor, ligand or reservoir) is encoded by node shape. Activation and inhibition is encoded by edge color combined with arrow type. Figure 6.14 depicts instantiated local VSs that emphasize external analysis data, or important structural properties of the species such as participation in feedback loops. In this way, the modeler is able to visually analyze which species has a specific property. In Figure 6.14a, all species can have one of two logical states – active or inactive. The node fill color reflects the state, namely green for an active state and red for an inactive state. For that purpose, a discrete transfer function is used to map each activation state to a discrete node fill color using a lookup table. In Figure 6.14b, the participation of species in feedback loops is presented¹². The ‘participation’ value is mapped to the node fill color via a continuous transfer function. A red gradient is used – dark red means high participation in loops; light red means low participation. A species represented by a white node fill color does not participate in any feedback loop. In summary, VSs can emphasize entities with a specific network property, but also reveals network-wide structures and patterns. This may guide further analysis.

6.4.4. Visualization of Time-Series Data

In Figure 6.15, the logical EGFR/Erbb model is visualized applying VSs. The visual representation is enriched with time-series data from experiments. Each subfigure shows the status of the system under

¹²For further information on the analyses the reader is referred to Samaga et al. [167].

investigation at a specific time point. At the beginning (0 minutes) only the protein *gsk3* is active; most of the other proteins are inactive (Figure 6.15a). Then, during the early phase (0–10 minutes) several proteins become active and the active protein *gsk3* becomes inactive (Figure 6.15b). In the later phase (60–240 minutes) the protein *gsk3* switches back to the active state, some others remain active but some proteins become inactive again (Figure 6.15c). In the VS a certain color coding is used. Green means the species is in an active state; red means the species has an inactive state. For species that are semi-transparent, no analysis data are available; that is, they are not measured during the experiment. However, the semi-transparent species are used as contextual information for the improved understanding of the network.

The visualization using VSs can reflect the behavior of the underlying system over time and aid the interpretation of experimental results. For instance, it can support the modeler in observing and understanding changes in conditions, propagation of signals or processes of up- and downregulation in the context of the network. Hence, the modeler can obtain a clue about the signaling processes within the cell. VSs can be switched in a fast and convenient manner in VISUAL EXPLORER. However, so far there is no possibility to ‘play’ different VSs at a constant frame rate in order to show animated time-series data.

6.5. Conclusions

As pointed out by Bourqui and Westenberg [26], each modeling or analysis task has its own adequate visualization. Depending on that, the modeler should be able to define the most effective visual settings. In this chapter, the general idea of ‘Visual Scenarios’, a mechanism for encapsulating visual settings for a customized visual representation, is introduced. It is based primarily on the approach of the VIZMAPPER, but extends it by several visual properties like scale or transparency and incorporates components for layout and exploration scheme, and the applicability to modular models. Integrating visual scenarios into VISUAL EXPLORER, gives the modeler not only better control over the exploration schemes and the network layout but also the ability to adjust the visual representation of the network in a very flexible manner. This is only constrained by the available set of visual properties for the networks elements, the number of available layouts and exploration schemes.

Several global VSs are implemented to ensure homogeneous views for common modeling and analysis tasks where the set of visual settings is predefined. By using global VSs the researcher can be guided in a visual sense, and saves time and effort to become familiarized with the visual representation and the signaling processes behind it. For instance, the global VSs ‘Dynamic Model’ and ‘Logical Model’ formalize the visual representations for dynamic and logical models. Another use case of global VSs is the analysis and visualization of structural properties. Unlike in many other tools, the analysis of the network structure (handled, for example, by the global VSs ‘Connectivity’ and ‘Interaction Distance’) is performed in a very interactive way incorporating the network context and the modular structure.

Global VSs can be instantiated to local VSs and further modified. They can be very easily and precisely adapted to the specific modeling task or individual preferences. Each VS created or modified for a specific task can be saved and reused when the modeler has data with similar characteristics. It is important to state that predefined and customized visualizations do not contradict, but rather complement each other. For instance, a network visualized with the VS ‘Logical Model’ can be enriched with additional datasets. This data can be visually represented by mapping it on further visual properties (by creating a new local VS derived from the global VS). In this way, the characteristics of the global VS and the new visual encodings can be combined.

Using VSs, the visual representation of external data in context of the network can be more intuitive and flexible, for example, by emphasizing species that are active in a particular experimental condition or other biological context. Integrating conditional or time-related data with the visualized network

6. *Visual Scenarios*

structure in a single view supports modelers in interpreting underlying signaling processes and may foster new insights and knowledge.

Switching VSs is very fast and with a direct visual feedback. However, when complex layout strategies are considered, it is difficult to meet this requirement. With that very flexible approach of ‘Visual Scenarios’, the visual representations for dynamic and logical models described earlier may adapt to the SBGN notations in order to familiarize users with the models handled in PROMOT.

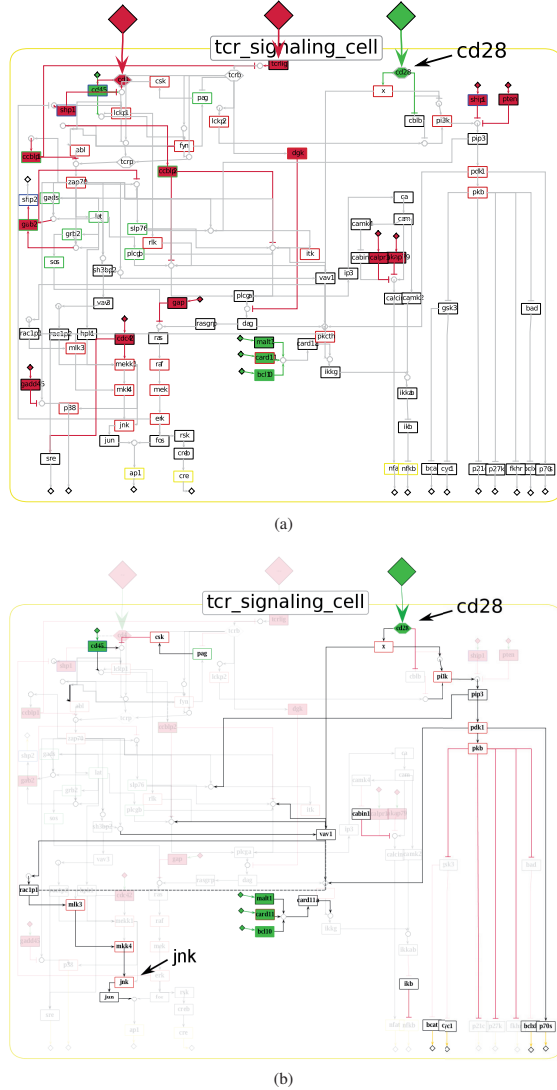
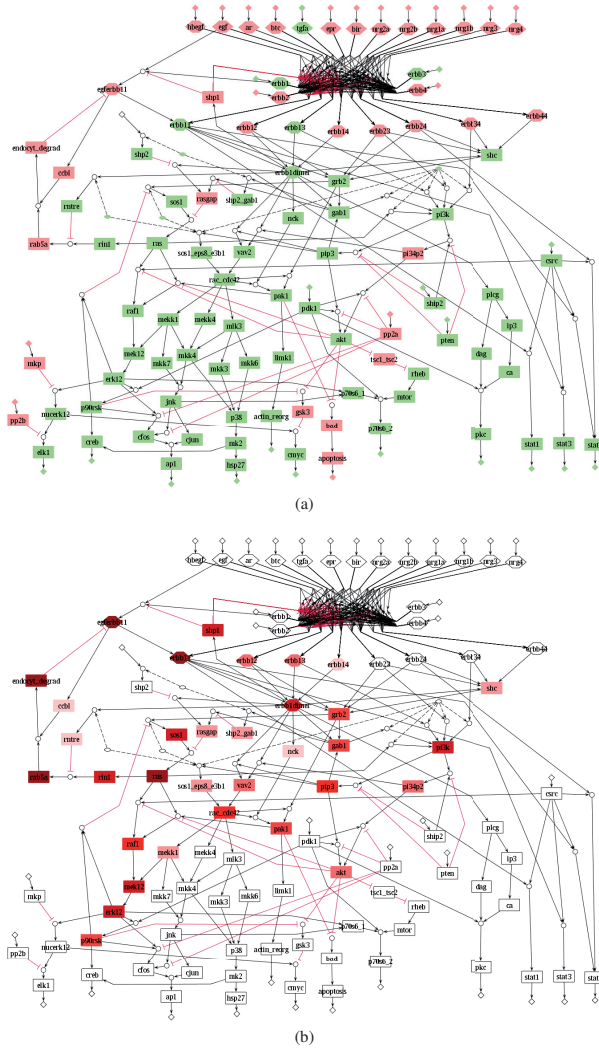


Figure 6.13.: Two different local VSs of the logical TCR model in VISUAL EXPLORER showing the activation of receptor molecule `cd28`. The global VS ‘Logical Model’, depicted in Chapter 4, Figure 4.15, is instantiated to a local VS. (a) Adequate representation of analysis results; highlighting of non-trivial results (not expected was activation of kinase `jnk`). (b) Visualization of initial values and analysis results simultaneously by encoding them using different visual properties (initial values `red = 0`, `green = 1`; analysis results, with transparent being not active and opaque being active). Figures courtesy of Julio Saez-Rodriguez. 121

6. Visual Scenarios



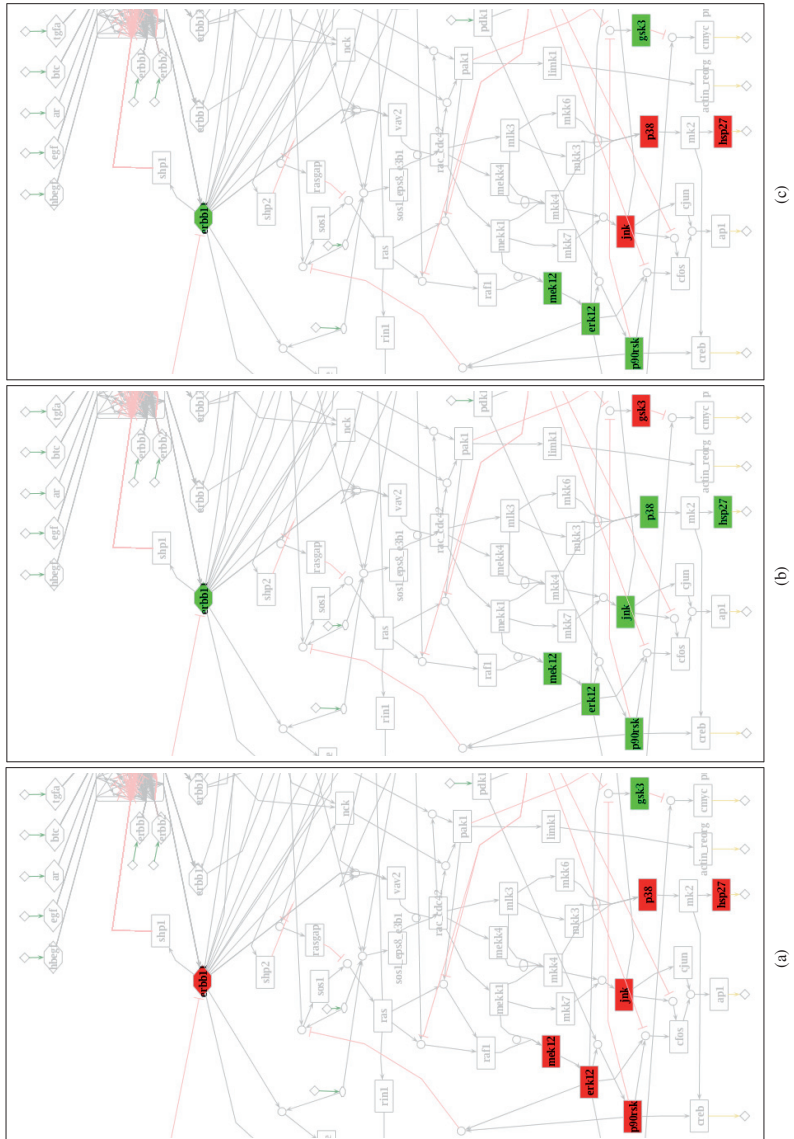


Figure 6.15.: Visualization of time-series data in the EGFR/Erbb network. Different time points at (a) 0 min, (b) 0–10 min and (c) 60–240 min are depicted. Underlying data courtesy of Regina Samaga.

6. *Visual Scenarios*

The structure of the underlying mathematical equations is represented but not their interpretation in their specific application domain.

W. Wiechert et al. [204]

CHAPTER 7

Layout Strategies for Modular Models

When dealing with models describing intra-cellular processes, not only the definition of components and their visual properties is essential but also their layout; that is, the placement of components and the routing of edges. This is particularly important for modular models since their structure is much more complicated. Thus, comprehensible layouts for modular model are needed for effective exploration and navigation, and especially for the understanding of the underlying complex system.

As described in Chapter 2, Section 2.4.2, in PROMOT modular models are set up either by hand, or obtained using the built-in SBML parser¹. In the past, modular models were usually created by hand and their layout was an integral, but also time-consuming, part of the modeling process. An example is illustrated in Figure 7.1. In recent years, model integration has become more and more important. For this, many models are provided in SBML format by diverse databases and internet repositories. Currently, these SBML models have no layout. Thus, after modularization of the models in PROMOT, an appropriate layout² must be generated, ideally in an automatic fashion.

Although it is desirable to work with modular models, it is still necessary to ensure compatibility and integration with tools and workflows that do not support modular modeling. This applies not only for the mathematical model but also for its visual representation. Hence, in this thesis when dealing with the layout of modular models the following questions are discussed:

1. How can modular models be laid out in order to emphasize their properties? How can local as well as global structures be incorporated in the layout process?
2. Once an appropriate layout for a modular model is given, it is often also desired to generate a flat graphical representation. This is necessary in order to present and publish it in print media or comply with tools that work on flat models. The question is how modular models can be laid out towards flat representations.

In this chapter, two different layout strategies for modular models are introduced that address the questions raised above.

¹In order to obtain a modular model the flat SBML model has to be modularized, for example, with the method of Saez-Rodriguez et al. [162]. For the modularization of SBML models PROMOT provides an automatic workflow.

²The layout process for modular models can be assigned to the ‘Mapping’ step in the visualization pipeline – in this case mainly geometric mapping.

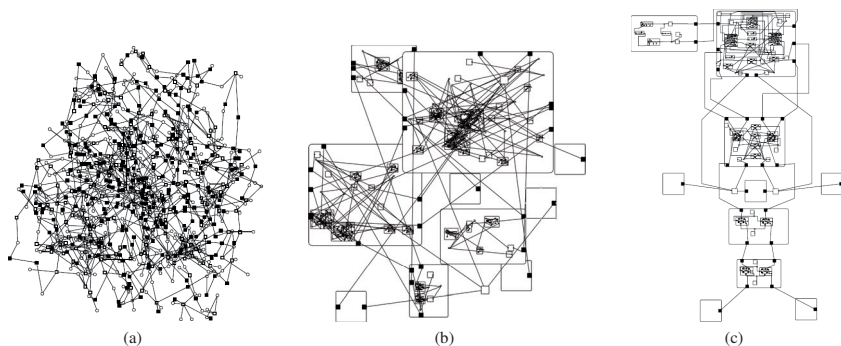


Figure 7.1.: Different layouts of the dynamic EGF model: (a) visualization of an unstructured (flat) version using the analysis and visualization tool PAJEK [14], (b) a decomposed, modularized version with random layout, and (c) a manual layout of the version in (b).

7.1. Research Problem

The generation of automatic layouts for modular models can be traced back to the basic problem of the layout of compound graphs. The drawing of compound graphs has been the subject of considerable research for many years. When drawing compound graphs in the context of *force-directed* methods [45] the researcher is traditionally faced with two contrary approaches: *local* [22] and *global* [48, 55] layout. The local approach calculates the layout for each subgraph independently of the global constraints. This results in fast computations, but in low quality regarding the global structure and across different hierarchical levels. In contrast, the global approach considers all subgraphs across all hierarchical levels in a single step to compute the final drawing. In general, this leads to a poor runtime but a reasonable good layout quality. These two approaches merely mark the outer boundary of a wide spectrum of solutions. However, there is a lack of practical solutions in-between.

In order to overcome this drawback the HyCoGLa algorithm is introduced. The core functions of the proposed algorithm are based on the spring-embedder first introduced by Eades [53]. Therefore, in the remainder of this thesis the algorithm is referred to as the *extended spring-embedder*. The basic concept of force-directed methods, relevant extensions, advantages and disadvantages are already introduced in Chapter 3, Section 3.2.3.

The HyCoGLa approach is aimed at fulfilling certain specific requirements which are formulated below. Some are taken from the compiled list of aesthetic criteria in Chapter 3, Section 3.2.1.1.

- Node overlaps should be reduced (cf. C1).
- Edge-edge crossings and edge-node crossings should be reduced (cf. C2).
- The length of edges should be uniform. The deviation in edge lengths should be small (cf. C4).
- Inclusion relations (parent-child relations) should be displayed explicitly (cf. C9).
- Scaled subgraphs should be considered.

- Terminal nodes should be incorporated and constrained in their position.
- Domain-specific requirements for signaling systems such as subcellular localization should be considered.

Another layout strategy for modular models is the flattening of the visual representation. Some research has been done in ‘model flattening’ based on the underlying equation or graph structure [151, 155, 156]. PROMOT, for instance, facilitates the flattening of the modular model by optimizing the particular equation set of the mathematical model. This allows the model to be used with existing simulation and analysis tools (cf. Chapter 2, Section 2.3.3.3). Analogous to these efforts, also the visual representation of the modular model itself should be flattened. A main characteristic of the visual representation of a modular model in VISUAL EXPLORER is that it is displayed using a single integrated view where elements are scaled according to their position in the hierarchical structure. Hence, elementary modules do not have the same size (cf. Chapter 4, Section 4.2).

For exploring the model on the screen several interactive exploration schemes can be used such as those described in this thesis. In this way, also elements on deeper hierarchical levels that are too small to be visible in the global view can be visually analyzed in a local view. However, when printing or exporting to a static image, interactive exploration facilities are no longer available and thus parts of the graphical model are not accessible (cf. elementary modules of the EGF model depicted in Figure 7.1c). Therefore, it is desired to display all elementary modules independently of their current hierarchical level in an equal size. In this way, equivalent modules throughout the network have the same size, and thus a similar visual emphasis level. Moreover, links between elements, for example, proteins which directly interact, are visually easier to follow. This type of representation is commonly used as, for example, depicted in the ‘EGFR Pathway Map’ [140] created with CELLDISIGNER [63]. For a flattened graphical representation the following requirements are important.

- In order to unify the size of elementary modules, more space needs to be allocated for each of them. Thus, the modular and hierarchical structure is visually broken up – the network no longer expands in depth but in breadth using more space³. In comparison to elementary modules, composite modules have no uniform size. Their size should be automatically computed based on the bounding box of all inner (elementary or composite) modules.
- It is necessary to unify the size of elementary modules, but it is also important to preserve structural properties of the original network layout. Constraints such as orthogonal relationships (e. g., non-overlapping nodes, proximity relationships) and overall graph topology [131] are crucial for preserving the mental map.
- The transformation from the original to the flat layout should be smoothly animated in order to maintain the visual context.

In the following, two new layout strategies are described in detail. The first approach, HyCoGLa, is suitable for drawing modular models while emphasizing the modular and hierarchical structure. The second approach, FlatLayout, is able to transform a modular layout representation into a flat layout representation in order to be compatible with common visual representations of other tools (cf. Chapter 3, Section 3.6.2). The former is described in Section 7.2; the latter is introduced in Section 7.3.

³For example, a default scale factor $s_d = 0.2$ and an elementary module with $L_T = 3$ are given. Then this module is 125 times smaller compared to an elementary module with $L_T = 0$ (cf. Chapter 4, Equation 4.1).

7.2. HyCoGLa

HyCoGLa, short for Hybrid Compound Graph Layout, is a new layout approach. It is dedicated to the layout of modular models set up by dynamic or logical modeling formalism. However, it also incorporates domain-specific constraints for signaling processes. In the following, the research problem is discussed and the HyCoGLa algorithm is introduced. Moreover, how domain-specific knowledge is considered and how the algorithm deals with concepts from modular modeling is shown in detail. The developed basic ideas and the related research on HyCoGLa was performed during a supervised diploma thesis [8].

7.2.1. The HyCoGLa Algorithm

The overall procedure of the HyCoGLa algorithm is illustrated in Figure 7.2 using the Toy model. The procedure is described in Appendix D, Algorithm 3. Basically, the inclusion tree of the modular model is traversed in level order (bottom-up). Thereby, in each traversal step two hierarchical levels of the model are considered for layout. First, the lower two levels in the model are used. When this step is finished the next hierarchical level along with the last already laid out hierarchical level is considered for layout. This will continue until all hierarchical levels are processed. Thereby, for two subsequent traversal steps one hierarchical level overlaps.

