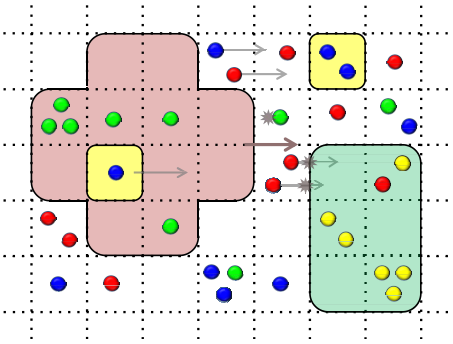
ML-Space: Hybrid Spatial Gillespie and Brownian Motion Simulation at Multiple Levels, and a Rule-based Description Language



Dissertation
zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

vorgelegt von
Arne T. Bittig, geb. am 21.12.1980 in Cottbus
aus Rostock

urn:nbn:de:gbv:28-diss2017-0047-4

Principal advisor	Prof. Dr. rer. nat. habil. Adelinde M. Uhrmacher *Universität Rostock*
Other reviewers	Prof. James R. Faeder, PhD *University of Pittsburgh*
	Prof. Dr. techn. Heinz Koeppl *Technische Univerität Darmstadt*
Date of submission	07.09.2016
Date of defense	31.01.2017

Abstract

Computer models and simulations of micro-biological processes facilitate structuring knowledge and making testable predictions about their behavior. Methods that treat cells like well-stirred systems are well established in that regard, but the spatial distribution of key actors often cannot be neglected. Spatio-temporal dynamics of cellular processes can be simulated at different levels of detail, from (deterministic) partial differential equations via the spatial Stochastic Simulation algorithm to tracking Brownian trajectories of individual particles.

In this thesis, a spatial simulation algorithm for multi-level models, including dynamically hierarchically nested cellular compartments and entities is presented. The approach, called ML-Space, combines stochastic spatial algorithms in discretized space, i. e. population- and subvolume-based simulation, with individual particles moving in continuous space that have spatial extensions and can contain other particles. For a formal description of the systems to be simulated spatially, ML-Space provides a rule-based specification language. It incorporates spatial properties of the model actors via named attributes and supports concise and compact descriptions of models and allows easy adaptation of the spatial resolution, while abstracting from the concrete simulation algorithm as much as possible.

ML-Space has successfully been used to model disease-relevant aspects of various cellular processes, including mitochondrial fission dynamics and actin filament formation. Additionally, existing spatial models with multi-level aspects were successfully reproduced and an abstract multi-level model was developed to explore the simulation approaches' capabilities and performance.

Zusammenfassung

Computermodelle und Simulationen mikrobiologischer Prozesse erleichtern es, das Wissen über diese zu strukturieren und falsifizierbare Vorhersagen über deren Verhalten zu machen. Insbesondere Methoden, bei denen eine homogene Verteilung von Molekülen innerhalb einer Zelle angenommen wird, sind gut etabliert. In vielen Systemen ist die räumliche Position der Akteure aber eben doch relevant.

Räumlich-zeitliche Dynamiken zellulärer Prozesse können mit verschieden Detailgraden simuliert werden, von (deterministischen) partiellen Differentialgleichungen über den Räumlichen Stochastischen Simulationsalgorithmus bis hin zu individuellen Partikeln mit jeweils eigener Trajektorie, der Brownschen Molekularbewegung folgend. In dieser Arbeit wird ein räumlicher Simulationsalgorithmus für Mehrebenenmodelle vorgestellt, der auch dynamische hierarchische Strukturen von Zellen und Entitäten darin abdeckt. Dieser ML-Space genannte Ansatz vereint stochastische räumliche Algorithmen in diskretisiertem Raum, d.h. Populations- und Subvolume-basierte Simulation, mit individuellen Partikeln mit kontinuierlichen Koordinaten, die andere Partikel enthalten können.

Zur formalen Beschreibung der räumlich zu simulierenden Systeme bietet ML-Space eine regelbasierte Modellierungssprache. Räumliche Eigenschaften der Akteure werden über Attribute mit speziellen Namen spezifiziert, was präzise und kompakte Beschreibungen der Modelle und einfache Anpassung der räumlichen Auflösung erlaubt, aber gleichzeitig von der eigentlichen Umsetzung des Simlationsalgorithmus abstrahiert.

Mit ML-Space wurden krankheitsrelevante Aspekte verschiedener zellulärer Prozesse erfolgreich modelliert und simuliert, insbesondere die Regulierung mitochondrialer Teilung und die Bildung von Aktinfilamenten. Außerdem wurden existiertende Modelle erfolgreich reproduziert und ein abstraktes Modell zur Veranschaulichung der Simulator-Einsatzmöglichkeiten und Performance entwickelt.