The final layout of each traversal step is computed in different phases. This includes the handling of the skeleton graph (see Chapter 3, Section 3.2.3.2), the layout of the skeleton graph and the overall layout of the graph using the extended spring-embedder, and a post-processing phase. They are described in the following sections.

7.2.1.1. Dealing with the Skeleton Graph

In the first phase the coarse structure of the subgraph (skeleton graph) is computed by considering all nodes excluding leaf nodes; that is, leaf nodes are temporarily removed before laying out the skeleton graph with the extended spring-embedder. In the next phase based on the skeleton layout the overall layout is computed. For this purpose each of the removed leaf nodes of the former phase is set to the position of its adjacent node in the skeleton graph. Finally, the complete graph is laid out including the temporarily removed leaf nodes using the extended spring-embedder. There are several advantages using the skeleton. For instance, the algorithm has an improved runtime because of the lower number of nodes in the skeleton graph, or the removed leaf nodes do not influence the overall layout significantly. This optimization also leads to the reduction of local minima⁴.

7.2.1.2. Layout Using the Extended Spring-Embedder

In order to compute the layout of the basic skeleton and the overall layout of the graph, the developed extended spring-embedder (Appendix D, Algorithm 4) is applied. Initially, the various forces acting on the nodes (e. g., spring, repulsive or gravitational forces) are computed. This is explained in Section 7.2.1.4. Subsequently, the forces are combined and the nodes are moved in the direction of the forces (see Section 7.2.1.5). This procedure is repeated as long as either the maximum number of iterations is reached or the standard deviation of the sum of the forces acting on the nodes is below a certain threshold. As a further optimization the update of the nodes is improved by the concept of local temperature [59], which avoids oscillations and improves the termination of the algorithm.

⁴Local minima in force-directed approaches are potential configurations of nodes, which in an energetic sense are convenient but not optimal. They force the termination of the algorithm.

7.2.1.3. Post-Processing

After calculating the layout of the complete graph different post-processing steps have to be performed in order to refine the layout. First, terminal nodes had to meet the most important constraint: They should be located at the border of the respective module. However, due to the strong relationships of terminal nodes to nodes outside of the subgraph, mostly the terminal nodes are not located at the border of the subgraph. Therefore, in a post-processing step, the terminal nodes are positioned at the nearest border of the subgraph. Another post-processing step is necessary since the spring-embedder algorithm is not able to completely eliminate node overlaps [118]. Hence, after computation of node positions, potential node overlaps are removed.

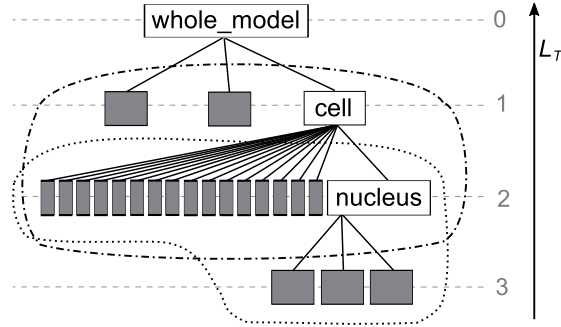


Figure 7.2.: Traversal of the inclusion tree of the Toy model. Starting with $L_T = 3$, the composite module `nucleus` along with the content of composite module `cell` (nodes within dotted line area) are laid out. Hereby, the positions of nodes of `nucleus` under consideration of relationships to outer nodes are calculated. In the next and final step, with $L_T = 2$, the composite module `cell` along with the content of composite module `whole_cell` (nodes within dashed-dotted line area) are drawn. The inner layout of the composite module `cell` is adjusted related to the outer layouts. Note that module `nucleus` is treated as leaf node of `cell` and, therefore, its inner layout is not changed. The modular Toy model is depicted in Chapter 4, Figure 4.12.

7.2.1.4. The Force Model

In HyCoGLa the force model is based on several forces used in common spring-embedder approaches. However, adjustments are necessary due to the modular structure with scaled subgraphs. The interplay of forces is illustrated in Figure 7.3.

Spring forces F_S affect the nodes connected directly by edges. In the standard spring-embedder, the length d of an edge $e(u, v)$ is equal to the distance between the node centers u_{center} and v_{center} . In contrast, in HyCoGLa the actual length of the edge is computed by getting the distance between the nodes (in case of link nodes) or between the terminal nodes (in case of elementary or composite modules). Thus, the node sizes are considered in the calculation and an attempt is made to avoid node overlapping [118]. The spring forces F_S are calculated as depicted in Appendix D, Algorithm 5. *Repulsion forces* F_R which affect the nodes are calculated including the size and scaling of the nodes like the edge forces. Repulsion

7. Layout Strategies for Modular Models

forces are calculated solely between nodes with the same parent node to improve the calculation. The calculation of the repulsion forces F_R is depicted in Appendix D, Algorithm 6. *Gravitational forces* F_G affect nodes that are not connected to the graph and are used to prevent them from drifting apart. That means the nodes are moved by the forces in the direction of the center of mass of the (sub-)graph (see Figure 7.3b). Furthermore, for the layout of subgraphs they are an important force for compacting the inner nodes and thus emphasizing their strong relations. Algorithm 7 in Appendix D depicts the calculation of the gravitational forces F_G . *Magnetic forces* F_M are added in order to consider directed networks and their encoded domain-specific knowledge. Their application is discussed in Section 7.2.2. Algorithm 8 in Appendix D depicts the calculation.

All forces are computed by considering the hierarchical structure. For instance, the repulsion and gravitational forces which affect the submodule are added to its child nodes afterwards (as shown in Section 7.2.1.5). Since the edges between nodes of scaled subgraphs also have to be scaled, the scaling is integrated in the calculation of the ideal edge length l (Appendix D, Algorithm 5).

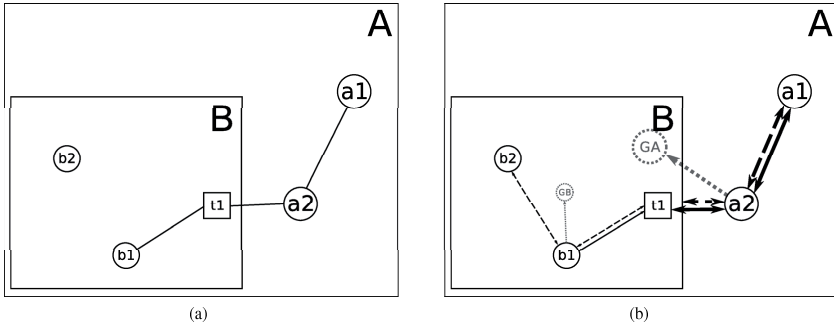


Figure 7.3.: Illustration of the force model. (a) Example model with two composite modules A and B, elementary modules a1, a2, b1 and b2, and a terminal t1. (b) Applied forces are illustrated exemplarily on elementary modules a2 and b1. Thin lines in module B denote scaled forces. Spring forces F_S are represented by solid lines, repulsion forces F_R are depicted with dashed lines and gravitational forces F_G are represented by dotted lines. G_A and G_B represent the gravitational centers of module A and module B, respectively. Adapted from Aschenbrenner [8].

7.2.1.5. Update of Node Positions

After each layout iteration all nodes in the network are smoothly translated to their new position. In order to determine the new position of a node v , the calculated forces of the outer nodes are used directly. The combination of the applied forces is shown in Equation 7.1.

$$F(v) = F_S(v) - F_R(v) - F_G(v) + F_M(v) \quad (7.1)$$

First, the repulsion forces F_R and gravitational forces F_G of the nodes, which affect the parent nodes are added to the inner nodes. Hereby, these forces have to be scaled. Afterwards the node positions of

all inner nodes are also updated. Thereafter, the new size and position of the submodules are set to the bounding box of the inner nodes.

HyCoGLa like other force-directed approaches supports animation by default (by the different layout iterations). Therefore, the layout process can be interactively displayed and thus provides a visual feedback how the layout evolves over time. Additionally, the mental map can be preserved.

7.2.2. Domain-Specific Knowledge

As already mentioned in Chapter 3, Section 3.2.4, many standard graph layout methods consider only the topological information given by the graph structure. However, for an improved representation of the properties of the modeled system, domain-specific characteristics must be included in the layout process. Recently, some research has been carried out in order to define and characterize domain-specific knowledge for signaling systems [11, 70]. Such knowledge of signaling processes is also considered in the HyCoGLa approach, but for logical rather than dynamic models. In the following, particular domain-specific knowledge and its consideration in HyCoGLa is discussed.

7.2.2.1. Direction of Signal Flow

Describing signaling processes based on logical modeling, the causal flow of the signals is important. In common visual representations this is often encoded in a top-down arrangement of the network elements. Thus, modelers can easily identify the sequence of events described by the logical model [209].

Figure 7.4 shows the consideration of directionality of reactions resulting in an overall causal signal flow in from top to down. This directional constraint is formulated using magnetic forces F_M [189] that are added to the force model of the nodes (see Section 7.2.1.4). The strength of the magnetic forces depends on the length of the edge and the angle between edge and magnetic field. In HyCoGLa a global magnetic field with the orientation $(0, 1)$ (i. e., north-south orientation) is used to adjust the direction of the edges from top to down. Another example can be found in Figure 7.5 (cf. Subfigures 7.5a and 7.5b).

In principle, it is also possible to align edges in other directions, for example, the signal flow from left to right by applying a magnetic field with an appropriate orientation.

7.2.2.2. Subcellular Localization

Logical models of signaling systems are often organized in *input*-, *intermediate*- and *output* layer [167]. Usually, these layers are associated with subcellular compartments such as cell membrane, cytoplasm or nucleus (cf. Figure 2.7 in Chapter 2). *Input* elements allow the definition of incoming signals of the model. As illustrated in Figure 7.5c, they are aligned on top in the input layer of the network. *Output* elements describe the outgoing signals of the model. Analogous to *input* elements, they are aligned on the bottom in the output layer of the network. All other elements are placed in the intermediate layer located between *input*- and *output* layer.

A great deal of domain-specific information exists, in particular coded in topological and functional features such as cycles, cascades or molecular complexes. In principle, all these biological conventions might be relevant for a comprehensive layout. However, this is not within the scope of the thesis and not captured by the HyCoGLa approach. As a suggestion, some of these relevant structures can be encapsulated in appropriate module structures and therefore are considered in the layout process.

7. Layout Strategies for Modular Models

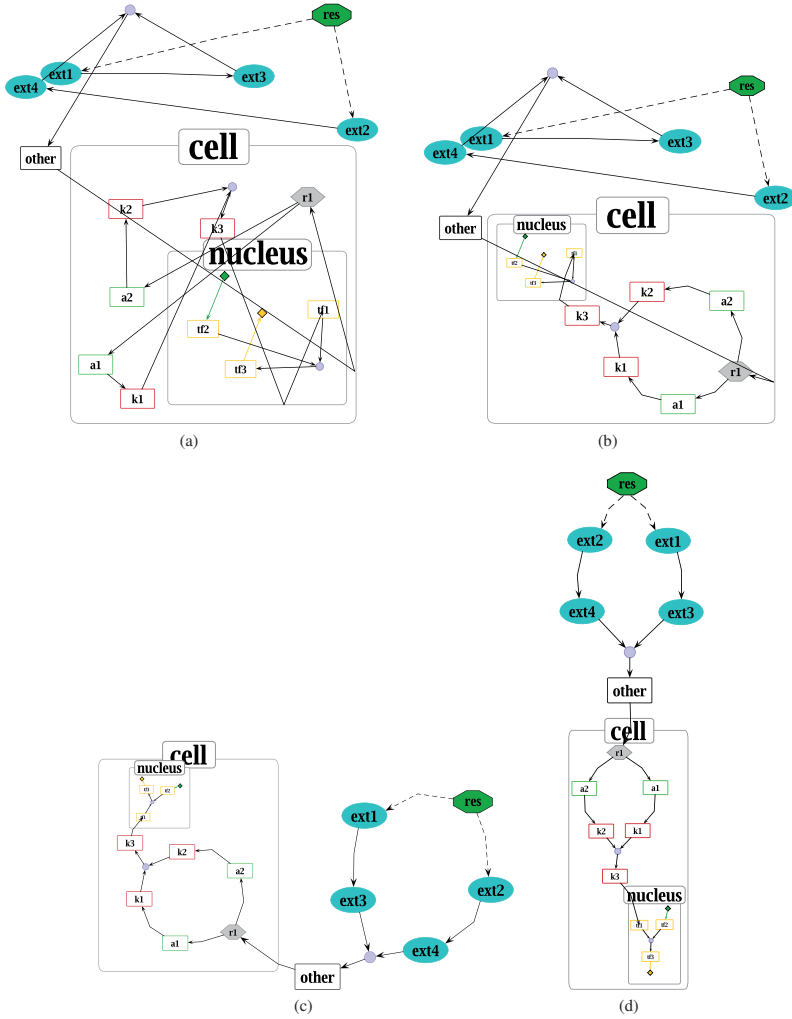


Figure 7.4.: Comparison of different layouts using a simplified Toy model. (a) A local random layout for each of the modules is computed. (b) A local, spring-embedder [53] for the module `cell` is applied. Initially, the layout of (a) is used. (c) The HyCoGLa layout for all modules is applied; fewer crossings occur. (d) The HyCoGLa layout for all modules is depicted; the direction of signal flow from top to down is considered.

7.2.3. Visual Handling of Multiple Instances

As described in Chapter 2, Section 2.3.3.1, PROMoT supports the reuse of modules. Thus, in a complex modular model several instances of the same module, for example, the MAPK signaling cascade may occur. An interesting question is how the layout of such identical modules is handled.

In Figure 7.6, two possible approaches are presented. One approach is to compute the layout for each module instance individually (Figure 7.6a). Then an adequate layout⁵ can be assigned to each particular instance. However, multiple instances with different layouts are difficult to detect and hard to distinguish from other modules. A second approach is to compute the layout of one module instance and apply it to all other instances automatically (Figure 7.6b). The advantage of this approach is that the layout has to be computed just once. Since each instance of the module and its subunits has the same layout, it is easier to recognize modules that appear several times in a model. Thus, patterns like network motifs can be detected easier. In this way, not only the module itself can be reused, but also its layout.

7.2.4. Generalization of the Concept

So far, as described in Section 7.2.1, in each traversal step two hierarchical levels are considered for layout. In some cases, this is not flexible enough.

In order to adjust the trade-off between layout quality and runtime two additional parameters can be introduced – the number of hierarchical levels n , and the number of overlapping hierarchical levels m ⁶. The use of the parameters n and m is illustrated in Figure 7.7. The parameter n determines how many hierarchical levels (including their nodes) are considered in a single traversal or calculation step, and thus how much ‘local’ or ‘global’ the algorithm will work. When choosing $n = 1$ the algorithm works similar to a local approach because only a single level is drawn independently from global constraints. In contrast, when choosing $n = \text{depth}$, the result is comparable to a global approach where all nodes in the graph are considered in a single calculation step. For all other values of the parameter n with $1 < n < \text{depth}$ the algorithm results in interim solutions. The parameter m defines how many hierarchical levels are overlapped in a calculation step. When choosing $m = 1$ only one level overlaps; that is, the last hierarchical level from the current step and the first hierarchical level from the subsequent step. In this case, very little information about already laid out parts of the network is considered. In contrast, when choosing $m = n - 1$ much of the already calculated (i. e., laid out) information and only a small amount of new information is considered for calculation. Consequently, from the definition of the two parameters n and m , it follows that $n > m$.

7.2.5. Runtime and Layout Quality

For measuring the quality of the new HyCoGLa method, the runtime of the algorithm and the layout quality are determined⁷. This is done for the HyCoGLa and a local standard layout method [53].

In Figure 7.8a, the runtime of the HyCoGLa and the local approach is compared. As expected, the overall runtime of HyCoGLa is higher compared to the local approach. Especially, the four times higher runtime of the layout calculation T_C is induced by the consideration of the global relationships. However, the runtime of the HyCoGLa algorithm depends on the hierarchical structure of the graph. Using graphs with $\text{depth} \leq 2$ the complexity of the algorithm is quadratic with respect to the number of nodes ($\mathcal{O}(k * n^2)$), equally to other spring-embedder methods. Graphs with $\text{depth} > 2$ can benefit from the HyCoGLa

⁵Adequate in terms of the related network context.

⁶Note that the parameters n and m are not yet considered in the implementation.

⁷All experiments were performed on a personal computer with Intel® Pentium 4, 3GHz CPU, 2GB RAM and Linux®.

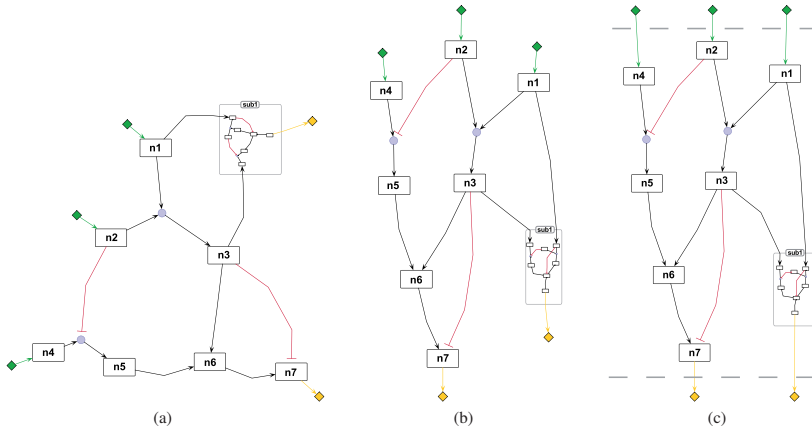


Figure 7.5.: A logical model of a signaling system laid out with the HyCoGLa algorithm using different constraints. (a) No domain-specific constraints; (b) the directionality of reactions in the network is considered which results in a top-down orientation. (c) Consider special semantics of nodes in order to place them accordingly. A green rhombus refers to an input element; a yellow rhombus encodes an output element. For instance, input and output elements align in respect to an imaginary input- and output layer (dashed lines). Adapted from Aschenbrenner [8].

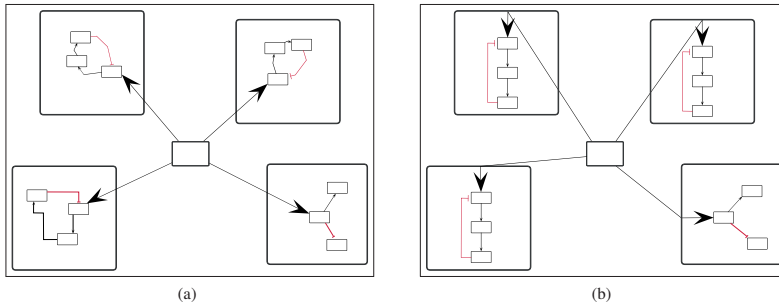


Figure 7.6.: Handling of multiple instances of a negative feedback motif encapsulated in a module. (a) Individual layout for each of the motif instances according to the equilibrium of the forces, and (b) uniform layout of all motif instances. Thus, equal instances can be recognized better. Figure is adapted from Aschenbrenner [8].

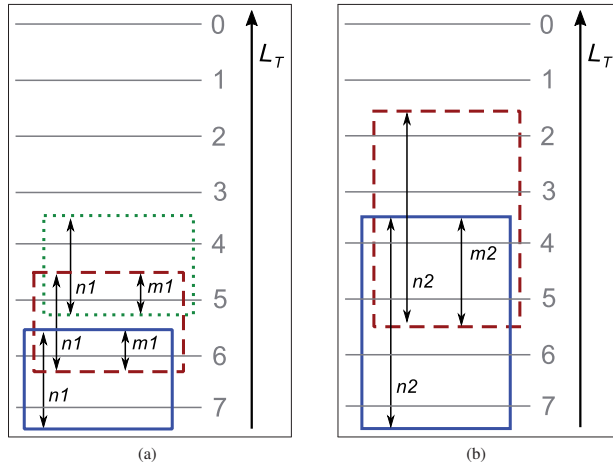


Figure 7.7.: Traversing of inclusion tree with different values for the parameters n and m . Each box illustrates one subsequent calculation step (from solid over dashed to dotted border). (a) Three subsequent calculation steps starting from level $L_T = 7$ using $n_1 = 2$ and $m_1 = 1$. (b) Two subsequent calculation steps starting from level $L_T = 7$ with parameters $n_2 = 4$ and $m_2 = 2$.

approach. In this case, the layout of the whole graph is not computed at once but in subgraphs (cf. Figure 7.2).

The layout quality can be evaluated using a number of criteria. As shown in Figure 7.8b, the experiments on layout quality reveal that the HyCoGLa algorithm produces better results in the measured criteria compared to the local method. In particular, the criteria are the number of edge-edge crossings, the number of edge-vertex crossings, and the deviation of the average edge length from the sum of the ideal edge length.

The graph in Figure 7.9 is an reasonable example to show the differences between the local method and the HyCoGLa method. In Figure 7.9b the global relationships are considered whereas in Figure 7.9a the layouts of the subgraphs are calculated without any global constraints. Figure 7.9b illustrates the advantages of the HyCoGLa method – the nodes are well placed across different hierarchical levels, edge-vertex crossings are minimized and edge lengths are almost uniform.

7.3. Flat Layout

Although it is desirable to work with modular models, it is still necessary to ensure compatibility and integration with tools and workflows. For instance, the mathematical model that is used to describe a modular model can be obtained by flattening. An important area in this context is the visual representation of modular models. Often, they need to be represented in a flat manner in order to be compatible

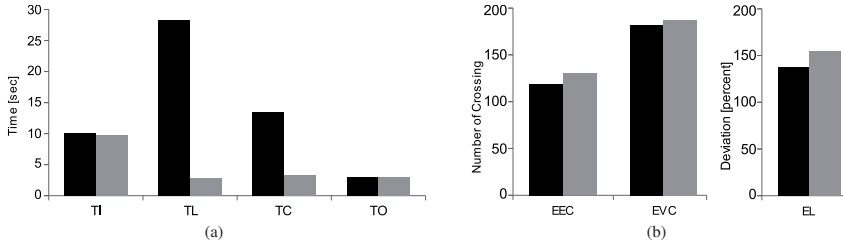


Figure 7.8.: Results for HyCoGLa (black) and a local layout method [53] (gray). (a) The runtime for the major steps of the algorithm including the setup of the inclusion tree (T_I), initialization of the special data structure (T_L), layout calculation (T_C) and removal of node overlapping (T_O). (b) The layout quality for measured parameters including the number of edge-edge crossings (EEC), the number of edge-vertex crossings (EVC) and the deviation of the edge lengths from the sum of the ideal edge length (EL). Adapted from Aschenbrenner [8].

with print media, publications, scientific communication, and tools that do not support modular modeling (e. g., CELLNETANALYZER).

The FlatLayout method is able to automatically convert a modular visual representation into a flat visual representation. ‘Flat’ does not mean to remove hierarchy, but rather to give each elementary module in the network the same size. An example is given in Figure 7.11b.

In the following, the algorithmic solution is introduced and described. It is shown in detail how flat and comprehensible visual representations of a modular model can be generated.

7.3.1. The Flattening Algorithm

As described in the introduction, the overall aim is to flatten the graphical representation of the modular model. The approach is based on the ContextualZoom introduced in Chapter 5, Section 5.2.2 but with two major differences according to the following requirements: First, all elementary modules in the model have the same size independent of focus and context information. Second, the overall size of the model is not fixed. Additionally, the FlatLayout maintains the relative positions of elements to reduce user disorientation by using layout constraints, for example, stated in Storey and Mueller [187].

The layout method consists of several steps illustrated in Figure 7.10: First, a certain elementary module is scaled to the appropriate, uniform size. For each surrounding module in the same parent module a translation vector is computed. The surrounding modules are translated according to these translation vectors (Figure 7.10b). Then the size of the parent module is adjusted according to the bounding box of the inner elements (Figure 7.10c). The translation is recursively applied to all elements outside the parent module in order to prevent node overlapping until the root element is reached. Then, this procedure is applied to the next elementary module until all elementary modules in the network are processed. The algorithm of the FlatLayout is given in Appendix D, Algorithms 1 and 2.

The complexity of the algorithm is linear with respect to the number of elementary modules in the model ($\mathcal{O}(k * n)$), whereas n is the number of elementary modules and k is the number of elementary modules that are needed to be resized.

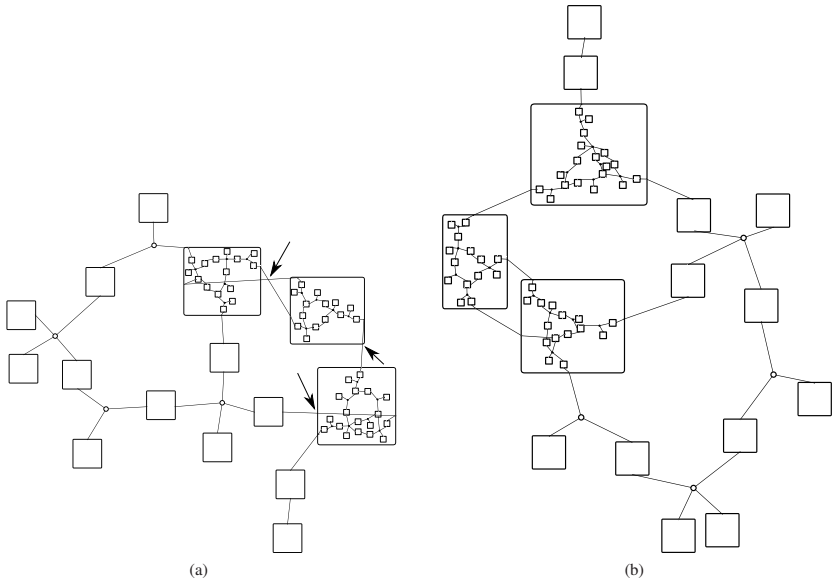


Figure 7.9.: Example graph with 73 nodes, 96 edges and 3 subgraphs. Applying (a) a local method [53] and (b) the HyCoGLa method. In (a) edge-vertex and edge-edge crossings may occur due to the local focus of the method (indicated as arrows). In contrast, in (b) the nodes are placed in an improved way across different hierarchical levels; edge-edge crossings are minimized.

7.3.2. Applications

Figure 7.11 shows, exemplarily using the `Toy` model, how the `FlatLayout` algorithm performs. For that, the model is laid out, applying the original hierarchical layout. Thereby, elementary modules on deeper hierarchical levels are difficult to see or are even not visible (because of their small size). As shown in Figure 7.11b a `FlatLayout` of the same model results in a uniform scaling of all species and gates across different hierarchical levels (cf. elementary modules `k3p2` and `tf1`). The size of a composite module is computed by taking the bounding box of all inner modules. A large model with applied `FlatLayout` is depicted in Appendix B, Figure D.1.

In Figure 7.12, a modular model with high ‘structural symmetry’ is shown. The modular, hierarchical layout (Figure 7.12a) is transformed to a flat layout (Figure 7.12b). As depicted, the algorithm is also able to preserve network properties such as orthogonal ordering or proximity even across different hierarchical levels and module borders.

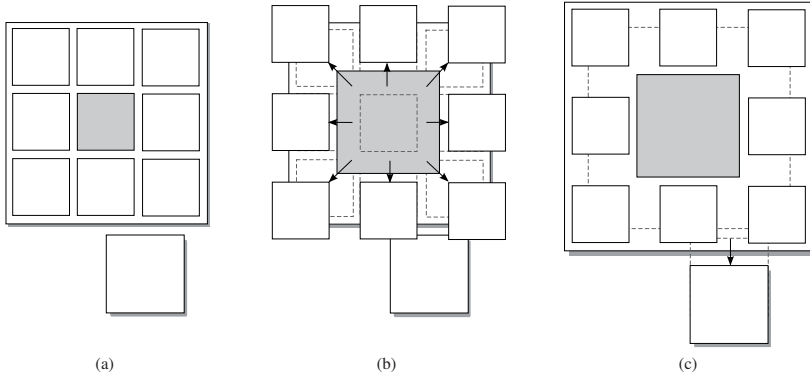


Figure 7.10.: Important steps of the `FlatLayout` algorithm applied to an example graph without edges. (a) Graph before any transformation is applied. (b) The elementary module colored in gray is increased in size according to the computed scale factor. Surrounding modules are translated using the computed translation vectors. (c) The bounding box of all inner modules is computed and the bounds of the parent module are adjusted accordingly. Modules outside the parent module are recursively translated in order to avoid node overlapping. Translation vectors in (b) and (c) are depicted as arrows; previous states of modules are depicted as rectangles with a dashed border.

7.4. Conclusions

In this chapter two layout strategies, `HyCoGLa` and `FlatLayout`, have been presented. They have been developed to layout modular models; that is, optimizing the positions of elements on different hierarchical levels.

`HyCoGLa` is based on the spring-embedder paradigm. It tries to minimize the manual effort in the process of layout. The key feature is a hybrid method which allows the advantageous combination of both local and global approaches. Consequently, the trade-off between layout quality and computational time can be optimized. The algorithm is implemented in the modeling tool `PROMOT` and is used to draw modular models of complex signaling systems. Thereby, different domain-specific constraints such as information about the direction of signal flow are incorporated in the layout process. This results in an improved representation of modular structures across different hierarchical levels and increases the readability of the network model. Moreover, the layout process can be constrained by different options adjustable by the modeler.

Although it is desirable to work with structured models, it is still necessary to generate comprehensible, flat graphical representations for use in traditional media, for example, for publications, presentations or documentation. Moreover, it is essential to flatten the visual representation of modular models in order to make visual representations available for tools not supporting modular models. For this purpose, the `FlatLayout` was developed. With this layout strategy any graphical model depicted by a single integrated view in `VISUAL EXPLORER` can be converted to a flat graphical representation where each

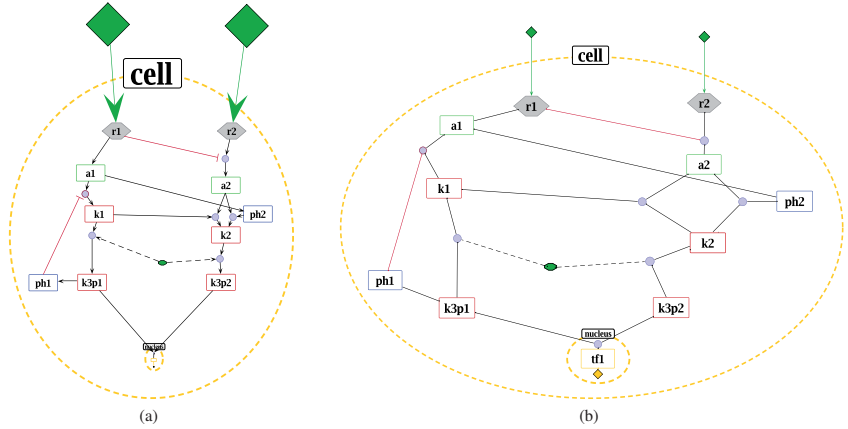


Figure 7.11.: FlatLayout applied to the Toy model. The hierarchical structure of the Toy model is depicted in Figure 7.2; it is the same for both layouts. (a) Original modular layout with scaled size of elementary modules; (b) FlatLayout with uniform size of elementary modules; proximity and orthogonal ordering are preserved.

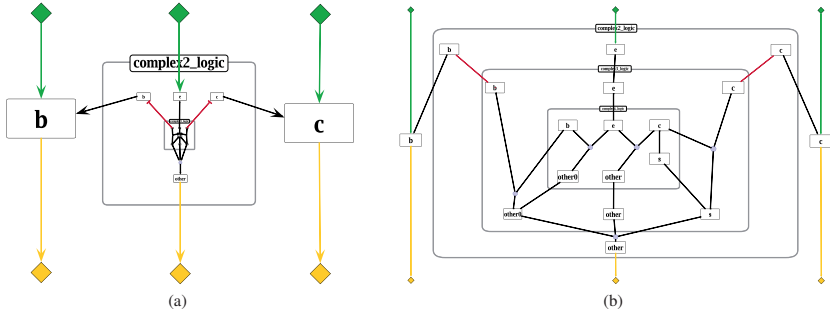


Figure 7.12.: Manual, modular layout versus FlatLayout: example network based on a logical model with a high symmetry in structure and the layout. (a) Manual, modular layout with scaled size of elementary modules and, (b) the derived layout with uniform size of elementary modules using the FlatLayout method. It is shown in the final layout how proximity information are preserved even across hierarchical levels and module borders.

elementary module is uniform in size. Similar to the HyCoGLa layout, the FlatLayout explicitly considers the modular and hierarchical properties during the layout process. Thereby, the layout adjustments are smoothly animated in order to maintain the visual context.

7. Layout Strategies for Modular Models

The visual representation produced by the `FlatLayout` has several advantages: For instance, elementary modules located at different hierarchical levels have the same size and their relations to each other, for example, emphasized by a focus path, can be visualized in an intuitive way. This flat representation is also comparable to the visual representations used in most network tools, and thus is familiar to most modelers. However, due to the unification of the elements sizes, the space allocated by the network may increase strongly. Here, techniques for layout compaction [45] might be an appropriate solution. Applying such techniques for reducing the space between elementary modules, the layout area can be minimized.

Both approaches use animations to smoothly transform the previous layout state into the new state. This provides immediate feedback about the current layout progress and the effect of different layout settings. In this way, visual context is maintained and the mental map is preserved.

Future advances in science depend on our ability to comprehend the vast amounts of data being produced and acquired, and scientific visualization is a key enabling technology in this endeavor. . . . visualization should be better integrated with the data exploration process instead of being done after the fact – when all the science is done – simply to generate presentations of the findings.

Claudio T. Silva and Juliana Freire [178]

CHAPTER 8

Concluding Remarks

In systems biology, as in many other disciplines, visualizations play a pivotal role since it is a common basis for communication and transfer of knowledge¹. More specifically, graphical representation can assist the modeler in exploring and analyzing complex biochemical datasets, and in answering related questions. Thus, it is not surprising that this area is becoming more and more important – also reflected in an increasing number of conferences and journal issues. However, the area is still in its infancy. Large amounts of data require novel visualization and exploration techniques; comprehensible graphical representations are essential. Visualizations need standardization, but also customization towards the specific modeling task or individual preferences of the modeler. This requires a highly flexible handling of visual properties of the model.

In this thesis, it is discussed how dynamic as well as logical models of signaling processes can be depicted by adequate visual representations. Modularity as a fundamental property of models allows models to be dealt with in a systematic and engineering-like fashion. In this context relevant questions are addressed, for example: How complex structures of modular models and their properties be visualized and interactively explored? How can they be laid out in a hierarchical or flat representation? This thesis provides some answers. The overall motivation is to support modelers in setup and analysis of modular models.

This thesis makes several valuable contributions that are summarized in the following section. Thereafter, promising future directions are discussed.

8.1. Scientific Contributions

In the following, significant contributions of this work are given: First, in close cooperation with modelers, formalized visual representations for dynamic and logical models have been specified. They provide a symbol set and adequate visualizations for models defined in the modeling language MDL. This also implies the visual handling of complex modular and hierarchical structures in a systematic way (Chapter 4). Especially for the logical modeling formalism, the new visual representation aims at visualizing complex signaling systems in a biologically intuitive manner, but also at maintaining their systematic approach. The visual representation improves the workflow of model analysis in the context of logical

¹For instance, visual representations are often used on scientific conferences as well as in talks or publications.

8. Concluding Remarks

models with modular structure. This was described by using the example of multiple maps in *PROMOT* and *CELLNETANALYZER*. The formalized visual representations for dynamic and logical models are encapsulated in two global VSs – VS ‘Dynamic Model’ and VS ‘Logical Model’.

Second, a zoomable multiscale visual interface for modular models has been developed to support highly interactive tasks such as navigation through complex model structures and exploratory analysis. Hereby, interactive exploration schemes, semantic zooming as well as content-aware navigation are used and combined in order to integrate focus and contextual information, and efficiently navigate and explore the modular model (Chapter 5). In particular, different exploration schemes such as *InteractiveZoom*, *HierarchicalZoom* and *ContextualZoom* are derived from successful, existing solutions and are adapted to modular models and the domain of systems biology. They provide an intuitive and flexible data exploration. Through interactive exploration, relationships or correlations between biological entities or more complex units can be analyzed and novel hypotheses can be investigated. Semantic zooming and *Focus+Context* techniques reduce visual complexity by presenting only a relevant subset of the model. This results in less cluttered and more descriptive visualizations.

Third, different visual scenarios have been developed. Thereby, the trade-off between standardization and customization is realized by defining two different scenario types: global and local VSs. Applying VSs, the modeler is able to modify the representation of model elements in order to highlight, suppress or ignore them, depending on context, structure or search criteria. Thus, the visual representation can be designed towards a specific modeling task, given datasets and individual preferences (Chapter 6). An interesting idea is to take different additional aspects into account that directly affect the visual appearance of the network view, including exploration scheme and layout strategy.

VSs graphically aid the user in specific modeling tasks, for instance, in the visual interpretation of analysis results. External data can be visualized very flexibly in the context of the modular model, for example, by emphasizing species that were active in a particular experimental condition. Showing such external data along with the network structure in a single view supports modelers in interpreting underlying signaling processes.

Finally, two layout strategies, *HyCoGLa* and *FlatLayout*, have been developed. Both are dedicated to drawing complex signaling networks and to work with modular models (Chapter 7). The *HyCoGLa* layout algorithm is a new approach for laying out modular models that are structured by an arbitrary number of hierarchical levels. Thereby, it combines the advantages of local and global layout approaches by considering elements from adjacent hierarchical levels in each traversal step. *HyCoGLa* is based on the spring-embedder paradigm. Different domain-specific constraints are incorporated in *HyCoGLa* which improve the representation of modular structures and increase the readability of the network model. Results suggest that *HyCoGLa* is able to produce comprehensible visualizations for signaling models with modular structure. The *HyCoGLa* approach explicitly represents the modular and hierarchical structure. In contrast to that the *FlatLayout* provides a way to generate comprehensible, flat graphical representations. The single integrated view is flattened by unifying the size of elementary modules. Thus, the modular structure is still encoded in the view but de-emphasized. These representations can be used in the further workflow, for example, in the creation of publication-ready graphics, documentation and presentation of models, or integration with other tools.

In summary, the developed methods for exploring and visualizing modular models in systems biology assist the modeler in extracting biological meaning, understanding the system under investigation and formulating novel hypotheses. The developed methods stated above and their combination allow the modeler to smoothly and interactively explore and navigate model parts of interest in the context of the modular model as a whole. In addition, they facilitate the creation of versatile visualizations, which can be presented at interdisciplinary meetings or scientific conferences in order to stimulate discussions.

Thus, the methods presented in this thesis are a valuable contribution to addressing the challenges stated in recent reviews [69, 141, 170, 188].

The applicability of the developed methods was demonstrated by application to large modular models of signaling processes (e. g., EGF or TCR model) and confirmed by several modelers using the methods in their daily research.

A major contribution of this research is the software component VISUAL EXPLORER developed as integral part of the modeling tool PROMOT. VISUAL EXPLORER contains the software-based implementation of most of the theoretical and conceptual work presented in this thesis. PROMOT is installed at the MPI Magdeburg and can be used by all researchers within the institute. The developed methods are actively applied in several modeling projects in order to visualize and explore complex structures of modular models in systems biology. In particular, dynamic and logical models of signal transduction systems [164, 166, 167] are applied. Furthermore, it is used by members of different research initiatives. PROMOT is available for download and available for interested researchers world wide. The source code is distributed under an open-source software license.

8.2. Open Issues and Research Directions

There are several directions for future research in the context of this thesis. In the following, some of the promising directions are discussed.

First, there are several open research questions related to the application of rich semantic information in the navigation and exploration of modular models. In the near future, the available semantic information for models in systems biology will increase rapidly [119]. These metadata can also be used for improving and fine-tuning the developed navigation and exploration techniques and at the end helps to understand the underlying structure and processes. For instance, it might be useful to explore a certain protein family different from another with semantic zooming. In the same way, semantic information can be used to show different views of the same part of the model. Analogous to the smooth integration of overview and detailed view by using multiple cameras (cf. Chapter 5, Figure 5.12), the view of model parts can be visually modified by the approach of *Semantic Lenses* [24]. More specifically, a local view defined by a region of interest is altered by a filter operator while the remaining parts are left unchanged. This can also be enriched with semantic information. For example, the modeler might interactively move the magic lens over the modular model and the operator filters out all information deeper than a certain hierarchical level.

A second, very related area that could be investigated by future work is the visualization of simulation results. In principle, simulation results are semantic information that can be mapped back to the visual representation of the modular model. In Chapter 6, Section 6.4.1 it was already shown how this can be performed with different analysis results (e. g., from CELLNETANALYZER). However, this should also be possible for results of dynamic simulations, sensitivity analysis or optimization. Often, it is necessary to explore the dynamic behavior of a specific part of the model (e. g., a composite module) before investigating the overall model dynamics. The simulation results of a local model part should be integrated graphically in the context of the modular model. Furthermore, there are a number of open questions regarding the layout of modular models. It might be interesting to improve the network layout with the results of logical analysis or dynamic behavior.