Contents

Glossary / Terms and Acronyms

τ-leaping

Approximate realization of the chemical master equation (CME) where several reactions (happening in a time interval τ) are applied before propensities are recalculated (Gillespie 2001). Sacrifices accuracy for simulation speed compared to other implementations of the Stochastic Simulation algorithm (SSA). 14, 74, 86

chemical master equation

Differential equation (or a set thereof) describing the time evolution of the *probabilities* for a system to be in a certain (discrete) state when regarding chemical reactions as Markov processes (see also CTMC; Gillespie 1992). In contrast ODEs usually describe the time evolution of the state itself (then a continuous variable). 4, 12, 14, 19

CME

chemical master equation → v

combinatorial explosion

Macromolecules often can be modified at several different positions and some reaction may modify one single position independently of the state of the others. Such a reaction would then apply to many different forms of the actual macromolecules (a *multi-state* species), and would require specification of many rules or equations in approaches without special considerations for this combinatorial explosion. The term is also used more generally for problems where increasing the number of dimensions (here: the number of modifications positions) increases some (cost) function rapidly. 25, 30, 116

continuous-time Markov chain

A stochastic process with a finite or countable state space and a continuous time base whose future behavior depends only on the current state, not on any past state ("Markov property"). The time spent in each state follows an exponential distribution. 4, 11, 27, 28

CTMC

continuous-time Markov chain → v

Direct method

Exact realization of the CME by Gillespie (1976) and Gillespie (1977), along with the First Reaction method (FRM). 13, 45, 48, 50

First Reaction method

Exact realization of the CME by Gillespie (1976) and Gillespie (1977), along with the Direct method. 13

FRM

First Reaction method → vi

Law of Mass Action

Kinetic law for elementary reactions roughly in equilibrium. Here used for rate laws where the rate of a reaction is directly proportional to (the product of) the concentration of reactant(s), or the amounts or reactants in stochastic simulation. 10, 20, 23, 33, 54, 55, 113

Michaelis-Menten kinetics

Kinetic law for reactions mediated by an enzyme whose amount is constant. Assuming the enzyme E can reversibly bind a substrate S, that the concentration of enzyme-substrate complex is roughly constant (quasi-steady-state or equilibrium assumption) and that the enzyme-substrate complex may eventually react, non-reversibly, to a product and a free enzyme, the overall reaction from substrate to product can be described by two constants (the maximum reaction rate V_{max} and the Michaelis constant K_M giving the substrate concentration at which the actual rate will be $V_{max}/2$) instead of four (three mass action rates for the three relevant reactions and the total enzyme concentration). 20, 23, 26, 55

Next Reaction method

Exact realization of the CME introduced by Gibson and Bruck (2000), optimized for higher performance by using a priority queue structure to schedule reactions (i. e. store their times and efficiently retrieve those) and a reaction dependency graph to avoid some propensity recalculations, as well as by rescaling some random numbers (avoiding generation of new ones). 14, 45

NRM

Next Reaction method → vi

ODE

ordinary differential equation → vii

order of a reaction

Number of entities (usually molecules) required for a reaction to happen. Corresponds to the sum of exponents of species concentrations/amounts in the reaction rate. For example, the reaction $1\,A + 2\,B \to ...$ has order 3, and the rate would be $k[A]^1[B]^2$ for some rate constant k (the right hand side, i. e. the products, do not matter for the order). For individual-based simulation, reactions of order higher than 2 are usually broken down into steps of order 2 with intermediate reactants. 20, 43

ordinary differential equation

An equation describing the change in one variable, here usually concentrations over time as a function of this variable (and possibly others): $\frac{\mathrm{d}f}{\mathrm{d}x} = f(x)$. *Systems* of ODEs $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = f(\mathbf{x})$ are often used for non-spatial modeling and simulation of processes where concentrations are assumed to be high. These models are deterministic. 4, 10, 12, 23, 26, 28, 49

partial differential equation

Differential equation involving a multivariate function and its partial derivatives. While ODEs are used, for our purposes, to describe a quantity's change over time, PDEs can be used to specify spatio-temporal changes. They are then based on functions of time *and* space, i. e. coordinates. 18, 74, 115

PDE

partial differential equation → vii

RDME

reaction-diffusion master equation → vii

reaction-diffusion master equation

Extension of the chemical master equation (CME) from a state space $\{X_j\}$ to $\{X_{j,\mathbf{r}}\}$ covering species quantities in lattice cells (subvolumes), with an additional (diffusion coefficient D_j and lattice size ℓ dependent) term for diffusion events from one subvolume $\mathbf{r}$ to a neighboring one $\mathbf{r}'$ (Gardiner et al. 1976; Nicolis and Prigogine 1977). 19, 27, 34, 44, 45, 50, 57

SBML

Systems Biology Markup Language → viii

Separation of concerns

Specification of model and simulator independently, i. e. the modeler does not need to know about the exact implementation of the simulation algorithm, as

the model is specified in a defined modeling formalism. This also facilitates model reuse and composition. Properties (numerical or qualitative) to be observed during the simulation may also be separated from both model description and simulator specification. In spatial cell biological simulation, separation of concerns can also be applied to species, reactions and their parameters (which represent parts of the modeled real system) and the spatial resolution like subvolume size or movement steps (which are parameters of the simulator). 2, 22, 55, 73