A third area of future research is the extension of the visual scenario approach described in Chapter 6, Section 6.2. Two starting points are likely to investigate: mapping and the trade-off between standardization and customization. Often a limitation is the provided one-to-one mapping; that is, a single value is mapped to one visual property. Sometimes it is necessary to map multi-dimensional metadata to a one-dimensional visual property. This can be performed by aggregating several metadata (e. g., combined

8. Concluding Remarks

by different weights), and then mapping from this resulting single value to the visual property. Another issue is the trade-off between customization and standardization. It might be possible and reasonable to provide three levels of 'freedom': local VSs defined by the modeler, global VSs implemented by the tool developers and standardized VSs defined by the systems biology community like SBGN. In this way, the modeler can choose the appropriate visualization scheme for the current modeling task. Of course, related conversion strategies between the different VSs types must be provided.

In conclusion, as stated in the saying at the beginning of this chapter, visualization should not be only the result of the modeling and analysis process, but also an important part of the exploration and visualization step within that process.

- [1] ADAR, E. 2006. GUESS: a language and interface for graph exploration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, 791–800. 49
- [2] ADEREM, A. 2005. Systems Biology: its practice and challenges. *Cell* 4, 121, 511–513. 10
- [3] ALBERGHINA, L. AND WESTERHOFF, H. V. 2005. *Systems biology: definitions and perspectives*. Topics in Current Genetics, vol. 13. Springer. 16
- [4] ALBRECHT, M., KERREN, A., KLEIN, K., KOHLBACHER, O., MUTZEL, P., PAUL, W., SCHREIBER, F., AND WYBROW, M. 2010. On open problems in biological network visualization. In *Graph Drawing*. Lecture Notes in Computer Science, vol. 5849. Springer, 256–267. 2
- [5] ALDRIDGE, B. B., BURKE, J. M., LAUFFENBURGER, D. A., AND SORGER, P. K. 2006. Physicochemical modelling of cell signalling pathways. *Nature Cell Biology* 8, 11, 1195–1203. 10, 11, 13, 24
- [6] ALON, U. 2007. Network motifs: theory and experimental approaches. *Nature Review Genetics* 8, 6, 450–461. 8
- [7] ALVES, R., ANTUNES, F., AND SALVADOR, A. 2006. Tools for kinetic modeling of biochemical networks. *Nature Biotechnology* 24, 6, 667–672. 18, 42
- [8] ASCHENBRENNER, E. 2008. Layout strukturierter Modelle am Beispiel von Daten aus der Systembiologie. M.S. thesis, Department of Simulation and Graphics, Otto-von-Guericke University, Magdeburg. 128, 130, 134, 136, 185
- [9] ASSAULT SYSTEMES. 2011. Dymola. <http://www.3ds.com/products/catia/portfolio/dymola/>. On-line; accessed November 24, 2011. 18
- [10] BABU, M. M., LUSCOMBE, N. M., ARAVIND, L., GERSTEIN, M., AND TEICHMANN, S. A. 2004. Structure and evolution of transcriptional regulatory networks. *Current Opinion in Structural Biology* 14, 3, 283–291. 15
- [11] BARSKY, A., GARDY, J. L., HANCOCK, R. E. W., AND MUNZNER, T. 2007. Cerebral: a Cytoscape plugin for layout of and interaction with biological networks using subcellular localization annotation. *Bioinformatics* 23, 8, 1040–1042. 131

- [12] BARTRAM, L., HO, A., DILL, J., AND HENIGMAN, F. 1995. The continuous zoom: a constrained fisheye technique for viewing and navigating large information spaces. In *Proceedings of the 8th annual ACM Symposium on User Interface and Software Technology*. ACM Press, 207–215. 41, 80
- [13] BARTRAM, L., WARE, C., AND CALVERT, T. 2001. Moving icons: detection and distraction. In *Proceedings of the International Conference on Human-Computer Interaction*. IOS Press, 157–165. 41
- [14] BATAGELJ, V. AND MRVAR, A. 2003. Pajek – analysis and visualization of large networks. In *Graph Drawing Software*, P. Mutzel, M. Jünger, and S. Leipert, Eds. Lecture Notes in Computer Science, vol. 2265. Springer, 77–103. 126
- [15] BEARD, D. V. AND WALKER, J. Q. 1990. Navigational techniques to improve the display of large two-dimensional spaces. *Behavioral Information Technology* 6, 9, 451–466. 42
- [16] BECK, F., BURCH, M., AND DIEHL, S. 2009. Towards an aesthetic dimensions framework for dynamic graph visualisations. In *Proceedings of the International Conference Information Visualisation*. IV '09. IEEE Computer Society, 592–597. 30
- [17] BEDERSON, B. B. AND BOLTMAN, A. 1999. Does animation help users build mental maps of spatial information? In *Information Visualization*. IEEE Computer Society, 28–35. 41
- [18] BEDERSON, B. B., GROSJEAN, J., AND MEYER, J. 2004. Toolkit design for interactive structured graphics. *IEEE Transactions on Software Engineering* 30, 8, 535–546. 49
- [19] BEDERSON, B. B. AND HOLLAN, J. D. 1994. PAD++: a zooming graphical interface for exploring alternate interface physics. In *Proceedings of the 7th annual ACM Symposium on User Interface Software and Technology*. ACM Press, 17–26. 49
- [20] BEDERSON, B. B., MEYER, J., AND GOOD, L. 2000. Jazz: an extensible zoomable user interface graphics toolkit in java. In *Proceedings of the 13th annual ACM Symposium on User Interface Software and Technology*. ACM Press, 171–180. 49
- [21] BERGMANN, F. T. AND SAURO, H. M. 2006. SBW – a modular framework for systems biology. In *Proceedings of the 38th Conference on Winter Simulation*. WSC '06. IEEE Computer Society, 1637–1645. 49
- [22] BERTAULT, F. AND MILLER, M. 1999. An algorithm for drawing compound graphs. In *Proceedings of the 7th International Symposium on Graph Drawing*. GD '99. Springer, 197–204. 35, 126
- [23] BERTIN, J. 1967. *Sémiologie graphique: les diagrammes – les réseaux – les cartes*. Editions de l'Ecole des Hautes Etudes en Sciences. 38
- [24] BIER, E. A., STONE, M. C., PIER, K., BUXTON, W., AND DEROSE, T. D. 1993. Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '93. ACM Press, 73–80. 143
- [25] BIERMANN, S., GU, Y., SCHUMANN, H., AND UHRMACHER, A. 2005. Information visualization in systems biology. *Rostocker Informatik-Berichte* 29, 5–17. 10, 11

- [26] BOURQUI, R. AND WESTENBERG, M. A. 2009. Visualizing temporal dynamics at the genomic and metabolic level. In *Information Visualisation*. IEEE Computer Society, 317–322. 63, 115, 118, 119
- [27] BRADFORD, R. 2003. A man, a worm, and a nobel. *Salk Signals* 5, 2, 12–17. 1
- [28] BRANDENBURG, F. J., FORSTER, M., PICK, A., RAITNER, M., AND SCHREIBER, F. 2004. BioPath – exploration and visualization of biochemical pathways. In *Graph Drawing Software*, M. Jünger and P. Mutzel, Eds. Springer Mathematics and Visualization Series. Springer, 215–236. 35
- [29] BREITLING, R., DONALDSON, R., GILBERT, D. R., AND HEINER, M. 2010. Biomodel engineering – from structure to behavior. *Transactions on Computational Systems Biology* 12, 1–12. 12, 17, 20
- [30] CALZONE, L., GELAY, A., ZINOVYEV, A., RADVANYI, F., AND BARILLOT, E. 2008. A comprehensive modular map of molecular interactions in RB/E2F pathway. *Molecular Systems Biology* 4, 173. 58
- [31] CARD, S. K., MACKINLAY, J. D., AND SHNEIDERMAN, B. 1999. *Readings in information visualization*. Morgan Kaufmann Publishers, Inc. 36, 37, 43, 45
- [32] CARD, S. K., MORAN, T. P., AND NEWELL, A. 1986. The model human processor: an engineering model of human performance. *Handbook of Perception and Human Performance* 2, 4, 35–45. 41
- [33] CARD, S. K., ROBERTSON, G. G., AND MACKINLAY, J. D. 1991. The information visualizer, an information workspace. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, 181–186. 41
- [34] CARVALHO, A., DE SOUSA, A. A., RIBEIRO, C., AND COSTA, E. 2008. A temporal focus+context visualization model for handling valid-time spatial information. *Information Visualization* 7, 3, 265–274. 45
- [35] CHI, E. H. 2000. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of the IEEE Symposium on Information Visualization*. IEEE Computer Society, 69–75. 37, 102, 103
- [36] CITRI, A. AND YARDEN, Y. 2006. EGF-ERBB signalling: towards the systems level. *Nature Review Molecular Cell Biology* 7, 7, 505–516. 8, 22
- [37] CLEVELAND, W. S. 1985. *The elements of graphing data*. Wadsworth Publishing Co. 38
- [38] CLINE, M. S., SMOOT, M., CERAMI, E., KUCHINSKY, A., LANDYS, N., WORKMAN, C., CHRISTMAS, R., AVILA-CAMPILO, I., CREECH, M., GROSS, B., HANSPERS, K., ISSERLIN, R., KELLEY, R., KILLCOYNE, S., LOTIA, S., MAERE, S., MORRIS, J., ONO, K., PAVLOVIC, V., PICO, A. R., VAILAYA, A., WANG, P.-L., ADLER, A., CONKLIN, B. R., HOOD, L., KUIPER, M., SANDER, C., SCHMULEVICH, I., SCHWIKOWSKI, B., WARNER, G. J., IDEKER, T., AND BADER, G. D. 2007. Integration of biological networks and gene expression data using Cytoscape. *Nature Protocols* 2, 10, 2366–2382. 49
- [39] COCKBURN, A., KARLSON, A., AND BEDERSON, B. B. 2008. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys* 41, 1, 1–31. 41, 42, 90

- [40] DAVIDSON, R. AND HAREL, D. 1996. Drawing graphs nicely using simulated annealing. *ACM Transaction on Graphics* 15, 4, 301–331. 34
- [41] DEL VECCHIO, D., NINFA, A. J., AND SONTAG, E. D. 2008. Modular cell biology: retroactivity and insulation. *Molecular Systems Biology* 4, 161, 1–16. 16
- [42] DEMIR, E., BABUR, O., DOGRUSÖZ, U., GÜRSOY, A., AYAZ, A., GULESIR, G., NISANCI, G., AND ÇETIN ATALAY, R. 2004. An ontology for collaborative construction and analysis of cellular pathways. *Bioinformatics* 20, 3, 349–356. 7
- [43] DEMIR, E., BABUR, O., DOGRUSÖZ, U., GÜRSOY, A., NISANCI, G., ÇETIN ATALAY, R., AND ÖZTURK, M. 2002. PATIKA: an integrated visual environment for collaborative construction and analysis of cellular pathways. *Bioinformatics* 18, 7, 996–1003. 49
- [44] DEMIR, E., CARY, M. P., PALEY, S., FUKUDA, K., LEMER, C., VASTRIK, I., WU, G., D'EUSTACHIO, P., SCHAEFER, C., LUCIANO, J., SCHACHERER, F., MARTINEZ-FLORES, I., HU, Z., JIMENEZ-JACINTO, V., JOSHI-TOPE, G., KANDASAMY, K., LOPEZ-FUENTES, A. C., MI, H., PICHLER, E., RODCHENKOV, I., SPLENDIANI, A., TKACHEV, S., ZUCKER, J., GOPINATH, G., RAJASIMHA, H., RAMAKRISHNAN, R., SHAH, I., SYED, M., ANWAR, N., BABUR, O., BLINOV, M., BRAUNER, E., CORWIN, D., DONALDSON, S., GIBBONS, F., GOLDBERG, R., HORNBECK, P., LUNA, A., MURRAY-RUST, P., NEUMANN, E., REUBENACKER, O., SAMWALD, M., VAN IERSEL, M., WIMALARATNE, S., ALLEN, K., BRAUN, B., WHIRL-CARRILLO, M., CHEUNG, K.-H., DAHLQUIST, K., FINNEY, A., GILLESPIE, M., GLASS, E., GONG, L., HAW, R., HONIG, M., HUBAUT, O., KANE, D., KRUPA, S., KUTMON, M., LEONARD, J., MARKS, D., MERBERG, D., PETRI, V., PICO, A., RAVENSCROFT, D., REN, L., SHAH, N., SUNSHINE, M., TANG, R., WHALEY, R., LETOVKSY, S., BUETOW, K. H., RZHETSKY, A., SCHACHTER, V., SOBRAL, B. S., DOGRUSOZ, U., MCWEENEY, S., ALADJEM, M., BIRNEY, E., COLLADO-VIDES, J., GOTO, S., HUCKA, M., LE NOVÈRE, N., MALTSEV, N., PANDEY, A., THOMAS, P., WINGENDER, E., KARP, P. D., SANDER, C., AND BADER, G. D. 2010. The BioPAX community standard for pathway data sharing. *Nature Biotechnology* 28, 9, 935–942. 1
- [45] DI BATTISTA, G., EADES, P., TAMASSIA, R., AND TOLLIS, I. G. 1999. *Graph drawing: algorithms for the visualization of graphs*. Prentice Hall. 27, 28, 30, 33, 35, 126, 140
- [46] DÍAZ, J., PETIT, J., AND SERNA, M. 2002. A survey of graph layout problems. *ACM Computing Surveys* 34, 3, 313–356. 27
- [47] DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1, 1, 269–271. 113
- [48] DOGRUSOZ, U., GIRAL, E., CETINTAS, A., CIVRIL, A., AND DEMIR, E. 2004. A compound graph layout algorithm for biological pathways. In *Graph Drawing*. Lecture Notes in Computer Science, vol. 3383. Springer, 442–447. 35, 126
- [49] DOS SANTOS, S. AND BRODLIE, K. 2004. Gaining understanding of multivariate and multidimensional data through visualization. *Computers and Graphics* 28, 3, 311–325. 37
- [50] DROSTE, P., NOACK, S., NÖH, K., AND WIECHERT, W. 2009. Customizable visualization of multi-omics data in the context of biochemical networks. In *Proceedings of the International Conference in Visualisation*. VIZ '09. IEEE Computer Society, 21–25. 32, 49, 100, 101, 115

- [51] DROSTE, P., VON LIERES, E., WIECHERT, W., AND NÖH, K. 2010. Customizable visualization on demand for hierarchically organized information in biochemical networks. *Computational Modeling of Objects Represented in Images* 6026, 163–174. 101
- [52] DWYER, T. 2005. Two and a half dimensional visualisation of relational networks. Ph.D. thesis, The School of Information Technologies, University of Sydney, Sydney, Australia. 36
- [53] EADES, P. 1984. A heuristic for graph drawing. *Congressus Numerantium* 42, 149–160. 34, 126, 132, 133, 136, 137
- [54] EADES, P. AND FENG, Q. W. 1997. Multilevel visualization of clustered graphs. In *Graph Drawing*. Lecture Notes in Computer Science, vol. 1190. Springer, 101–112. 31
- [55] EADES, P. AND HUANG, M. L. 2000. Navigating clustered graphs using force-directed methods. *Graph Algorithms and Applications: Special Issue on Selected Papers from 1998* 4, 3, 157–181. 35, 126
- [56] EADES, P., LAI, W., MISUE, K., AND SUGIYAMA, K. 1991. Preserving the mental map of a diagram. In *Proceedings of Compugraphics*. Vol. 91. 24–33. 39
- [57] ELMQVIST, N. AND FEKETE, J. D. 2010. Hierarchical aggregation for information visualization: overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics* 16, 3, 439–454. 2, 56
- [58] FELL, D. A. AND WAGNER, A. 2000. The small world of metabolism. *Nature Biotechnology* 18, 11, 1121–1122. 109
- [59] FRICK, A., LUDWIG, A., AND MEHLDAU, H. 1994. A fast adaptive layout algorithm for undirected graphs. In *Proceedings of the DIMACS International Workshop on Graph Drawing*. Lecture Notes in Computer Science, vol. 894. Springer, 388–403. 34, 128
- [60] FRISCH, M., DACHSELT, R., AND BRÜCKMANN, T. 2008. Towards seamless semantic zooming techniques for UML diagrams. In *Proceedings of the 4th ACM Symposium on Software Visualization*. ACM Press, 207–208. 49, 107
- [61] FRITZSON, P. 2004. *Principles of object-oriented modeling and simulation with Modelica 2.1*. Wiley-IEEE Press. 18
- [62] FRUCHTERMAN, T. M. J. AND REINGOLD, E. M. 1991. Graph drawing by force-directed placement. *Software Practical Experiences* 21, 11, 1129–1164. 34
- [63] FUNAHASHI, A., MATSUOKA, Y., JOURAKU, A., MOROHASHI, M., KIKUCHI, N., AND KITANO, H. 2008. CellDesigner 3.5: a versatile modeling tool for biochemical networks. *Proceedings of the IEEE* 96, 8, 1254–1265. 49, 127
- [64] FURNAS, G. W. 1986. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '86. ACM Press, 16–23. 45
- [65] FURNAS, G. W. AND BEDERSON, B. B. 1995. Space-scale diagrams: understanding multiscale interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 234–241. 43, 44, 48, 89

Bibliography

- [66] GALPERIN, M. Y. AND FERNÁNDEZ-SUÁREZ, X. M. 2012. The 2012 nucleic acids research database issue and the online molecular biology database collection. *Nucleic Acids Research* 40, D1, D1–D8. 10
- [67] GARNY, A., NICKERSON, D. P., COOPER, J., SANTOS, R. W. D., MILLER, A. K., MCKEEVER, S., NIELSEN, P. M., AND HUNTER, P. J. 2008. CellML and associated tools and techniques. *Philosophical Transactions of the Royal Society – Series A: Mathematical, Physical and Engineering Sciences* 366, 1878, 3017–3043. 1, 18
- [68] GASTEIGER, E., GATTIKER, A., HOOGLAND, C., IVANYI, I., APPEL, R. D., AND BAIROCH, A. 2003. ExPASy: the proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Research* 31, 13, 3784–3788. 46
- [69] GEHLENBORG, N., O'DONOGHUE, S. I., BALIGA, N. S., GOESMANN, A., HIBBS, M. A., KITANO, H., KOHLBACHER, O., NEUWEGER, H., SCHNEIDER, R., TENENBAUM, D., AND GAVIN, A. C. 2010. Visualization of omics data for systems biology. *Nature Methods* 7, 3, S56–S68. 1, 2, 41, 45, 48, 143
- [70] GENC, B. AND DOGRUSOZ, U. 2006. A layout algorithm for signaling pathways. *Information Sciences* 176, 2, 135–149. 131
- [71] GILBERT, D., FUSS, H., GU, X., ORTON, R., ROBINSON, S., VYSEMIIRSKY, V., KURTH, M. J., DOWNES, C. S., AND DUBITZKY, W. 2006. Computational methodologies for modelling, analysis and simulation of signalling networks. *Briefings in Bioinformatics* 7, 4, 339–353. 8, 13
- [72] GILLES, E. D. 1998. Network theory for chemical processes. *Chemical Engineering and Technology* 21, 2, 121–132. 17
- [73] GINKEL, M., KREMLING, A., NUTSCH, T., REHNER, R., AND GILLES, E. D. 2003. Modular modeling of cellular systems with ProMoT/DIVA. *Bioinformatics* 19, 9, 1169–1176. 10, 11, 16, 17, 21, 22
- [74] GOLDSTEIN, E. B. 2001. *Sensation and perception*, 6 ed. Wadsworth Publishing Co. 38
- [75] GOOGLE INC. 2012. Google Maps. <http://www.google.de/maps/>. Online; accessed February 25, 2012. 89
- [76] GREEN, T. R. G. AND PETRE, M. 1996. Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *Journal of Visual Languages and Computing* 7, 131–174. 55
- [77] HADLAK, S., SCHULZ, H. J., AND SCHUMANN, H. 2011. In situ exploration of large dynamic networks. *IEEE Transactions on Visualization and Computer Graphics* 17, 12, 2334–2343. 2, 56
- [78] HANKE, R., MANGOLD, M., AND SUNDMACHER, K. 2005. Application of hierarchical process modelling strategies to fuel cell systems towards a virtual fuel cell laboratory. *Fuel Cells* 1, 5, 133–147. 19
- [79] HARTWELL, L. H., HOPFIELD, J. J., LEIBLER, S., AND MURRAY, A. W. 1999. From molecular to modular cell biology. *Nature* 402, 6761. 15, 16, 24
- [80] HELMS, V. 2008. *Principles of computational cell biology – from protein complexes to cellular networks*. Wiley-VCH. 46

- [81] HERMAN, I., MELANÇON, G., AND MARSHALL, M. S. 2000. Graph visualization and navigation in information visualization: a survey. *IEEE Transactions on Visualization and Computer Graphics* 6, 1, 24–43. 27, 30
- [82] 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. 49
- [83] HORNBEK, K., BEDERSON, B. B., AND PLAISANT, C. 2002. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM Transactions on Computer-Human Interaction* 9, 362–389. 43
- [84] HU, Z., MELLOR, J., WU, J., KANEHISA, M., STUART, J. M., AND DELISI, C. 2007. Towards zoomable multidimensional maps of the cell. *Nature Biotechnology* 25, 5, 547–554. 27, 28
- [85] HU, Z., NG, D. M., YAMADA, T., CHEN, C., KAWASHIMA, S., MELLOR, J., LINGHU, B., KANEHISA, M., STUART, J. M., AND DELISI, C. 2007. VisANT 3.0: new modules for pathway visualization, editing, prediction and construction. *Nucleic Acids Research* 35, 625–632. 49
- [86] HUCKA, M., FINNEY, A., SAURO, H. M., BOLOURI, H., DOYLE, J. C., KITANO, H., THE REST OF THE SBML FORUM: A. P. ARKIN, BORNSTEIN, B. J., BRAY, D., CORNISH-BOWDEN, A., CUELLAR, A. A., DRONOV, S., GILLES, E. D., GINKEL, M., GOR, V., GORYANIN, I. I., HEDLEY, W. J., HODGMAN, T. C., HOFMEYR, J.-H., HUNTER, P. J., JUTY, N. S., KASBERGER, J. L., KREMLING, A., KUMMER, U., LE NOVÈRE, N., LOEW, L. M., LUCIO, D., MENDES, P., MINCH, E., MJOLSNES, E. D., NAKAYAMA, Y., NELSON, M. R., NIELSEN, P. F., SAKURADA, T., SCHAFF, J. C., SHAPIRO, B. E., SHIMIZU, T. S., SPENCE, H. D., STELLING, J., TAKAHASHI, K., TOMITA, M., WAGNER, J., AND WANG, J. 2003. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 19, 524–531. 1, 18
- [87] JACOBS, T. AND MUSIAL, B. 2003. Interactive visual debugging with UML. In *Proceedings of the 2003 ACM Symposium on Software Visualization*. ACM Press, 115–122. 49
- [88] JANECEK, P. 2004. Interactive semantic fisheye views for information workspaces. Ph.D. thesis, School of Computer and Communication Sciences, Swiss Federal Institute of Technology, Lausanne, Switzerland. 37
- [89] JANECEK, P. AND PU, P. 2002. A framework for designing fisheye views to support multiple semantic contexts. In *International Conference on Advanced Visual Interfaces*. ACM Press, 51–58. 38, 39
- [90] JEONG, H., MASON, S. P., BARABASI, A. L., AND OLTVAI, Z. N. 2001. Lethality and centrality in protein networks. *Nature* 411, 6833, 41–42. 109
- [91] JUNKER, B., KLUKAS, C., AND SCHREIBER, F. 2006. VANTED: A system for advanced data analysis and visualization in the context of biological networks. *BMC Bioinformatics* 7, 109. 49
- [92] KAMADA, T. AND KAWAI, S. 1989. An algorithm for drawing general undirected graphs. *Information Processing Letters* 31, 1, 7–15. 34
- [93] KANEHISA, M. AND GOTO, S. 2000. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research* 28, 27–30. 1

- [94] KAUFMANN, M. AND WAGNER, D. 2001. Drawing graphs, methods and models. In *Drawing Graphs*, M. Kaufmann and D. Wagner, Eds. Lecture Notes in Computer Science, vol. 2025. Springer. 30, 33, 34
- [95] KAUFMANN, M. AND WIESE, R. 2002. Maintaining the mental map for circular drawings. In *Revised Papers from the 10th International Symposium on Graph Drawing*, GD '02. Springer, 12–22. 33
- [96] KEIM, D., ANDRIENKO, G., FEKETE, J. D., GÖRG, C., KOHLHAMMER, J., AND MELANÇON, G. 2008. Visual analytics: Definition, process, and challenges. In *Information Visualization*, A. Kerren, J. T. Skasko, J. D. Fekete, and C. North, Eds. Lecture Notes in Computer Science, vol. 4950. Springer, 154–175. 36
- [97] KERREN, A., EBERT, A., AND MEYER, J. 2007. *Human-centered visualization environments*. Lecture Notes in Computer Science, vol. 4417. Springer. 36, 47
- [98] KIM, J. S., YUN, H., KIM, H. U., CHOI, H., KIM, T. Y., WOO, H. M., AND LEE, S. Y. 2006. Resources for systems biology research. *Journal of Microbiological Biotechnology* 16, 832–848. 18
- [99] KITANO, H. 2002a. Computational systems biology. *Nature* 420, 6912, 206–210. 9
- [100] KITANO, H. 2002b. Standards for modeling. *Nature Biotechnology* 20, 4, 337–337. 17
- [101] KITANO, H. 2003. A graphical notation for biochemical networks. *BIOSILICO* 1, 5, 169–176. 46
- [102] KITANO, H., FUNAHASHI, A., MATSUOKA, Y., AND ODA, K. 2005. Using process diagrams for the graphical representation of biological networks. *Nature Biotechnology* 23, 8, 961–966. 27
- [103] KLAMT, S., SAEZ-RODRIGUEZ, J., AND GILLES, E. D. 2007. Structural and functional analysis of cellular networks with CellNetAnalyzer. *BMC Systems Biology* 1, 2, 1–13. 19, 20
- [104] KLAMT, S., SAEZ-RODRIGUEZ, J., LINDQUIST, J., SIMEONI, L., AND GILLES, E. D. 2006. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics* 7, 56. 14, 19, 64, 68
- [105] KLIPP, E., LIEBERMEISTER, W., WIERLING, C., KOWALD, A., LEHRACH, H., AND HERWIG, R. 2009. *Systems biology: a textbook*. Wiley-VCH. 12, 14
- [106] KOCH, I., REISIG, W., AND SCHREIBER, F. 2011. *Modeling in systems biology: the Petri net approach*. Springer Book Series Computational Biology. Springer. 12, 46
- [107] KOHN, K. W. 1999. Molecular interaction map of the mammalian cell cycle control and DNA repair systems. *Molecular Biology of the Cell* 10, 8, 2703–2734. 46
- [108] KOTH, O. AND MINAS, M. 2002. Structure, abstraction, and direct manipulation in diagram editors. In *Proceedings of the Second International Conference on Diagrammatic Representation and Inference*. Springer, 290–304. 49
- [109] KRASNYK, M., BONDAREVA, K., MILOKHOV, O., TEPLINSKIY, K., GINKEL, M., AND KIENLE, A. 2006. The ProMoT/Diana simulation environment. *Proceedings of the 16th European Symposium on Computer Aided Process Engineering and 9th international Symposium on Process Systems Engineering*, 445–450. 19, 20

- [110] KREMLING, A. AND GILLES, E. D. 2001. The organization of metabolic reaction networks: II. signal processing in hierarchical structured functional units. *Metabolic Engineering* 3, 2, 138–150. 17, 19, 20
- [111] KUMAR, H., PLAISANT, C., AND SHNEIDERMAN, B. 1995. Browsing hierarchical data with multi-level dynamic queries and pruning. *International Journal of Human-Computer Studies* 46, 103–124. 100
- [112] KÖHLER, J., BAUMBACH, J., TAUBERT, J., SPECHT, M., SKUSA, A., RÜEGG, A., RAWLINGS, C., VERRIER, P., AND PHILIPPI, S. 2006. Graph-based analysis and visualization of experimental results with ONDEX. *Bioinformatics* 22, 11, 1383–1390. 49
- [113] LAUFFENBURGER, D. A. 2000. Cell signaling pathways as control modules: complexity for simplicity? *Proceedings of the National Academy of Sciences of the United States of America* 97, 10, 5031–5033. 15
- [114] LE NOVÈRE, N., HUCKA, M., MI, H., MOODIE, S., SCHREIBER, F., SOROKIN, A., DEMIR, E., WEGNER, K., ALADJEM, M. I., WIMALARATNE, S. M., BERGMAN, F. T., GAUGES, R., GHAZAL, P., KAWAJI, H., LI, L., MATSUOKA, Y., VILLEGGER, A., BOYD, S. E., CALZONE, L., COURTOT, M., DOGRUSOZ, U., FREEMAN, T. C., FUNAHASHI, A., GHOSH, S., JOURAKU, A., KIM, S., KOLPAKOV, F., LUNA, A., SAHLE, S., SCHMIDT, E., WATTERSON, S., WU, G., GORYANIN, I., KELL, D. B., SANDER, C., SAURO, H., SNOEP, J. L., KOHN, K., AND KITANO, H. 2009. The systems biology graphical notation. *Nature Biotechnology* 27, 8, 735–741. 2, 46, 49, 100
- [115] LEE, Y. Y., LIN, C. C., AND YEN, H. C. 2006. Mental map preserving graph drawing using simulated annealing. In *Proceedings of the Asia Pacific Symposium on Information Visualization*. APVis '06, vol. 60. Australian Computer Society, 179–188. 40
- [116] LEUNG, Y. K. AND APPERLEY, M. D. 1994. A review and taxonomy of distortion-oriented presentation techniques. *ACM Trans. Comput.-Hum. Interact.* 1, 2, 126–160. 101
- [117] 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. 1
- [118] LI, W., EADES, P., AND NIKOLOV, N. 2005. Using spring algorithms to remove node overlapping. In *Proceedings of the 2005 Asia-Pacific Symposium on Information Visualisation*. Australian Computer Society, 131–140. 34, 129
- [119] LIEBERMAN, M. D., TAHERI, S., GUO, W., MIRRAHED, F., YAHAV, I., ARIS, A., AND SHNEIDERMAN, B. 2011. Visual exploration across biomedical databases. *IEEE Transactions on Computational Biology and Bioinformatics* 8, 2, 536–550. 143
- [120] LLOYD, C. M., HALSTEAD, M. D. B., AND NIELSEN, P. F. 2004. CellML: its future, present and past. *Progress in Biophysics and Molecular Biology* 85, 2-3, 433–450. 18
- [121] MACHNÉ, R. 2011. SBML ODE Solver Library. <http://www.tbi.univie.ac.at/~raim/odeSolver/>. Online; accessed October 4, 2011. 49
- [122] MAIWALD, T. AND TIMMER, J. 2008. Dynamical modeling and multi-experiment fitting with PottersWheel. *Bioinformatics* 24, 18, 2037–2043. 10

Bibliography

- [123] MALLAVARAPU, A., THOMSON, M., ULLIAN, B., AND GUNAWARDENA, J. 2009. Programming with models: modularity and abstraction provide powerful capabilities for systems biology. *Journal of The Royal Society Interface* 6, 32, 257–270. 18
- [124] MANGOLD, M., ANGELES-PALACIOS, O., GINKEL, M., WASCHLER, R., KIENLE, A., AND GILLES, E. D. 2005. Computer aided modeling of chemical and biological systems – methods, tools, and applications. *Industrial and Engineering Chemistry Research* 44, 2579–2591. 17
- [125] MANGOLD, M., GINKEL, M., AND GILLES, E. D. 2004. A model library for membrane reactors implemented in the process modelling tool ProMoT. *Computers & Chemical Engineering* 28, 3, 319–332. 19
- [126] MICHAL, G. 1999. *Biochemical pathways*. Spektrum Akademischer Verlag. 46
- [127] MILO, R., SHEN-ORR, S., ITZKOVITZ, S., KASHTAN, N., CHKLOVSKII, D., AND ALON, U. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594, 824–827. 8
- [128] MIREL, B. 2009. Supporting cognition in systems biology analysis: findings on users’ processes and design implications. *Journal of Biomedical Discovery and Collaboration* 4, 1, 2. 48
- [129] MIRSCHEL, S., STEINMETZ, K., AND KREMLING, A. 2009. Two different approaches for modeling biochemical reaction networks with ProMoT. In *Proceedings of the 6th International Conference of Mathematical Modelling*. MATHMOD ’09, vol. 35. ARGESIM, 2322–2330. 13, 21, 22
- [130] MIRSCHEL, S., STEINMETZ, K., REMPEL, M., GINKEL, M., AND GILLES, E. D. 2009. ProMoT: modular modeling for systems biology. *Bioinformatics* 25, 5, 687–689. 19, 22
- [131] MISUE, K., EADES, P., LAI, W., AND SUGIYAMA, K. 1995. Layout adjustment and the mental map. *Journal of Visual Languages and Computing* 6, 2, 183–210. 40, 41, 127
- [132] MODELICA CONSORTIUM. 2012. OpenModelica. <http://www.openmodelica.org>. Online; accessed January 16, 2012. 18
- [133] MOHL, K. D., SPIEKER, A., KÖHLER, R., GILLES, E. D., AND ZEITZ, M. DIVA – a simulation environment for chemical engineering applications. In *Proceedings of Informatics, Cybernetics and Computer Science*. ICCS ’97. Donetsk State Technical University, 8–15. 19
- [134] MOSCOVICH, T., CHEVALIER, F., HENRY, N., PIETRIGA, E., AND FEKETE, J. D. 2009. Topology-aware navigation in large networks. In *Proceedings of the 27th international Conference on Human Factors in Computing Systems*. ACM Press, 2319–2328. 82, 95
- [135] MOWBRAY, T. J. AND RUH, W. A. 1998. *Inside CORBA – distributed object standards and applications*. Addison-Wesley object technology series. Addison-Wesley-Longman. 22
- [136] NEUMANN, P., SCHLECHTWEIG, S., AND CARPENDALE, S. 2005. ArcTrees: visualizing relations in hierarchical data. In *Data Visualization 2005, Eurographics/IEEE VGTC Symposium on Visualization Symposium Proceedings*, K. W. Brodlie, D. J. Duke, and K. I. Joy, Eds. *Simulation*, 53–60. 31
- [137] NICHOLSON, D. E. 2000. The evolution of the IUBMB-Nicholson maps. *IUBMB Life* 50, 6, 341–344. 46

- [138] NICKERSON, D., NASH, M., NIELSEN, P., SMITH, N. P., AND HUNTER, P. J. 2006. Computational multiscale modeling in the IUPS physiome project: modeling cardiac electromechanics. *IBM Journal of Research and Development* 50, 6, 617–630. 18
- [139] NOACK, A. AND LEWERENTZ, C. 2005. A space of layout styles for hierarchical graph models of software systems. In *Proceedings of the 2005 ACM Symposium on Software Visualization*. ACM Press, 155–164. 35
- [140] ODA, K., MATSUOKA, Y., FUNAHASHI, A., AND KITANO, H. 2005. A comprehensive pathway map of epidermal growth factor receptor signaling. *Molecular Systems Biology* 1, 1, E1–E17. 127
- [141] O'DONOGHUE, S. I., GAIN, A.-C., GEHLENBORG, N., GOODSSELL, D. S., HERICHE, J.-K., NIELSEN, CYDNEY B NORTH, C., OLSON, A. J., PROCTER, J. B., SHATTUCK, D. W., WALTER, T., AND WONG, B. 2010. Visualizing biological data – now and in the future. *Nature Methods* 7, 3, S2–S4. 143
- [142] OLIVIER, B. G. AND SNOEP, J. L. 2004. Web-based kinetic modelling using JWS Online. *Bioinformatics* 20, 13, 2143–2144. 1
- [143] OMOTE, H. AND SUGIYAMA, K. 2007. Method for visualizing complicated structures based on unified simplification strategy. *IEICE – Transactions on Information and Systems* 90-D, 10, 1649–1656. 30
- [144] PALMER, S. E. AND ROCK, I. 1994. Rethinking perceptual organization: the role of uniform connectedness. *Psychonomic Bulletin & Review* 1, 1, 29–55. 38
- [145] PAVLOPOULOS, G., WEGENER, A. L., AND SCHNEIDER, R. 2008. A survey of visualization tools for biological network analysis. *BioData Mining* 1, 1, 12. 2, 48
- [146] PERLIN, K. AND FOX, D. 1993. Pad: an alternative approach to the computer interfaces. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '93. ACM Press, 57–64. 49
- [147] PETTINEN, A., AHO, T., SMOLANDER, O. P., MANNINEN, T., SAARINEN, A., TAATTOLA, K.-L., YLI HARJA, O., AND LINNE, M. L. 2005. Simulation tools for biochemical networks: evaluation of performance and usability. *Bioinformatics* 21, 3, 357–363. 18
- [148] PICCOLO2D. 2011. Piccolo2D – A structured 2d graphics framework. <http://piccolo2d.org/>. Online; accessed December 8, 2011. 49
- [149] PIETRIGA, E. 2005. A toolkit for addressing HCI issues in visual language environments. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE Computer Society, 145–152. 49
- [150] PLAISANT, C., CARR, D., AND SHNEIDERMAN, B. 1995. Image browsers: taxonomy, guidelines, and informal specifications. *IEEE Software* 12, 21–32. 42
- [151] PRELLER, A., MUGNIER, M. L., AND CHEIN, M. 1998. Logic for nested graphs. *Computational Intelligence* 14, 3, 335–357. 127
- [152] PURCHASE, H. C. 2000. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interacting with Computers* 13, 2, 147–162. 33

- [153] QELI, E. 2007. Information visualization techniques for metabolic engineering. Ph.D. thesis, Faculty of Mathematics and Computer Science, Philipps University, Marburg, Germany. 38
- [154] QUACKENBUSH, J. 2006. Weighing our measures of gene expression. *Molecular Systems Biology* 2, 63. 10
- [155] RANDHAWA, R. 2008. Model composition and aggregation in macromolecular regulatory networks. Ph.D. thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA. 127
- [156] RANDHAWA, R., SHAFFER, C. A., AND TYSON, J. J. 2009. Model aggregation: a building-block approach to creating large macromolecular regulatory networks. *Bioinformatics* 25, 24, 3289–3295. 10, 17, 127
- [157] RAVASZ, E., SOMERA, A. L., MONGRU, D. A., OLTVAI, Z. N., AND BARABÁSI, A.-L. 2002. Hierarchical organization of modularity in metabolic networks. *Science* 297, 5586, 1551–1555. 16
- [158] REINHARD, T., MEIER, S., STOIBER, R., CRAMER, C., AND GLINZ, M. 2008. Tool support for the navigation in graphical models. In *Proceedings of the 30th International Conference on Software Engineering*. ACM Press, 823–826. 48, 50, 78, 80
- [159] ROBERTSON, G. G., CARD, S. K., AND MACKINLAY, J. D. 1993. Information visualization using 3D interactive animation. *Communications of the ACM – Special Issue on Graphical User Interfaces* 36, 4, 57–71. 41
- [160] ROBINSON, J. A. AND FLORES, T. P. 1997. Novel techniques for visualizing biological information. In *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, 241–249. 45, 79
- [161] SAEZ-RODRIGUEZ, J. 2007. Modular analysis of signal transduction networks. Ph.D. thesis, Faculty of Process and Systems Engineering, Otto von Guericke University, Magdeburg, Germany. 24, 177
- [162] SAEZ-RODRIGUEZ, J., GAYER, S., GINKEL, M., AND GILLES, E. D. 2008. Automatic decomposition of kinetic models of signaling networks minimizing the retroactivity among modules. *Bioinformatics* 24, 16, i213–i219. 16, 125
- [163] SAEZ-RODRIGUEZ, J., KREMLING, A., CONZELMANN, H., BETTENBROCK, K., AND GILLES, E. D. 2004. Modular analysis of signal transduction networks. *IEEE Control Systems* 24, 4, 35–52. 16
- [164] SAEZ-RODRIGUEZ, J., KREMLING, A., AND GILLES, E. D. 2005. Dissecting the puzzle of life: modularization of signal transduction networks. *Computers & Chemical Engineering, Computational Challenges in Biology* 29, 3, 619–629. 15, 19, 22, 143
- [165] SAEZ-RODRIGUEZ, J., MIRSCHEL, S., HEMENWAY, R., KLAMT, S., GILLES, E. D., AND GINKEL, M. 2006. Visual setup of logical models of signaling and regulatory networks with ProMoT. *BMC Bioinformatics* 7, 506. 19, 22, 24, 58, 63, 70, 74, 174
- [166] SAEZ-RODRIGUEZ, J., SIMEONI, L., LINDQUIST, J. A., HEMENWAY, R., BOMMHARDT, U., ARNDT, B., HAUS, U.-U., WEISMANTEL, R., GILLES, E. D., KLAMT, S., AND SCHRAVEN, B. 2007. A logical model provides insights into T cell receptor signaling. *PLoS Computational Biology* 3, 8. 22, 24, 143