Separation of scales

Method for analyzing complex systems by grouping processes by their relevant time scales into fast, dynamic and slow, then assuming slow processes to be fixed and studying the fast processes first in isolation and using their average behavior when investigating their interactions with the other dynamic processes. (Bar-Yam 2011) 3

SSA

Stochastic Simulation algorithm → viii

Stochastic Simulation algorithm

"A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions" popularised by Gillespie (1976) in a paper of that name, also used as umbrella term for different *exact* realizations of the CME (e. g., FRM, Direct method, NRM, but not τ-leaping). 4, 11, 19, 26, 44

Systems Biology Markup Language

Standardized exchange format for models in systems biology based on XML (Hucka et al. 2003). SBML is supported by the vast majority of tools for simulating non-spatial and compartmental models. It was not designed to be human-readable, and support for spatial models beyond discrete compartments is limited (packages extending the SBML 3 core exist in draft status as of summer 2016). 22

List of Figures

Copyright Notice

List of Tables

List of Model and Other Listings

List of Algorithms

1 Introduction

1.1 Why Model Cell Biological Systems?

Since the first quantitative description of a chemical reaction in the 19th century (Wilhelmy 1850), tremendous progress has been made towards understanding the interplay of chemical agents in biological cells. The life sciences, however, are still far from a thorough explanation of mechanisms that have evolved over thousands, if not millions of generations. Reasoning about cellular function *bottom-up*, i. e. from first principles, is hampered by the huge number of involved actors (molecules) and the complexity of their interaction network.

Prominent systematic investigations in the other direction, i. e. *top-down*, date back more than a century, too, as exemplified by what is now known as Mendelian inheritance laws (Mendel 1866), where the outward appearance of organisms (the phenotype) was explained by information carrying agents (genes) whose exact nature and function became clear only much later.

However, thorough investigations of biological organisms to this day are not possible by observation alone. On the level of chemical reactions, even tracking the amount of one chemical species in one cell can be a time- and/or money-consuming process that delivers results with not always high accuracy, and this is before possible cell-to-cell variations are taken into account. Therefore, models of the studied processes are needed not only as abstract explanations of directly observed phenomena (as customary in the scientific method), but already as means to simulate parts of the system whose direct measurement is impractical or impossible (Kitano 2002).

The models of interest in cell biology are thus executable in nature (as opposed to merely conceptual), meaning that given certain, possibly quantitative, inputs they should make testable predictions about the future state of the modeled system. We are thus talking about mathematical or computational models.[1]

With computational models, it is not uncommon to see the actual model, i. e. the conceptual part regarding the rules of the interactions of model components, implemented directly in a programming language, i. e. intertwined with the simulator. This may

[1] When considering a computer as a deterministic system executing a list of rules, mathematical models can be considered to be a superset of computational models. However, the former is often used for models that can best be described in the form of mathematical equations that may even have an analytic solution, while the latter is used for models whose complexity and/or nonlinearity requires investigation by simulation, i. e. execution of the model on a computer. Mathematical models in the former sense, however, are often analyzed by, numerically integrated on and even formulated using a computer.

be unavoidable when the approach taken is sufficiently new or the model is large and incorporates aspects not usually modeled with the same formalism (Karr et al. 2012; Waltemath et al. 2016). However, when using established approaches, separation of concerns, i. e. of model and simulation, facilitates not only understanding of the model by outsiders, but also model reuse and composition.

1.2 Multi-Level, Multi-Scale and Multi-Resolution Simulation

One of the most useful abstraction in describing complex systems is to split them into parts that can be described rather independently of other parts and where the interactions among parts then are described without direct consideration of their inner life. The interactions among parts and the mechanisms inside parts are then considered different organizational levels.

Multi-level modeling will thus reflect the structure of the knowledge in the domain. This interpretation of the term is in line with its use in organizational research, social sciences, and the statistics where multi-level refers to study interacting processes taking place at different organizational levels of society (Tilly 1997), or to the modeling and simulation of a society at individual and population level (Möhring 1996).

For biological examples, consider a mammalian organism. It can be roughly understood as a composition of organs of different function without knowing the details of how a liver or a brain works (cf. Figure 1.1, right). Similarly, once the function of a protein (enzyme) is established with reasonable certainty, its exact molecular composition and formation are not directly relevant for the overall role of the enzyme in a cellular environment (although knowing the amino acid composition of two enzymes may explain why they can or cannot interact; Figure 1.1, left).

In this view, a complex system is decomposed into a graph structure similar to a tree. However, there may be actors relevant to several components on the level above, for example enzymes having a role in different pathways (which are abstract groups of enzymes and reactions relevant for a certain cellular function), a notion also termed *crosstalk*. When reasoning from dynamics at one level to the level above or below, the metaphors of *upward* and *downward causation* are commonly used (Noble 2012).