- [167] 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 Computational Biology* 5, 8. 22, 23, 73, 118, 122, 131, 143, 176
- [168] SARAIYA, P., LEE, P., AND NORTH, C. 2005. Visualization of graphs with associated timeseries data. In *Proceedings of the 2005 IEEE Symposium on Information Visualization*. IEEE Computer Society, 30–42. 115
- [169] SARAIYA, P., NORTH, C., AND DUCA, K. 2005a. An insight-based methodology for evaluating bioinformatics visualizations. *IEEE Transactions on Visualization and Computer Graphics* 11, 4, 443–456. 115
- [170] SARAIYA, P., NORTH, C., AND DUCA, K. 2005b. visualization of biological pathways: requirements analysis, systems evaluation, and research agenda. *Information Visualization* 4, 3, 1–15. 2, 48, 49, 58, 61, 143
- [171] SARKAR, M. AND BROWN, M. H. 1992. Graphical fisheye views of graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, 83–91. 43, 80
- [172] SCHAFER, D., ZUO, Z., BARTRAM, L., DILL, J., DUBS, S., GREENBERG, S., AND ROSEMAN, M. 1993. Comparing fisheye and full-zoom techniques for navigation of hierarchically clustered networks. In *Proceedings of Graphics Interface*. GI '93. 87–96. 80, 83, 92
- [173] SCHOEBERL, B., EICHLER-JONSSON, C., GILLES, E. D., AND MUELLER, G. 2002. Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors. *Nature Biotechnology* 20, 4, 370–375. 22
- [174] 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. 49, 100
- [175] SHNEIDERMAN, B. 1992. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transaction on Graphics* 11, 1, 92–99. 108
- [176] SHNEIDERMAN, B. 1996. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*. IEEE Computer Society Press, 336–343. 90
- [177] SHNEIDERMAN, B. 2001. Supporting creativity with advanced information-abundant user interfaces. Tech. rep., Department of Computer Science, University of Maryland, USA. 36
- [178] SILVA, C. T. AND FREIRE, J. 2008. Software infrastructure for exploratory visualization and data analysis: past, present, and future. *Journal of Physics: Conference Series* 125, 1. 141
- [179] SIX, J. AND TOLLIS, I. 1999. A framework for circular drawings of networks. In *Graph Drawing*, J. Kratochvíř, Ed. Lecture Notes in Computer Science, vol. 1731. Springer, 107–116. 33
- [180] SMITH, L. P., BERGMANN, F. T., CHANDRAN, D., AND SAURO, H. M. 2009. Antimony: a modular model definition language. *Bioinformatics* 25, 18, 2452–2454. 18
- [181] SOLIMAN, S. 2009. Modelling biochemical reaction networks with BIOCHAM extracting qualitative and quantitative information from the structure. In *Proceedings of the 6th International Conference on Mathematical Modelling*. MATHMOD '09, vol. 35. ARGESIM, 2304–2312. 15

Bibliography

- [182] SOROKIN, A., PALIY, K., SELKOV, A., DEMIN, O. V., DRONOV, S., GHAZAL, P., AND GORYANIN, I. 2006. The Pathway Editor: a tool for managing complex biological networks. *IBM Journal of Research and Development* 50, 6, 561–573. 49, 55
- [183] SPENCE, R. 2006. *Information visualization – design for interaction*. Pearson Education. 11
- [184] SPRINZAK, E., SATTATH, S., AND MARGALIT, H. 2003. How reliable are experimental protein-protein interaction data? *Journal of Molecular Biology* 327, 919–923. 10
- [185] STOREY, M. A., BEST, C., MICHAUD, J., RAYSIDE, D., LITOIU, M., AND MUSEN, M. 2002. SHriMP views: an interactive environment for information visualization and navigation. In *Extended abstracts on Human Factors in Computing Systems*. ACM Press, 520–521. 49
- [186] STOREY, M. A. D., FRACCHIA, F. D., AND MUELLER, H. A. 1999. Customizing a fisheye view algorithm to preserve the mental map. *Journal of Visual Languages and Computing* 10, 3, 245–267. 80
- [187] STOREY, M. A. D. AND MUELLER, H. A. 1995. Graph layout adjustment strategies. In *Proceedings of the 3rd International Symposium on Graph Drawing*. Springer, 487–499. 136
- [188] SUDERMAN, M. AND HALLETT, M. 2007. Tools for visually exploring biological networks. *Bioinformatics* 23, 20, 2651–2659. 2, 48, 143
- [189] SUGIYAMA, K. AND MISUE, K. 1995. Graph drawing by the magnetic spring model. *Journal of Visual Languages & Computing* 6, 3, 217–231. 34, 131
- [190] SUGIYAMA, K., TAGAWA, S., AND TODA, M. 1981. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics* 11, 2, 109–125. 33
- [191] TAMASSIA, R. 1987. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing* 16, 3, 421–444. 33, 34
- [192] TOLMAN, E. C. 1948. Cognitive maps in rats and men. *Psychological Review* 55, 4, 189–208. 39
- [193] TOM SAWYER. 2012. Tom Sawyer Software. <http://www.tomsawyer.com/>. Online; accessed February 15, 2012. 108
- [194] TOMITA, M., HASHIMOTO, K., TAKAHASHI, K., SHIMIZU, T. S., MATSUZAKI, Y., MIYOSHI, F., SAITO, K., TANIDA, S., YUGI, K., VENTER, J. C., AND HUTCHISON, C. A. 1999. E-CELL: software environment for whole-cell simulation. *Bioinformatics* 15, 1, 72–84. 18
- [195] TOYODA, T., MOCHIZUKI, Y., AND KONAGAYA, A. 2003. GScoPe: a clipped fisheye viewer effective for highly complicated biomolecular network graphs. *Bioinformatics* 3, 19, 437–438. 49, 110
- [196] TREISMAN, A. 1985. Preattentive processing in vision. *Computer Vision, Graphics, and Image Processing* 31, 2, 156–177. 38, 41
- [197] TUFTE, E. R. 1990. *Envisioning information*. Graphics Press. 99
- [198] TUFTE, E. R. 1997. *Visual explanations: images and quantities, evidence and narrative*. Graphics Press. 36

- [199] UETZ, P., IDEKER, T. G., AND SCHWIKOWSKI, B. 2002. Visualization and integration of protein-protein interactions. In *Protein-Protein Interactions – A Molecular Cloning Manual*, E. Golemis, Ed. CSHL Press, 623–646. 30
- [200] VILLÉGER, A. C., PETTIFER, S. R., AND KELL, D. B. 2010. Arcadia: a visualization tool for metabolic pathways. *Bioinformatics* 26, 11, 1470–1471. 109
- [201] VON LANDESBERGER, T., KUIJPER, A., SCHRECK, T., KOHLHAMMER, J., VAN WIJK, J., FEKETE, J. D., AND FELLNER, D. 2011. Visual analysis of large graphs: state-of-the-art and future research challenges. *Computer Graphics Forum* 30, 6, 1719–1749. 27, 28, 47
- [202] WALSHAW, C. 2003. A multilevel algorithm for force-directed graph drawing. *Journal of Graph Algorithms and Applications* 7, 3, 253–285. 34
- [203] WARE, C. 2000. *Information visualization: perception for design*. Morgan Kaufmann Publishers. 36, 41, 77
- [204] WIECHERT, W., NOACK, S., AND ELSHEIKH, A. 2010. Modeling languages for biochemical network simulation: reaction vs equation based approaches. In *Biosystems Engineering II*, C. Wittmann and R. Krull, Eds. Vol. 121. Springer, 109–138. 8, 13, 20, 125
- [205] WIESE, R., EIGLSPERGER, M., AND KAUFMANN, M. 2002. yFiles: visualization and automatic layout of graphs. In *Graph Drawing*, P. Mutzel, M. Juenger, and S. Leipert, Eds. Vol. 2265. Springer, 588–590. 34, 108
- [206] WIKIPEDIA. 2011. Signal transduction – Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/wiki/Signal_transduction. Online; accessed November 5, 2011. 9
- [207] WILLS, G. J. 1997. NicheWorks – interactive visualization of very large graphs. In *Proceedings of the 5th International Symposium on Graph Drawing*. GD '97. Springer, 403–414. 33
- [208] WIMALARATNE, S. M., HALSTEAD, M. D. B., LLOYD, C. M., COOLING, M. T., CRAMPIN, E. J., AND NIELSEN, P. F. 2009a. Facilitating modularity and reuse: guidelines for structuring CellML 1.1 models by isolating common biophysical concepts. *Experimental Physiology* 94, 5, 472–485. 7, 17
- [209] WIMALARATNE, S. M., HALSTEAD, M. D. B., LLOYD, C. M., COOLING, M. T., CRAMPIN, E. J., AND NIELSEN, P. F. 2009b. A method for visualizing CellML models. *Bioinformatics* 25, 22, 3012–3019. 16, 17, 131
- [210] YARDEN, Y. AND SLIWKOWSKI, M. X. 2001. Untangling the ErbB signalling network. *Nature Reviews Molecular Cell Biology* 2, 2, 127–137. 23
- [211] YWORKS. 2012. yEd graph editor. <http://www.yworks.com/products/yed/>. Online; accessed March 10, 2012. 34, 108

- CellNetAnalyzer, 19, 56, 115
- ContextualZoom, 79
- domain-specific, 35, 86, 110, 131
- FlatLayout, 135
- focus path, 48, 90, 140
- Focus+Context, 43, 79
- gene regulation, 8
- global VS
 - Connectivity, 109
 - Dynamic Model, 107
 - Hierarchical Level, 107
 - Interaction Distance, 109
 - Logical Model, 107
- graph
 - compound, 28
 - hierarchical, 28
 - nested, 29
 - simple, 28
 - terminal cluster, 29, 56, 129
- hierarchical level, 29, 47, 57, 78, 93, 128, 137
- HierarchicalZoom, 82
- HyCoGLa, 128
- interactive exploration scheme, 78, 90, 127
- InteractiveZoom, 79
- level of detail, 43, 85
- MAPK cascade, 22–24
- MDL, 55, 60, 61
- mental map, 39
- metabolism, 7
- model
 - dynamic EGF, 22
 - logical EGFR/ErbB, 23
 - logical TCR, 24
- model type
 - biological, 13
 - computational, 12
 - graphical, 13
 - mathematical, 12
 - mental, 13
 - modular, 13
- modeling
 - dynamic, 14
 - logical, 14
 - modular, 15
- modular modeling concept, 17
- module
 - composite, 21, 59, 66
 - elementary, 21
- multiple maps, 68
- network, 45
 - cellular, 13, 19, 46
 - gene regulatory, 45
 - metabolic, 45
 - protein-protein-interaction, 45
 - signaling, 45
- Overview+Detail, 42, 82
- Pan+Zoom, 43, 50, 79, 82
- ProMoT, 19
 - Class Browser, 22
 - Visual Editor, 22

Index

Visual Explorer, 5

SBGN, 46, 49, 56, 99

SBML, 1, 18, 50, 125

scene graph, 49

shortest path, 29, 113

signal transduction, 7, 110

space scale diagram, 43

spring-embedder, 34

UML, 164

view

global, 47, 89

integrated, 47

level, 47

local, 47, 69, 82, 96

multiple, 47

standard, 47, 55

Visual Scenario

Logical Model, 107

visual scenario

global, 105

local, 106

zoomable user interface, 43, 78

zooming, 43

CD Information, Software Architecture and UML Diagrams

This appendix gives some information on the supplied CD and its content. Thereafter, some information on the architecture of PROMoT and the classes of VISUAL EXPLORER is given.

A.1. Supplied CD

Many examples illustrating animations and interactions used within VISUAL EXPLORER are included as movie files on the CD provided with this thesis. These files are stored in popular codecs and containers such as MPEG-4, AVI, FLV and MOV. The particular file name indicates the appropriate figure in the thesis.

The movies can be watched by clicking on the particular file using a multimedia player such as VIDEO-LAN CLIENT (VLC)TM. VLC is an open-source and cross-platform multimedia player and framework that plays most video formats. It can be downloaded free of charge from <http://www.videolan.org/>.

A.2. Architecture of PROMoT

In Figure A.1, the high-level software architecture of PROMoT is depicted. With respect to the modeling formalism, PROMoT can read three different formats: MDL, SBML and CELLNETANALYZER. The input model will be interpreted and stored in the PROMoT kernel. For analysis, a model instance is created and processed by one of the different writers (dynamic or logical) located in the output layer. In order to analyze or simulate models, PROMoT can export to several simulation backends and analysis tools, which results in an optimized modeling workflow (MDL, MATLAB, DIVA/DIANA, SBML, CELLNETANALYZER). VISUAL EXPLORER supports import and export of graph formats (e. g., GRAPHML) in order to save the layout. Further, the user can also export the graphics into various bitmap and vector formats. For instance, the modeler can use vector graphics, for example, PDF, EMF or SVG, in order to maintain the details and allow zooming in. The exported model or graphics can be processed further in order to get an adequate visual representation of the model.

A.3. UML Diagrams

In Figure A.2, the UML diagram of VISUAL EXPLORER is depicted. It presents an overview picture of the implemented classes, related PROMoT classes and third-party classes. Another UML diagram, given in Figure A.3, shows the package structure of VISUAL EXPLORER.

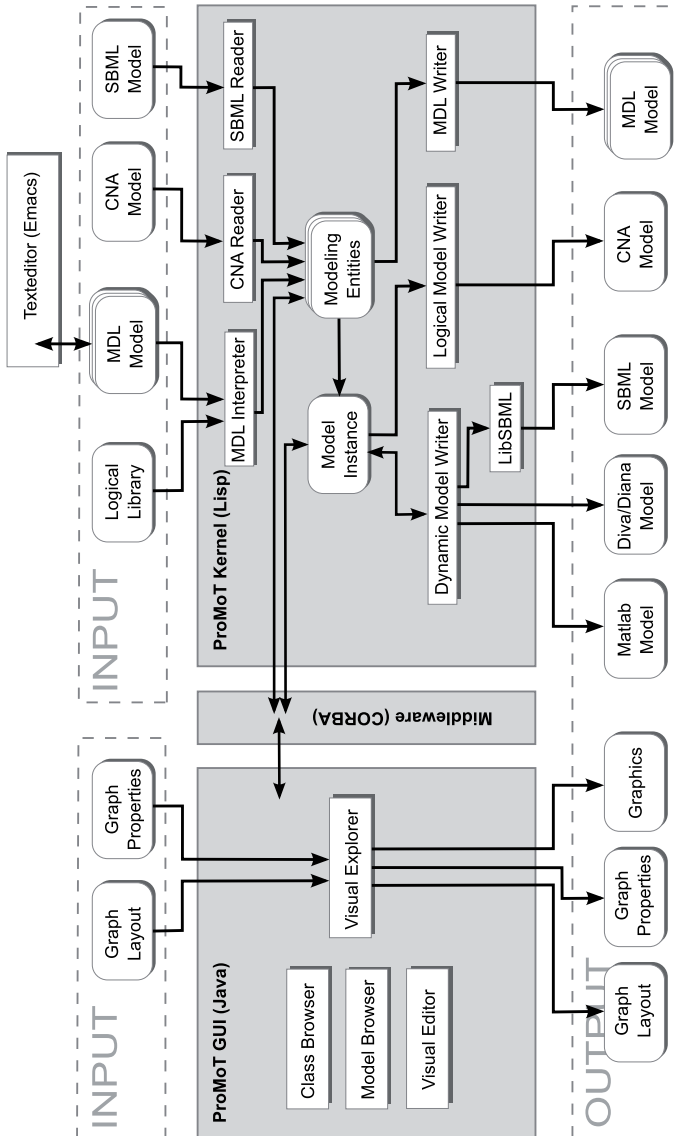


Figure A.1.: Diagram of the high-level software architecture of PROMoT and the corresponding input and output layer. **VISUAL EDITOR** and **VISUAL EXPLORER** are part of the PROMoT GUI written in the programming language **JAVA**.

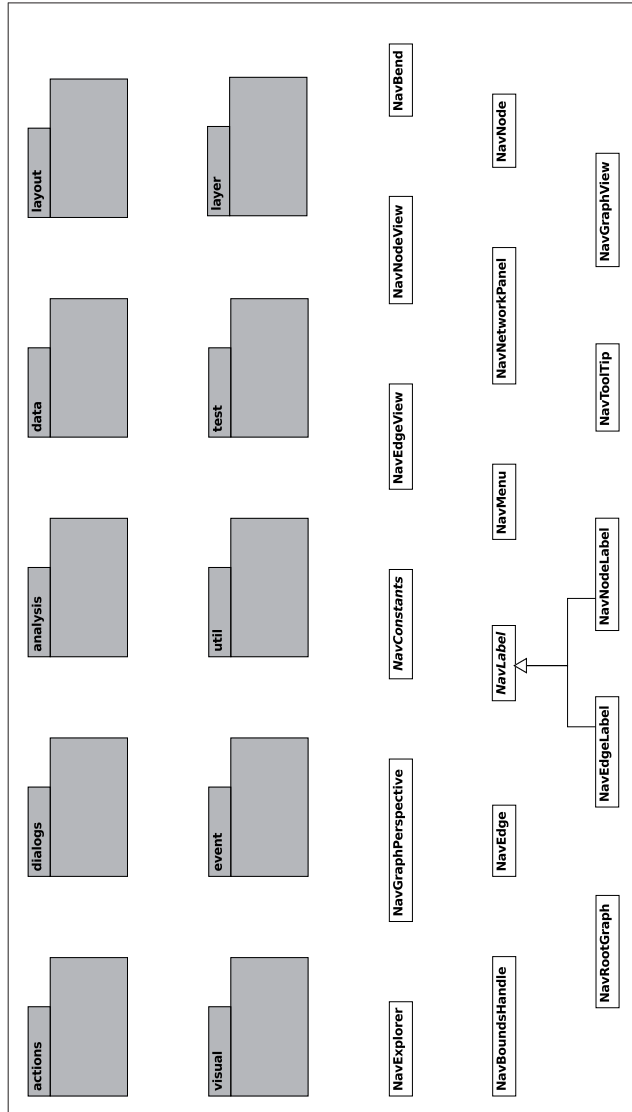


Figure A.3.: UML diagram compiling the package structure and high-level classes of VISUAL EXPLORER. ‘ $\text{--}\triangleright$ ’ encodes a generalization between a subclass and a superclass.

Visual Representations and Visual Scenarios

In this appendix symbol reference cards for dynamic and logical models are presented. Furthermore, some comparisons on different interactive exploration schemes are given. Finally, an overview of the shortcuts and controls to navigate and explore modular models defined in VISUAL EXPLORER is listed.

B.1. Symbol Reference Cards

In Figure B.1, the notation symbols used for the visual representation of logical models are depicted. They are provided in the form of a reference card. Analogous to that, in Figure B.2, notation symbols used for dynamic models are shown. These reference cards provide also an overview of the visual mappings defined in the global VS ‘Dynamic Model’ and in the global VS ‘Logical Model’.

The set of symbols is tailored to the modeling entities in the modeling libraries for dynamic and logical models. They are visually modified either by altering their visual properties (e. g., border color) or by using a bitmap icon.

B.2. Comparison of Elements

In Table B.1 a comparison between the visual representations of elements for logical modeling in VISUAL EDITOR and VISUAL EXPLORER is given. All elements are standard modeling entities taken from the logical library.

B.3. Visual Handling of Additional Parameters

In the following the visual handling of additional parameters used to encode properties of the logical model is given. Important parameters and their visual handling are already discussed in Chapter 4, Section 4.4.3.

Time Scale

For modeling different time-dependent reactions, the `time scale` can be used. It is coupled to reactions in the model. For instance, to reflect a reaction that is active earlier in the signaling process, the

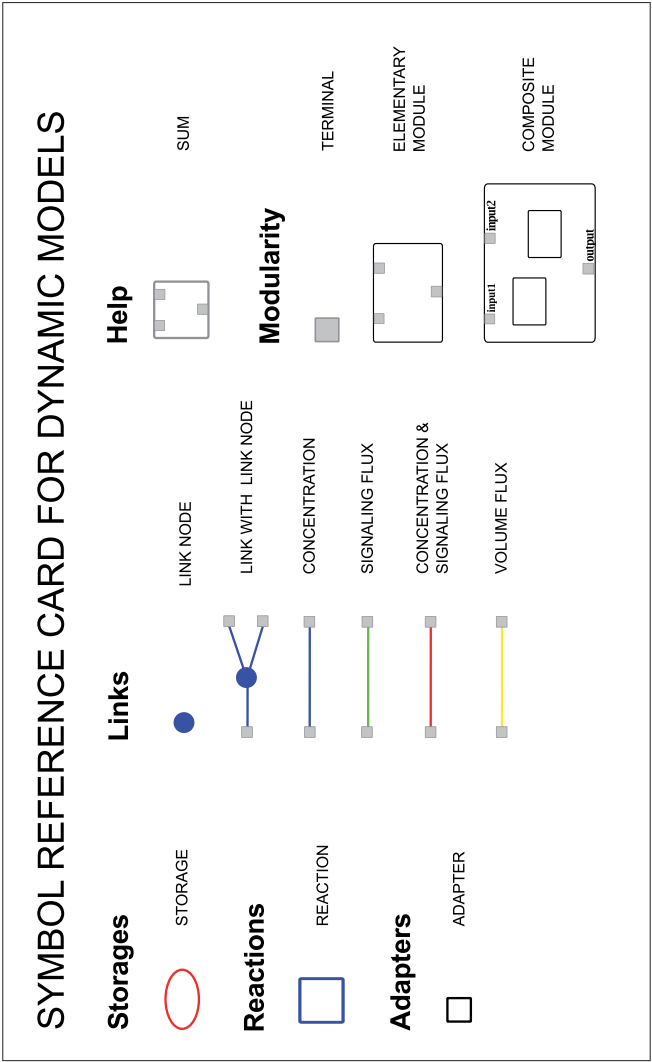


Figure B.1.: Reference card showing the basic set of symbols used for the visual representation of dynamic models.