It is often the case that components at different levels differ in size. The time frame of interest for their interactions is also smaller on the lower organizational levels. *Multi-scale modeling* or simulation refers to a scale in, e. g., size, numbers, time, or space and requires to take widely spread areas of this scale in the phenomena being studied into account, for example both small inorganic molecules and large protein complexes, processes working on nanoseconds and hours, or some molecules available in vast abundance whereas others are rare. Since these scales are a straight-forward way to categorize similar or related system components, the term *multi-scale* is often encountered without explicit reference to organizational levels, even when different scales could be interpreted such. Specifically,

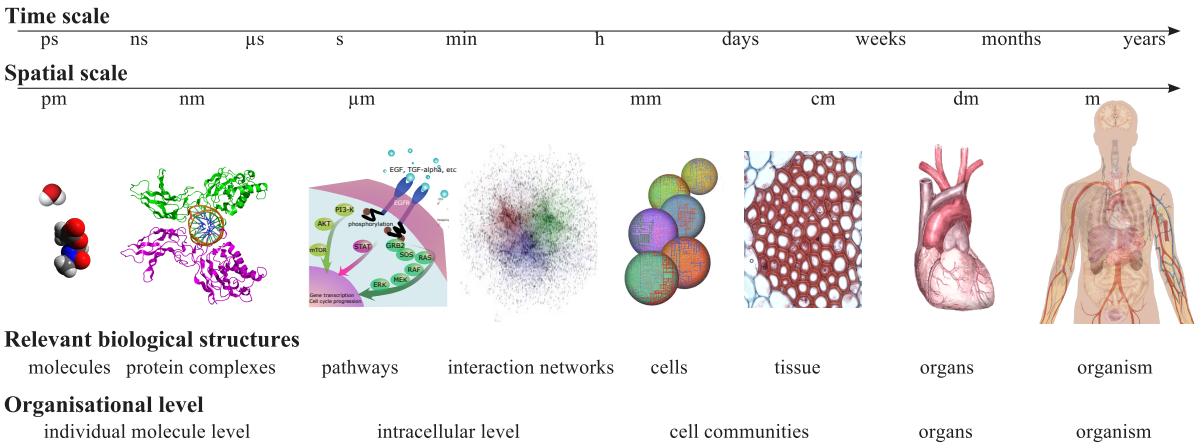


Figure 1.1. Structures relevant in biology, their spatial extension and the time scales at which they are often considered, exemplifying the use in the terms *level* and *scale* in systems biology.

separation of scales means taking an implicit multi-level view of a system where parts that are interesting at a small time scale are first studied in isolation and their average behavior stands in for them when processes at large time scales are studied.

The term multi-resolution, finally, may be found associated with simulation algorithms that combine several approaches, which may encompass multi-level or multi-scale models (Takahashi et al. 2004) or may not, e. g., when entities at the same level are simulated with different levels of granularity (Jeschke and Uhrmacher 2008). ML-Space, the approach laid out in this thesis, targets all aspects: multi-resolution or single-algorithm simulation of single or multi-level models.

Note that models with multi-level aspects may still be "flattened" and simulated in classic, single-level approaches, for example when reactions in the cytosol and those in the nucleus are expressed in the same set of differential equations, distinguished only by variable names (e. g., Lai et al. 2009).

1.3 Models, Formalisms and Languages

Each model about a real system must be encoded in some formal framework to allow predictive, repeatable experiments on the model. This may simply be done by finding appropriate data structures to represent model entities and implementing their interactions on a computer. Then, the programming language used could be considered the formal framework. However, this means mixing implementation detail (what can be represented on a computer and how the data structures can be manipulated) with the model detail, which can be especially inconvenient when scientists from different fields come together, i. e. when the modeler or the domain experts and target audience for the model's predictions have no computer science background.

Thus, a suitable layer of abstraction has to be found. A modeling formalism must therefore provide two things: a model specification format and an execution algorithm (Sarjoughian and Zeigler 2000). In the words of Sarjoughian (2006), "[t]he former is a mathematical theory describing the kinds of structure and behavior that can be described with it. The latter specifies an algorithm that can correctly execute any model that is described in accordance with the model specification."

Ordinary differential equations (ODEs) can be considered such a formalism: They are expressed in, essentially, the language of mathematics and describe changes in one variable in relation to other variables, and executing them simply means solving them mathematically. When nonlinearity forbids analytical solutions, different algorithms for deriving numerical approximations are available. Differential equations, however, are not a particularly intuitive way of describing systems of interacting entities. Rule-based modeling languages (Hlavacek et al. 2006; Chylek et al. 2015), an approach that gained popularity for being able to deal with multi-state entities like proteins with different modification sites, are a convenient alternative. Here, entire models can be expressed basically by transformation rules giving substrates (input), products (output) and a rate. These models can be translated to differential equations for deterministic execution (with some limitations regarding combinatorial complexity) or be executed stochastically with continuous-time Markov chain (CTMC) semantics.

1.4 Determinism, Stochasticity and Space

Simulations of *deterministic* models will always predict the same system trajectory for the same set of model inputs (parameters, initial state).[2] This also makes them relatively easy to fit to a limited amount of data. They are most popular in the form of ODEs, where systems are assumed to be continuous.

This assumption is not always reasonable, for example when copy numbers (naturally discrete values) of some molecules are low, or (not mutually exclusive) when bi- or multistability of a system is not adequately captured or when inherent noise in the system can induce different outcomes, e. g., cell fates (Pahle 2009).

Stochastic models are most commonly based on the assumption of chemical systems as memoryless processes: reactions happen in random time intervals with the time to each next reaction independent of the time since the last reaction, only dependent on the current system state (McQuarrie 1967). This CTMC interpretation gives rise to a probabilistic description of the system evolution via the chemical master equation (CME) and simulation via variants of the Stochastic Simulation algorithm (SSA) (Bortz, Kalos, and Lebowitz 1975; Gillespie 1976; Gillespie 1977). (For an overview of hybrid methods combining stochastic and deterministic approaches, see Pahle (2009), Table 1. Note that we will subsequently use the term *hybrid* differently, i. e. to refer to our combination of two stochastic spatial approaches.)

[2] except when the limitations of the numerical integration are reached, e. g., for stiff systems

Common to these approaches as described so far is the assumption of a well-mixed system. However, some molecules may localize to certain regions, e. g., near the cell wall or near the nucleus, and thus be less likely to encounter certain others, or be located in different compartments entirely. (The latter case may be covered by coarse-grained compartmentalization already indicated at the end of section 1.2; cf. Kholodenko, Hancock, and Kolch 2010.) Spatial dynamics are relevant, for example, in processes where an extracellular signal is sensed at the cell's boundary and triggers a cascade of reactions eventually resulting in transcription of some DNA segments to produce certain proteins as a response to the signal. It should also be noted that the cell is a crowded environment including large entities that do not move easily towards a well-mixed state.

Spatial dynamics can be covered with deterministic or stochastic approaches. In the former case, they are expressed via gradients of concentrations and described via partial differential equations. A somewhat analogous approach for stochastic models is to split the region of interest into virtual subunits, each presumed well-mixed, with appropriate interchange of components between them. Alternatively, one may consider molecules individually, with a position in continuous-space for each. Their movements and interactions can then again be simulated deterministically (via the physical forces the molecules exert on each other; Molecular dynamics) or stochastically (starting from random movement of large molecules in a solvent of smaller molecules; Brownian motion). As spatial simulation is an integral part of this thesis, these methods are compared in more detail in the next chapter.

1.5 Contributions of ML-Space

For this dissertation (and a few publications that preceded it, cf. section 1.7), I developed a modeling language for spatial and multi-level phenomena. Key part of a model description are entities of different types and attribute configurations, and their interactions as expressed by reaction rules. The language is based on ML-Rules (Maus, Rybacki, and Uhrmacher 2011; Maus 2013), but due to its spatial aspects has significantly different execution semantics.

For this part, the language was aimed not only to abstract from the implementation of its simulator, but from the actual simulation algorithm used therein. This allows us to provide different types of simulators, specifically one based on stochastic reaction-diffusion simulation of dimensionless entities (which alone is not applicable to models with dynamics on multiple levels, however; Stundzia and Lumsden 1996; Elf and Ehrenberg 2004) as well as one based on particle-tracking, Brownian motion approximating simulation. The key aspect of the ML-Space simulator is then the developed hybrid approach, a combination of these two with adaptations for the models' potential multi-level structure.

The implementation is aimed to allow for relatively easy combination with other spatial simulation strategies, e. g., deterministic concentration changes over space and time (via

partial differential equations or a suitable approximation; Kossow et al. 2015), but this was not yet the focus of this work.

ML-Space has successfully been applied to two different biological problems, firstly patterns in the growth of actin filaments and secondly mitochondrial interactions. For these two cases (unlike the more abstract test problems also introduced later), the focus of ML-Space had to be shifted a bit to binding operations between entities, which now offer a possible extension point for eventually describing molecule formation and Molecular Dynamics models in ML-Space as well.

Availability

ML-Space' source code is available at bitbucket.org/jamesii as part of JAMES II under the latter's open source license (BSD/GPL dual license). A simple, stand-alone version, i. e. a `zip` file containing a runnable `jar`, a short pdf guide and examples, is available as supplementary material of Bittig and Uhrmacher (2016) or at dropbox.com/s/wauhdyy345on272/MLSpace-Sandbox-201608.zip.

1.6 Structure of the Thesis

In the next chapter, the foundations for ML-Space will be briefly summarized, first the approaches usually applied for simulation cell biological system, with a focus on spatial methods, then the modeling approaches useful for specifying cell biological models, with a focus on rule-based languages.