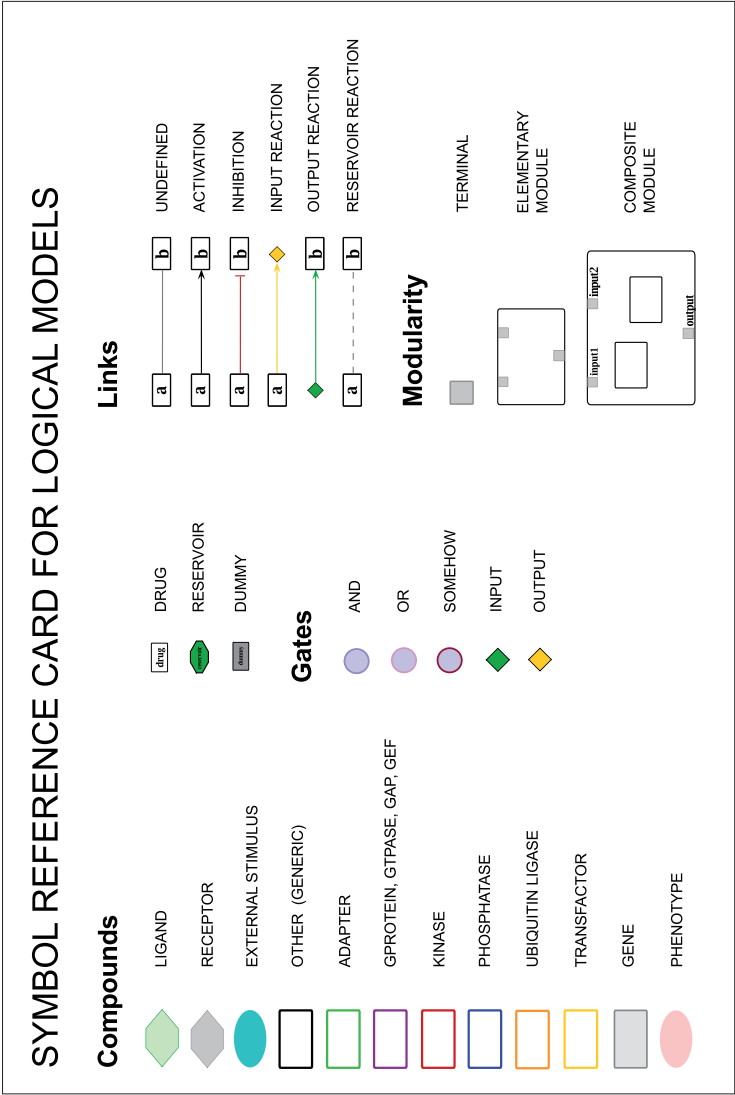


Figure B.2.: Reference card showing the basic set of symbols used for the visual representation of logical models.

B. Visual Representations and Visual Scenarios

<i>Logical Element</i>	<i>Visual Editor</i>	<i>Visual Explorer</i>
Compounds		
dummy	explicit (using a dummy element)	implicit (encoded by a bend point), explicit
reservoir	explicit (using a reservoir element)	implicit, explicit
other	explicit (using a other element)	explicit
Gates		
and	explicit (using an and element)	explicit (using an and element)
or	explicit (using an or element, implicit)	implicit
somehow	explicit (using a somehow element)	explicit (using a somehow element)
Miscellaneous		
not	explicit (using a not element)	implicit (encoded by a red edge with a T-end)
activ	explicit (using a activ element)	implicit (encoded by a black edge with arrow)

Table B.1.: Direct comparison of visual representations of relevant elements in `VISUAL EDITOR` and `VISUAL EXPLORER`. All elements are standard modeling entities taken from the logical library.

parameter `time scale` can be set to a lower value (e. g., value ‘1’). Whereas when a reaction takes place later in the process, the parameter `time scale` should be set to a larger value (e. g., value ‘2’). This property can be used to easily visualize the time behavior of reactions in context of the entire logical network. It can also be used as an important visual aid during analysis in `CELLNETANALYZER`.

Ignore in Export

The parameter `ignore in export` determines whether or not the related element is exported to `CELLNETANALYZER`. This parameter is not defined in species but rather in gates. If the parameter has the value ‘0’, it will not be considered in the export; whereas with the value ‘1’ it will be exported. The state of the gate can be visually reflected in the logical VS. For that, all gates with a value of ‘0’ can be automatically removed from the visual representation and are not present in `CELLNETANALYZER`. This reduces the visual clutter.

Logical Excluded

In contrast to the former parameter, elements with parameter `logical excluded` will be considered in export. This parameter is relevant for further proceeding in `CELLNETANALYZER` where elements having a parameter with value ‘1’ are excluded from the logical computation, whereas elements with parameter value ‘0’ are considered. Often, it is beneficial to know this information before export or at least during analysis in `CELLNETANALYZER`. For that, elements with a parameter value of ‘1’ can be automatically removed from the logical VS. This is important in order to get a visual overview of the states of this parameter in the network context.

Confidence Level

The parameter `confidence_level` is used to model uncertainties regarding the logical combinations in the model. It defines the level of confidence of the encoded biological knowledge. Modules with a value of '1.0' have a high confidence; that is, they are well-known and established. In contrast, modules having a small value (e.g., value '0.05') have a low confidence; that is, the knowledge about the element is very uncertain. In *VISUAL EXPLORER* the confidence level can be visualized using the transparency property – low confidence is depicted using high transparency; no transparency is used when high confidence is given. In the later analysis with *CELLNETANALYZER*, this can be an important visual aid.

B.4. Visual Representation of the Logical Toy Model

In this section, the transformation steps described in Chapter 4, Section 4.2 and encapsulated in the VS 'Logical Model' are illustrated using the *Toy* model. As already mentioned, this process aims at designing a biologically motivated representation.

The *Toy* model is set up in *PROMOT* in a modular fashion (Chapter 4, Figure 4.12a). Thereby, the local views are separated in different windows (composite modules: `whole_model`, `cell` and `nucleus`). Then, the *Toy* model is opened and automatically converted in *VISUAL EXPLORER* using the VS 'Logical Model' (Chapter 4, Figure 4.12b). The different separate views are integrated in a single nested view.

In contrast to Figure 4.12a (Chapter 4), in Figure 4.12b (Chapter 4) all `not` elements are removed. Instead, the information of a negative influence is encoded by a red colored edge of the particular reaction. A similar rule is applied to the positive effects where `activ` elements are removed from the network and encoded in black color of the related edge. Additionally, the direction of signal flow is indicated by target arrows that are different for activation and inhibition. Furthermore, the reservoir `k3r` has been automatically removed, since it belongs to the class `reservoir` and is not necessary for a clear biological interpretation. Instead, an edge connecting the corresponding compounds supplied by the entity pools is drawn using a dashed style.

For a more biological motivated representation the composite modules, in particular `cell` and `nucleus` are drawn as ellipses rather as strict rectangles. Additionally, the ellipses are rendered in yellow and dashed (for imitating a biological, semi-permeable membrane). Another area for 'optimization' are network edges. They are rendered as curved edges (i.e., Bézier curves) instead of strictly angular edges. Both representations, ellipses and curved edges, are more familiar to biologists.

Figure B.3 gives a side-by-side comparison of the visual representation of a simplified *Toy* model in *VISUAL EDITOR* and *VISUAL EXPLORER*.

As described in Chapter 6, Section 6.2.5, from the global VS 'Logical Model', different local VSs can be derived for matching certain modeling tasks or individual preferences. In Figure B.4, alternative visualizations using different local VSs are shown. In contrast to Figure 4.12b (Chapter 4), in Figure B.4a the visual focus is on the reservoir `k3r`. Furthermore, all logical gates are unified, for instance, there is no distinction between `and` and `somehow` gates. The network representation in Figure B.4b highlights the wiring of the network, not the species themselves. Species are depicted only by label name. The type of a species is encoded in the color of the label text, for example, yellow color for output elements.

B. Visual Representations and Visual Scenarios

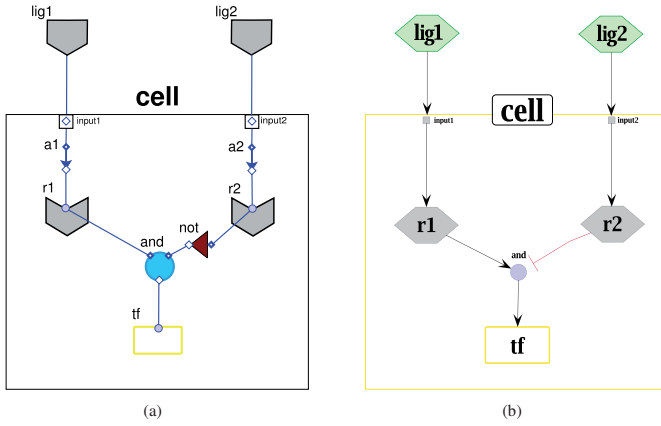


Figure B.3.: Comparison of the visual representation of a simplified Toy model, (a) in VISUAL EDITOR and (b) in VISUAL EXPLORER.

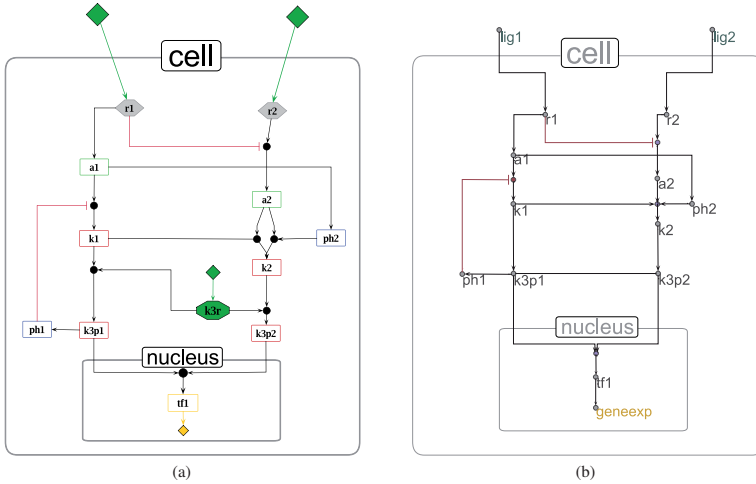


Figure B.4.: Alternative visualizations of the Toy model using different local VSs. (a) Representation focusing on reservoir $k3r$; logical gates are unified. (b) Representation emphasizing the wiring of the network; entities are depicted by label name. The Toy model using the global VS for logical models is depicted in Chapter 4, Figure 4.12b. Figure on the right is adapted from Saez-Rodriguez et al. [165].

B.5. Logical Model of EGFR/ErbB Signaling

An overview of visual representation in different steps in the logical modeling workflow of the EGFR/ErbB model is given in Figure B.5. In particular, these steps are setup and editing in VISUAL EDITOR, exploration and visualization in VISUAL EXPLORER, structural analysis in the CELLNETANALYZER and mapping of analysis results to the network in VISUAL EXPLORER.

B.6. Logical Model of TCR Signaling

Different VSs of the logical TCR model used for visual analysis are depicted in Figure B.6 and Figure B.7.

B.7. Visual Properties and Mappings

Visual properties of nodes and edges, their related data types, and their associated mappings in VISUAL EXPLORER are given in Table B.2.

B.8. Visual Scenarios and Model Types

In Table B.3 the relations between the different components and the global VS is given. Furthermore, in Table B.4 all global VSs are listed according their valid model types. Another table (Table B.5) compiles the network elements that are not considered in the computation of the VS ‘Connectivity’ and the VS ‘Interaction Distance’ according to the model type.

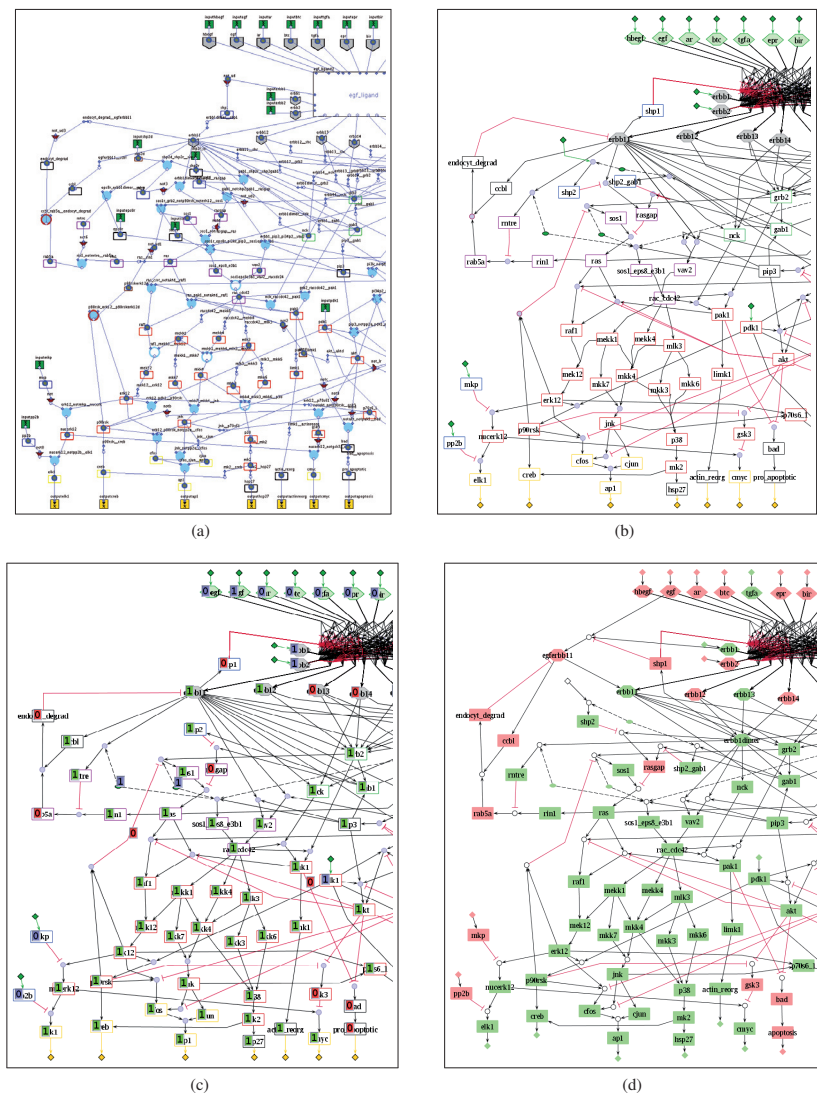
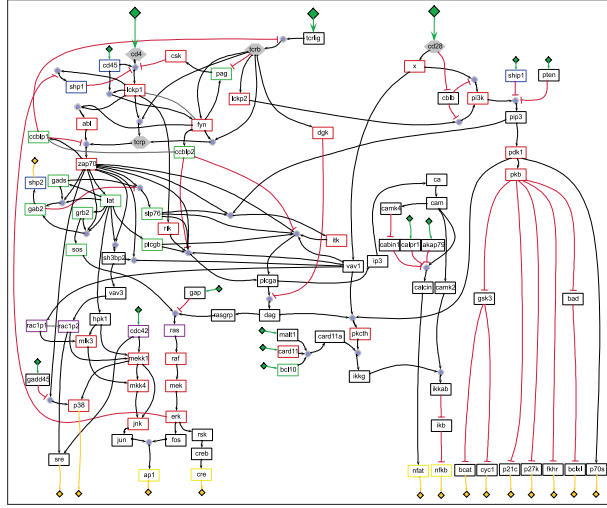
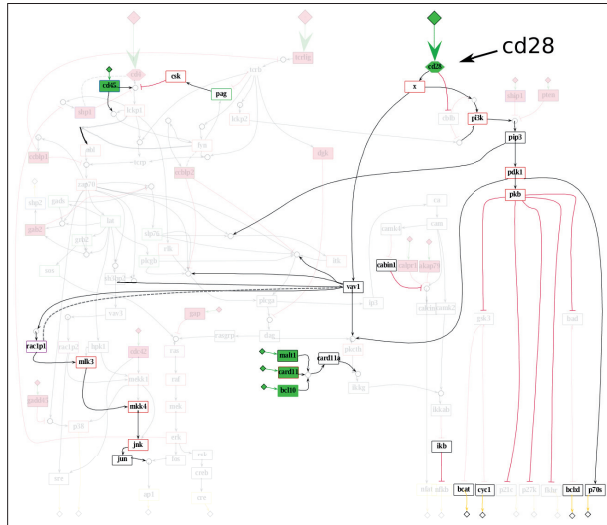


Figure B.5.: Compilation of visual representation in different steps in the logical modeling workflow of the logical EGFR/Erbb model. Only a part of the network is shown. (a) Setup and editing in VISUAL EDITOR; (b) exploration and visualization in VISUAL EXPLORER; (c) structural analysis in CELLNETANALYZER; (d) analysis results mapped back to the network in VISUAL EXPLORER. Figures courtesy of Regina Samaga, some of which were published in [167].



(a)



(b)

Figure B.6.: Different VSs of the logical TCR model used for visual analysis. (a) Using a global VS specifically adapted to logical models. The global VS is instantiated to different local VSs and then combined with analysis data from CELLNETANALYZER. (b) Paths activated upon stimulation of molecule `cd28`. Species in green and red represent states fixed to 1 and 0, respectively. Elements depicted in semi-transparent are not activated upon stimulation molecule `cd28`. Figures courtesy of Julio Saez-Rodriguez, published in [161].

<i>Property Type</i>	<i>Data Type</i>			<i>Transfer Function</i>		
	Categorical	Ordered	Continuous	Direct	Discrete	Continuous
Node Properties						
Node Border Color	.	✓	✓	.	✓	✓
Node Border Style	✓	.	.	.	✓	.
Node Border Width	.	✓	✓	.	✓	✓
Node Fill Color	.	✓	✓	.	✓	✓
Node Label Color	.	✓	✓	.	✓	✓
Node Label Font Face	✓	.	.	.	✓	.
Node Label Font Size	.	✓	✓	.	✓	✓
Node Label Text	✓	.	.	✓	✓	.
Node Label Position	✓	.	.	.	✓	.
Node Scale	.	.	✓	.	✓	✓
Node Shape	✓	.	.	.	✓	.
Node Transparency	.	✓	✓	.	✓	✓
Edge Properties						
Edge Label Color	.	✓	✓	.	✓	✓
Edge Label Font Face	✓	.	.	.	✓	.
Edge Label Font Size	.	✓	✓	.	✓	✓
Edge Label Text	✓	.	.	✓	✓	.
Edge Line Color	.	✓	✓	.	✓	✓
Edge Line Style	✓	.	.	.	✓	.
Edge Line Type	✓	.	.	.	✓	.
Edge Line Width	.	✓	✓	.	✓	✓
Edge Line Transparency	.	✓	✓	.	✓	✓
Edge Source Arrow	✓	.	.	.	✓	.
Edge Target Arrow	✓	.	.	.	✓	.
Global Properties						
Global Background Color	.	✓	✓	.	✓	✓
Global Highlight Node Color	.	✓	✓	.	✓	✓
Global Highlight Edge Color	✓	✓	✓	.	✓	✓

Table B.2.: Visual mappings for nodes and edges in VISUAL EXPLORER. ‘✓’ – mapping is supported for the specified visual property, ‘.’ – mapping is not supported for the specified visual property, ‘Data Type’ – which data type can be mapped to the visual property, ‘Transfer Function’ – which transfer function can be used to map the data type to the visual property.

B. Visual Representations and Visual Scenarios

<i>Global VS / Component</i>	<i>Filter</i>	<i>Analysis</i>	<i>Visual Mapping</i>	<i>Layout</i>	<i>Exploration Scheme</i>
Dynamic Model	\emptyset	\emptyset	✓	\emptyset	HierarchicalZoom
Logical Model	✓	✓	✓	✓	HierarchicalZoom
SBML Model	\emptyset	\emptyset	✓	\emptyset	HierarchicalZoom
Unknown Model	\emptyset	\emptyset	✓	\emptyset	InteractiveZoom
Connectivity	\emptyset	✓	✓	\emptyset	HierarchicalZoom
Hierarchical Level	\emptyset	✓	✓	\emptyset	HierarchicalZoom
Interaction Distance	\emptyset	✓	✓	\emptyset	\emptyset

Table B.3.: Each global VS has default components. ‘✓’ – component is set; ‘ \emptyset ’ – component is an empty set in VS.

<i>Global VS / Model Type</i>	<i>Dynamic Model</i>	<i>Logical Model</i>	<i>SBML Model</i>	<i>Unknown Model</i>
Dynamic Model	*	.	✓	.
Logical Model	.	*	.	.
SBML Model	✓	.	*	.
Unknown Model	✓	✓	✓	*
Connectivity	✓	✓	✓	✓
Hierarchical Level	✓	✓	✓	✓
Interaction Distance	✓	✓	✓	✓

Table B.4.: Each model type has a default global VS and a list of additional valid global VSs in VISUAL EXPLORER. ‘*’ – VS is the default option for model type; ‘✓’ – VS is supported for model type; ‘.’ – VS is not valid for the given model type.

<i>Model Type</i>	<i>Invalid Network Elements</i>
Dynamic Model	reaction, help, adapter, terminal, link node
Logical Model	dummy, and, or, somehow, terminal
SBML Model	reaction, help, adapter, terminal, link node
Unknown Model	terminal

Table B.5.: Model types and related network elements that are not considered in the computation of the VS ‘Connectivity’ and the VS ‘Interaction Distance’. Network elements are given by their class names (if suitable).

Interactive Exploration Schemes

This appendix gives some comparisons on different interactive exploration schemes, and an overview of the shortcuts and controls defined in VISUAL EXPLORER.

C.1. InteractiveZoom versus ContextualZoom

In Figure C.1, a comparison of the InteractiveZoom and the ContextualZoom using the example of the logical TCR model is given.

C.2. Zoom Sequence with LOD

In Figure C.2, a zoom sequence is performed within the model of central metabolism of *E. coli*. The model comprises 1430 nodes, 652 edges and consists of three modules on the top level and eight hierarchical levels. The zooming is performed using the HierarchicalZoom.

C.3. Shortcuts and Mouse Controls

The Table C.1 lists common shortcuts and mouse controls introduced with VISUAL EXPLORER. They are necessary to view, navigate and explore the compound graph.

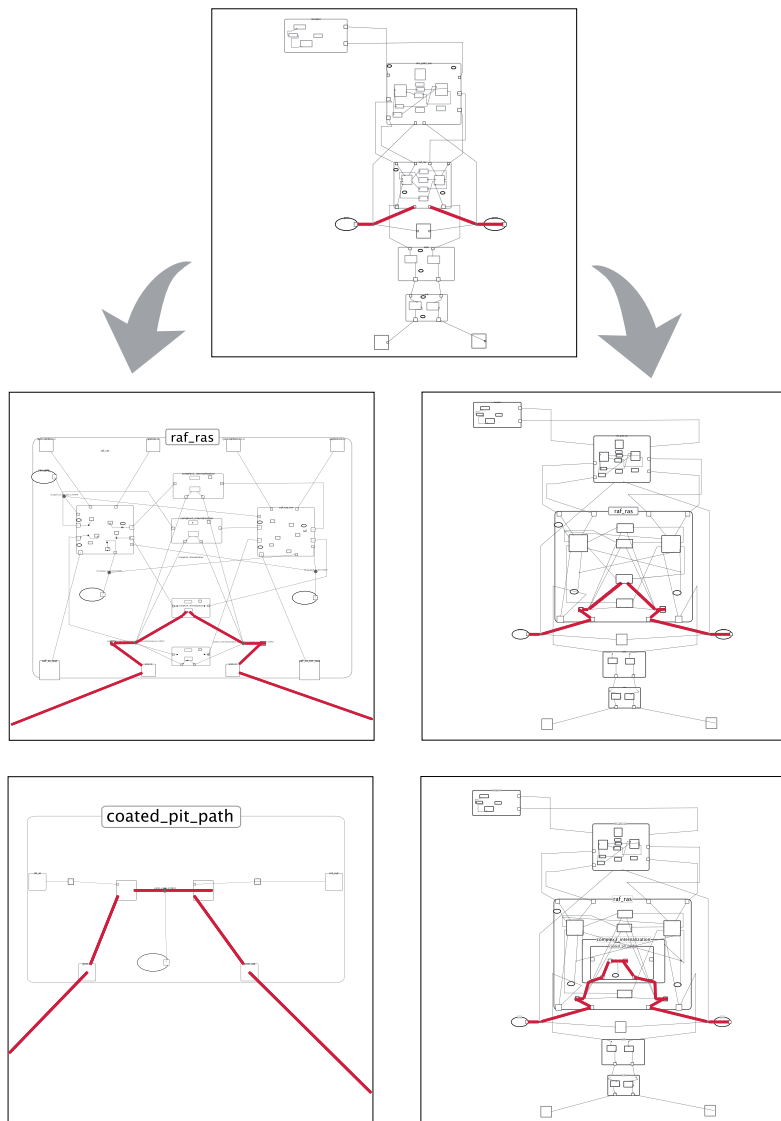


Figure C.1.: Highlight a path between two high-level elements using the InteractiveZoom (left) and the ContextualZoom (right). The path is emphasized using red. The zooming sequence using InteractiveZoom reveals more detail, but the context is lost. In contrast, the sequence using ContextualZoom is characterized by the whole model context, but with less detail. The dynamic EGF model is applied.

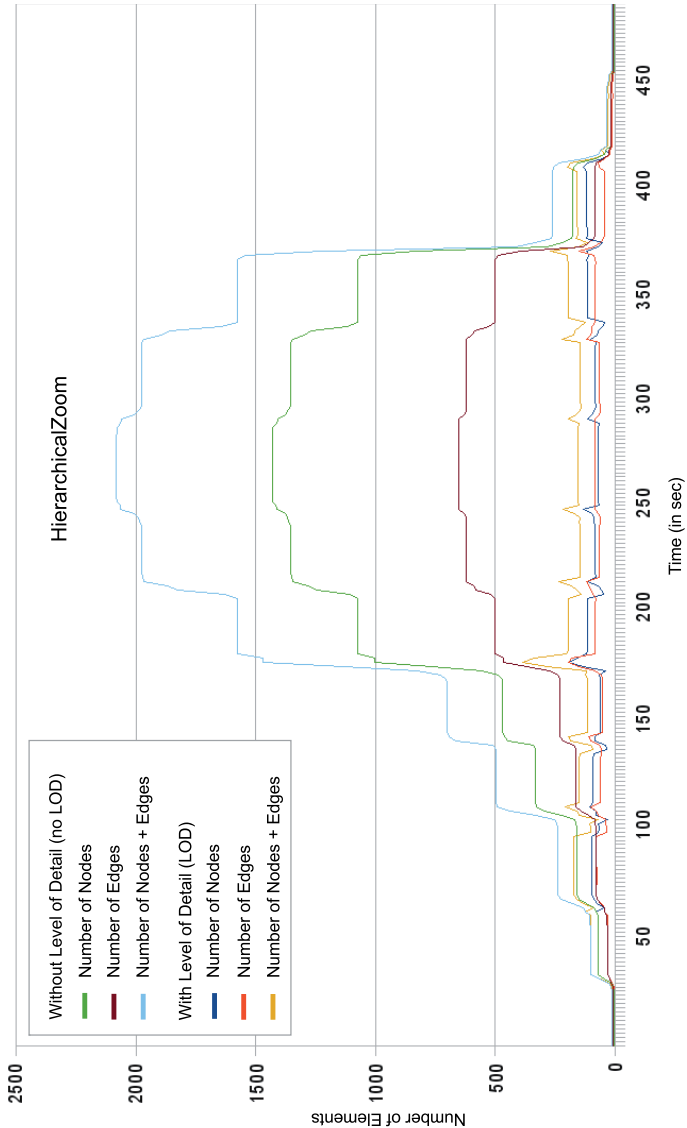


Figure C.2.: A typical zoom sequence with module-based navigation (HierarchicalZoom) using the `Ecoli` model. The zoom steps are performed either with or without LOD. As depicted with enabled LOD, the number of nodes, edges and overall elements are at a low constant level (around 130) while zooming within the hierarchical structure. In contrast without LOD, the number of elements increases up to 2300.

C. Interactive Exploration Schemes




























Operation	Shortcut	Description
Menus		
Open node menu	 on entity	Opens the context menu for nodes
Open edge menu	 on entity	Opens the context menu for edges
InteractiveZoom		
Zoom in	 + 	Zoom in the network
Zoom out	 + 	Zoom out the network
Pan	 + [Ctrl] + Drag	Move (pan) the entire canvas in any direction
HierarchicalZoom		
Zoom in	 on entity	Zoom in to focused entity in hierarchical steps
Zoom out	 on entity	Zoom out to focused entity in hierarchical steps
Fit network on screen	 + [Shift]	Zoom out the entire network to fit on screen
Pan	 + [Ctrl] + Drag	Move (pan) the entire canvas in any direction
ContextualZoom		
Zoom in	 on entity	Enlarge focused entity, shrink all other entities
Zoom out	 on entity + [Shift]	Shrink focused entity, enlarge all other entities
Pan	 + [Ctrl] + Drag	Move (pan) the entire canvas in any direction
Detail+Overview		
Pan	 on gray box + Drag	Move (pan) the entire canvas in any direction
Node Editing		
Move	 on entity + Drag	Move entity inside bounds of parent
Resize (all)	 on handle + Drag	Scale entity including children and label
Resize (bounds)	 on handle + [Shift] + Drag	Resize only bounds of entity, keep size of children and label
Pan	 + [Ctrl] + Drag	Move (pan) the entire canvas in any direction
Edge Editing		
Move handle	 on handle + Drag	Move edge handle
Add handle	 on edge for context menu	Add edge handle
Delete handle	 on edge for context menu	Remove edge handle
Delete all handles	 on edge for context menu	Remove all edge handles
Pan	 + [Ctrl] + Drag	Move (pan) the entire canvas in any direction
Global Visual Scenario ‘Interaction Distance’		
Show Int Dist	 on entity	Show Interaction Distance for focused entity
Show Int Dist (multi)	 on entity + [Shift]	Show Interaction Distance for previous and focused entity
Show Int Dist (path)	 on two entities + [Ctrl]	Show Interaction Distance for all entities on the shortest path

Table C.1.: Shortcuts and modifiers used in VISUAL EXPLORER.

This appendix, first, provides an example for the `FlatLayout` and gives an overview of the proposed algorithm. In a further section, the algorithmic realization of `HyCoGLa` is shown in detail.

D.1. Applying `FlatLayout`

In Figure D.1, a comparison of a manual, modular layout and a flat layout using the `FlatLayout` method is given. While in the manual layout all elementary nodes are scaled according to their hierarchical level, in the flat layout the elementary modules have the same size. As depicted, layout properties such as orthogonal ordering and proximity are preserved.

D.2. Algorithms for `FlatLayout`

The algorithm of the `FlatLayout` is given in Appendix D, Algorithms 1–2.

D.3. Algorithms for `HyCoGLa`

The algorithm of the `HyCoGLa` is given in Appendix D, Algorithms 3–8. They are adapted from Aschenbrenner [8].

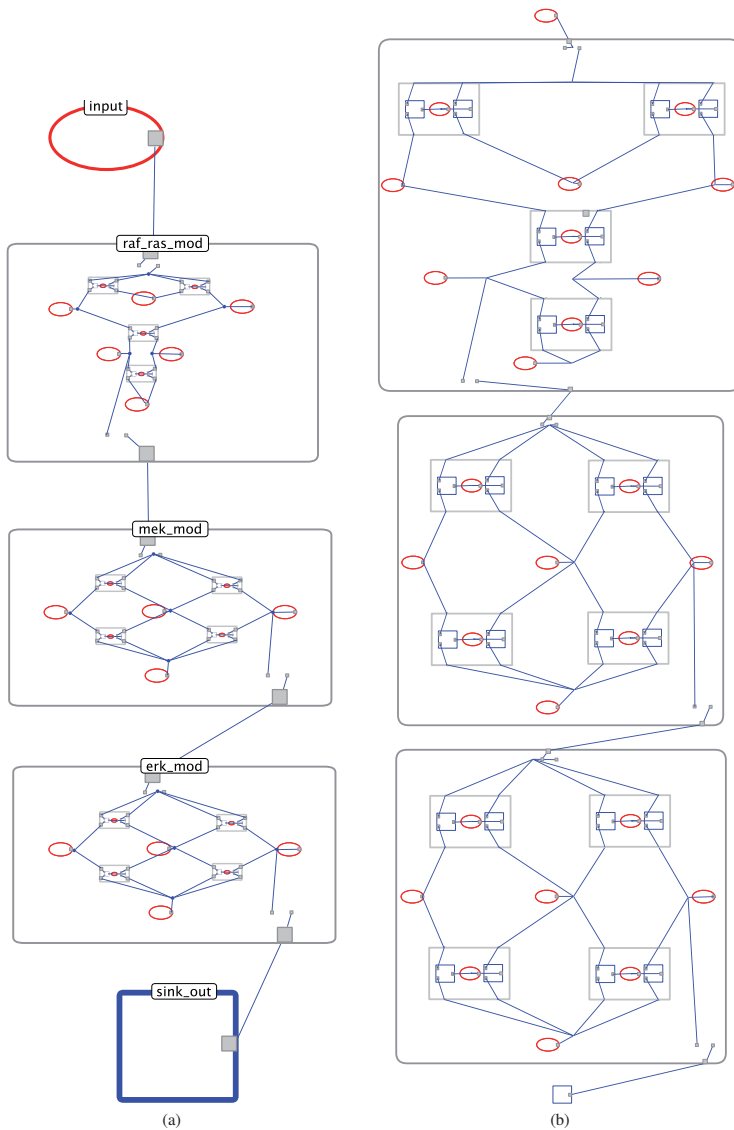


Figure D.1.: Comparison of a manual, modular layout and the FlatLayout method. The MAPK cascade, a part of the dynamic EGF model is shown. (a) Manual, modular layout, (b) FlatLayout where layout properties such as orthogonal ordering and proximity are preserved.

Algorithm 1 General FlatLayout algorithm

Input: Modular Model (graph containing subgraphs)**Output:** A FlatLayout for the modular model

```

1: time  $\leftarrow$  setAnimationTime()
2: emList  $\leftarrow$  getAllElementaryModules()
3: for all em  $\in$  emList do
4:   pm  $\leftarrow$  em.getParentModule()
5:   level  $\leftarrow$  em.getHierarchicalLevel()
6:   scale  $\leftarrow$  em.getScale() / em.getGlobalScale();
7:   transformElementaryModule(scale, time)
8:   smList  $\leftarrow$  em.getAllSiblingModules()
9:   for all sm  $\in$  smList do
10:    translationVector  $\leftarrow$  sm.computeTranslationVector()
11:    translateSiblingModule(translationVector, time)
12:   end for
13:   box  $\leftarrow$  computeBoundingBox(em, smList)
14:   transformParentModule(box, time)
15:   traverseHierarchy()
16: end for

```

Algorithm 2 Traverse hierarchy (traverseHierarchy)

```

1: if level > 0 then
2:   em  $\leftarrow$  pm
3:   pm  $\leftarrow$  pm.getParentModule()
4:   level  $\leftarrow$  em.getHierarchicalLevel()
5:   smList  $\leftarrow$  em.getAllSiblingModules()
6:   for all sm  $\in$  smList do
7:    translationVector  $\leftarrow$  sm.computeTranslationVector()
8:    translateSiblingModule(translationVector, time)
9:   end for
10:  box  $\leftarrow$  computeBoundingBox(em, smList)
11:  transformParentModule(box, time)
12:  traverseHierarchy()
13: end if

```

Algorithm 3 General HyCoGLa algorithm

Input: Inclusion tree of the modular model (graph containing subgraphs)**Output:** A layout for the modular model (graph containing subgraphs)

```

1: depth  $\leftarrow$  inclusionTree.depth
2: while depth > 1 do
3:   submodulesList  $\leftarrow$  getSubmodulesOfLevel(depth)
4:   for all submodule  $\in$  submodulesList do
5:     extendedSpringLayout(submodule)
6:   end for
7:   depth  $\leftarrow$  depth - 1
8: end while

```

Algorithm 4 Extended spring layout algorithm (extendedSpringLayout)

Input: Module (graph containing subgraphs)**Output:** A layout for the module

```

1:  $phase \leftarrow 1$ 
2: while  $phase \leq 2$  do
3:   if  $phase < 2$  then
4:      $details \leftarrow \text{calculateLeafNodes}(module)$ 
5:      $currentModule \leftarrow \text{calculateSkeleton}(module)$ 
6:   else
7:      $\text{placeLeafNodes}(details)$ 
8:      $currentModule \leftarrow module$ 
9:   end if
10:   $\text{initialize}(currentModule)$ 
11:   $i \leftarrow 0$ 
12:  while  $i \leq \text{maxIterations}$  or  $\text{stoppingConditions}() = \text{false}$  do
13:     $\text{calculateSpringForces}()$ 
14:     $\text{calculateRepulsiveForces}()$ 
15:     $\text{calculateGravitationalForces}()$ 
16:     $\text{calculateMagneticForces}()$ 
17:     $\text{updateNodes}()$ 
18:     $i \leftarrow i + 1$ 
19:  end while
20:   $phase \leftarrow phase + 1$ 
21: end while
22:  $\text{adjustTerminals}()$ 
23:  $\text{removeNodeOverlapping}()$ 

```

Algorithm 5 Calculate spring forces (calculateSpringForces)

Input: Module (graph containing subgraphs)**Output:** Spring forces F_S for all nodes

```

1: for all  $e = (u, v) \in E$  do
2:    $d \leftarrow \text{distance}(u.\text{terminal}, v.\text{terminal})$ 
3:   if  $u$  is elementary module and  $v$  is elementary module then
4:      $l \leftarrow \text{defaultSpringLength} * \text{scalingFactor}$ 
5:   else
6:      $l \leftarrow \text{defaultSpringLength}$ 
7:   end if
8:    $\text{spring}.x \leftarrow k * \log(d/l) * (u.\text{terminal}.x - v.\text{terminal}.x)/d$ 
9:    $\text{spring}.y \leftarrow k * \log(d/l) * (u.\text{terminal}.y - v.\text{terminal}.y)/d$ 
10:   $F_S(u) \leftarrow \text{spring} * u.\text{scalingFactor}$ 
11:   $F_S(v) \leftarrow -\text{spring} * v.\text{scalingFactor}$ 
12: end for

```

Algorithm 6 Calculate repulsive forces (calculateRepulsiveForces)**Input:** Module (graph containing subgraphs)**Output:** Repulsive forces F_R for all nodes

```

1:  $isFinished \leftarrow \{\}$ 
2: for all  $u \in V$  do
3:    $isFinished$  add  $u$ 
4:   for all  $v \in V - isFinished$  do
5:     if  $u.parent == v.parent$  then
6:        $d \leftarrow \text{distance}(u.center, v.center)$ 
7:       if  $d - u.radius - v.radius < repulsionRange$  then
8:          $l \leftarrow defaultSpringLength * u.scalingFactor + u.radius + v.radius$ 
9:          $repulsion.x \leftarrow e * (l/d) * (u.center.x - v.center.x)/d$ 
10:         $repulsion.y \leftarrow e * (l/d) * (u.center.y - v.center.y)/d$ 
11:         $F_R(u) \leftarrow repulsion * u.scalingFactor$ 
12:         $F_R(v) \leftarrow -repulsion * v.scalingFactor$ 
13:      end if
14:    end if
15:  end for
16: end for

```

Algorithm 7 Calculate gravitational forces (calculateGravitationalForces)**Input:** Module (graph containing subgraphs)**Output:** Gravitational forces F_G for all nodes

```

1: for all  $v \in V$  do
2:   if  $v$  is child of the parent module then
3:      $F_G(v) = (globalGravityCenter - v.center) * g * v.scalingFactor$ 
4:   else
5:      $F_G(v) = (v.parent.gravityCenter - v.center) * g * v.scalingFactor$ 
6:   end if
7: end for

```

Algorithm 8 Calculate magnetic forces (calculateMagneticForces)**Input:** Module (graph containing subgraphs), direction of magnetic field**Output:** Magnetic forces F_M for all nodes

```

1: for all  $e = (u, v) \in E$  do
2:    $edgeDirection \leftarrow \text{calculateEdgeDirection}(u, v)$ 
3:    $\phi \leftarrow \text{calculateAngle}(edgeDirection, magneticFieldDirection)$ 
4:    $F_M(u) \leftarrow \phi * u.scalingFactor$ 
5:    $F_M(v) \leftarrow \phi * v.scalingFactor$ 
6: end for

```


Selbständigkeitserklärung (§5, Absatz 2b)

Hiermit erkläre ich, dass ich diese eingereichte Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Frühere Bewerbungen um einen Doktorgrad (§5, Absatz 2c)

Hiermit erkläre ich, dass ich mich bisher um keinen weiteren Doktorgrad beworben habe.

Frühere Promotionsversuche (§5, Absatz 2d)

Hiermit erkläre ich, dass ich bisher keine vergeblichen Promotionsversuche unternommen habe.