ML-Space' contribution to these two aspects, modeling and simulation, will be investigated in the two chapters that follow. Chapter three introduces the developed rule-based modeling language for spatial simulation and the semantics as far as rule evaluation is concerned, i. e. the non-spatial aspects. The fourth chapter is devoted to the spatial semantics and the hybrid simulation algorithm developed. It also briefly covers the the software architecture of the implementation and how it is reconciled with the modular, plug-in based framework JAMES II upon which it is based.

Applications of ML-Space, both the modeling language as well as various aspects of the simulator, are presented in chapter 5. The thesis concludes with a summary and discussion of what ML-Space is and what it is not.

1.7 Previous Publications on ML-Space

An overview of modeling approaches that involved or could potentially be adapted for spatial aspects was presented at the Winter Simulation Conference 2010. It forms the basis for two sections in the (following) background chapter: Modeling Formalisms and Languages (2.4), which, however, is more focused on rule-based languages in particular, and Simulation: Adding Space (2.3).

Bittig and Uhrmacher (2010) Bittig, Arne T. and Adelinde M. Uhrmacher (2010). "Spatial Modeling in Cell Biology at Multiple Levels". In: *Proceedings of the 2010 Winter Simulation Conference*. Baltimore, MD, USA: IEEE Computer Science, pp. 608–619. © 2010 IEEE

First ideas towards the ML-Space language and simulator were presented at two further conferences.

Bittig, Jeschke, and Uhrmacher (2010) Bittig, Arne T., Matthias Jeschke, and Adelinde M. Uhrmacher (2010). "Towards Modelling and Simulation of Crowded Environments in Cell Biology". In: *ICNAAM 2010: International Conference of Numerical Analysis and Applied Mathematics*. Vol. 1281. 1. Rhodes, Greece: AIP, pp. 1326–1329.

Bittig et al. (2011) Bittig, Arne T., Fiete Haack, Carsten Maus, and Adelinde M. Uhrmacher (2011). "Adapting rule-based model descriptions for simulating in continuous and hybrid space". In: *Proceedings of the 9th International Conference on Computational Methods in Systems Biology*. CMSB '11. Paris, France: ACM, pp. 161–170.

A comprehensive explanation of the actual implementation of language and simulator is, however, only to be found in a later publication, which can be regarded as a condensed subset of the present thesis' chapters 3 and 4. Bindings between entities, for example, were not discussed there due to spatial constraints.

Bittig and Uhrmacher (2016) Bittig, Arne T. and Adelinde M. Uhrmacher (2016). "ML-Space: Hybrid Spatial Gillespie and Particle Simulation of Multi-level Rule-based Models in Cell Biology". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* PP. © 2016 IEEE

Before preparing this methodology paper, we focused on an application in microbiology, namely bone cells growing on structured titanium surfaces. We also briefly explored ML-Space' applicability to investigating mitochondrial health and reactive oxygen stress. Several aspects of both topics represent opportunities for future investigations.

Bittig et al. (2014a) Bittig, Arne T., Claudia Matschegewski, J. Barbara Nebe, Susanne Stählke, and Adelinde M. Uhrmacher (2014a). "Membrane related dynamics and the formation of actin in cells growing on micro-topographies: a spatial computational model". In: *BMC Systems Biology* 8, pp. 106+. CC-BY (Open Access)

Bittig et al. (2014b) Bittig, Arne T., Florian Reinhardt, Simone Baltrusch, and Adelinde M. Uhrmacher (2014b). "Predictive Modelling of Mitochondrial Spatial Structure and Health". In: *Computational Methods in Systems Biology*. Vol. 8859. Lecture Notes in Computer Science. Springer International Publishing. Chap. 20, pp. 252–255. © Springer International Publishing Switzerland 2014

Large-scale data sets from ML-Space simulations were analyzed with visualization methods developed in the Computer Graphics group of the University of Rostock. These aspects are not covered in detail in this thesis.

Luboschik et al. (2012) Luboschik, Martin, Christian Tominski, Arne Bittig, Adelinde M. Uhrmacher, and Heidrun Schumann (2012). "Towards Interactive Visual Analysis of Microscopic-Level Simulation Data". In: *Proceedings of SIGRAD 2012 – Interactive Visual Analysis of Data*. Växjö, Sweden, pp. 91–94.

Eichner et al. (2014) Eichner, Christian, Arne Bittig, Heidrun Schumann, and Christian Tominski (2014). "Analyzing simulations of biochemical systems with feature-based visual analytics". In: *Computers & Graphics* 38, pp. 18–26.

Luboschik et al. (2015) Luboschik, M., M. Röhlig, A. T. Bittig, N. Andrienko, H. Schumann, and C. Tominski (2015). "Feature-Driven Visual Analytics of Chaotic Parameter-Dependent Movement". In: *Computer Graphics Forum* 34.3, pp. 421–430.

2 Background

2.1 Basics: Reaction Networks

In this chapter, I will briefly revisit the methodological foundations for simulating models of biological systems, and approaches for describing such models.

The dynamics of the modeled systems are, for our purposes, abstracted to *reaction networks*, i. e. sets of chemical reactions, each consisting of *reactants* and *products* where the set of all reactants usually intersects the set of all products.

Formally, we have a set of species $S = \{S_1, ..., S_n\}$ in a system volume V, a state vector $\mathbf{x}(t) = (x_1(t), ..., x_n(t))^T$ containing the quantity of each species over time, and a set of reactions $R = \{R_1, ..., R_m\}$. Each reaction is a description of conversion of some species into others given by stoichiometric coefficients y with a certain rate depending on a constant k_i:

$$R_i : \sum_{j=1}^{n} y_{ij}^r S_j \xrightarrow{k_i} \sum_{j=1}^{n} y_{ij}^p S_j. \tag{2.1}$$

Simulating the reaction network then means to deduce the time evolution of $\mathbf{x}(t)$ given S, V, R and an initial state $\mathbf{x}(0)$.

As a simple reaction network example, consider an enzyme E catalyzing the conversion of a substrate S into a product P where the enzyme-substrate complex can also decompose back into enzyme and unconverted substrate.

$$\begin{aligned} R_1 &: E + S \xrightarrow{k_1} ES \\ R_2 &: ES \xrightarrow{k_2} E + S \\ R_3 &: ES \xrightarrow{k_3} E + P \end{aligned} \tag{2.2}$$

Regarding the species set $\{E, S, ES, P\}$, the first reaction here decreases the quantity of E and S and increases the quantity of ES. Then, assuming the state vector refers to the species' quantities in the given order, $\mathbf{y_1^r} = (1, 1, 0, 0)$ and $\mathbf{y_1^p} = (0, 0, 1, 0)$, giving an overall stoichiometry of $\mathbf{y}_1 = (-1, -1, 1, 0)$ – the (consumed) reactants are included with a negative sign. The *stoichiometric matrix* $\mathbf{N}$ is the $n \times m$ matrix composed of these $\mathbf{y}_i$ as columns.

Note that the overall stoichiometry does not give full information about the reaction when a reactant is also a product, i. e. unchanged. Examples would be translation of of mRNA to a protein, after which the mRNA is usually still available ($R : mRNA \rightarrow$

$mRNA + Protein$), or the presence of a mediating actor (like the enzyme in the overall reaction of the above network, $E + S \to E + P$).

Note also that one side of a reaction may be empty, i. e. either $\mathbf{y^p} = \mathbf{0}$ or $\mathbf{y^r} = \mathbf{0}$, when one does not wish to include a reactant that is assumed to be constant or a product that is irrelevant. An example of the former would be translation of a gene to mRNA (intuitively, $R : gene \to gene + mRNA$): the number of genes encoding each mRNA sequence is usually constant, thus one may simply write $R : \varnothing \to mRNA$ (with a different interpretation of the associated rate constant).

While reaction networks often describe dynamics at the level of molecules, they are also applicable at different scales (and levels), for example for population dynamics covering birth and death rates of biological species in response to external conditions, e. g., food availability, or when one species preys on another.

2.2 Simulation: Stochastic or Deterministic

2.2.1 Deterministic Simulation

An *ordinary differential equation (ODE)* describes the change of a variable over time in relation to this and possibly other variables. Assuming the reacting species are distributed homogeneously in space, reaction networks can be directly translated to systems of ODEs by interpreting the state vector in terms of species concentrations and the reaction rate constants as constant factors for the rate of change. The enzyme catalysis network of Equation 2.2 would be expressed (with square brackets denote the (time-dependent) concentrations of the respective species, i. e. $\mathbf{x}(t) = ([E], [S], [ES], [P])$)

$$\begin{array}{lcll} \frac{\mathrm{d}[E]}{\mathrm{d}t} & = - & k_1[E][S] + k_2[ES] & +k_3[ES] \\ \frac{\mathrm{d}[S]}{\mathrm{d}t} & = - & k_1[E][S] + k_2[ES] & \\ \frac{\mathrm{d}[ES]}{\mathrm{d}t} & = & k_1[E][S] - k_2[ES] & -k_3[ES] \\ \frac{\mathrm{d}[P]}{\mathrm{d}t} & = & & k_3[ES], \end{array} \tag{2.3}$$

or, more generally, $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{N} \cdot \mathbf{v}(t)$, where $\mathbf{N}$ is again the stoichiometric matrix and $\mathbf{v} = (v_1, ..., v_m)^T$ is a column vector of reaction rates v_i, which are a function of the kinetic rates and substrate concentrations for the respective reactions (more complex kinetics than the above mass action kinetics are discussed in section 2.3.6).

Solving a set of ordinary differential equations (usually numerically, i. e. by approximating computations, as analytic, let alone closed form solutions are usually unavailable) gives one time course for the variables per vector of initial conditions, in terms of real-valued concentrations.

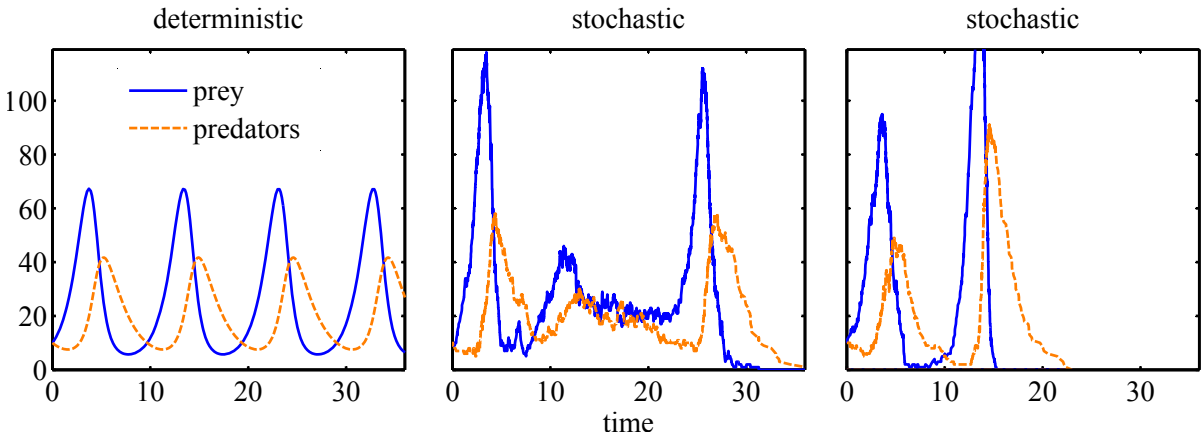


Figure 2.1. Deterministic and stochastic simulation of a Lotka-Volterra- (predator-prey)-system. The deterministic system produces stable oscillations of population sizes, the corresponding stochastic simulations result in somewhat oscillatory behavior until one species eventually dies out (i. e. its population size reaches 0).

2.2.2 Stochastic Simulation

Stochastic methods capture the effects of random fluctuations. A straightforward way to achieve this is adding a *noise term*, i. e. a continuous random variable ξ, usually normally distributed, to the right side of a differential equation. Numerically solving these *stochastic differential equations* then leads to different solutions, called *realizations*, per attempt (i. e. simulation run). The mean of many runs converges to the deterministic solution ideally, i. e. when the variance of the random noise is sufficiently small.

In the kind of stochastic approach more relevant to the present work, however, species' quantities are represented by particle amounts (copy numbers), i. e. integer numbers, and not approximated by concentrations. This is particularly relevant when copy numbers are low (cf. Figure 2.1).

Current popular stochastic simulation methods are based on what is now known simply as the *Stochastic Simulation algorithm (SSA)* or Gillespie's method after the author who popularized them (Gillespie 1976). It is similar to an earlier method by Bortz, Kalos, and Lebowitz (1975) for an application outside cell biology (which may be better known in the physics community, where it is also referred to as Kinetic Monte Carlo (KMC) or BKL after the authors, who called their method "the n-fold way"), although its roots go back even further (Kolmogoroff 1931; Feller 1940; Doob 1945; Kendall 1950).

In these approaches, the system is essentially a continuous-time Markov chain (CTMC), changing its state in jumps with exponentially distributed time between them. From the probability $a_i(\mathbf{x})\Delta t$ of reaction i happening in a small time interval $[t, t + \Delta t]$ given a current state $\mathbf{x}(t)$, one can derive the probability function for the system to be in a state

$\mathbf{x}$ at time $t + \Delta t$ as

$$P(\mathbf{x}, t + \Delta t) = P(\mathbf{x}, t)\big[1 - \sum_{i=1}^{m} a_i(\mathbf{x})\Delta t\big] + \sum_{i=1}^{m} P(\mathbf{x} - \mathbf{y}_i, t)a_i(\mathbf{x} - \mathbf{y}_i)\Delta t, \tag{2.4}$$

where the first summand on the right hand side corresponds to the probability that the system was in the same state before and did not change, while the following sum covers the cases where a reaction fired and the system was previously in the state that differed from the current one by the relevant stoichiometry vector $\mathbf{y}_i$. The reaction *propensities* $a_i = h_i k_i$ are the product of the number of distinct molecular reactant combinations for reaction R_i in the system at the current time, h_i (dependent on the current state $\mathbf{x}$; omitted for shorter notation), and the reaction rate (or *reaction parameter*) k_i related to the probability $k_i \Delta t$ that a single particular reactant combination will react in the next time interval Δt.

Taking the limit $\Delta t \to 0$ in Equation 2.4 and mathematical transformations eventually yield the chemical master equation (CME), which gives the state probability density over time:

$$\frac{\partial P(\mathbf{x}, t)}{\partial t} = \sum_{i=1}^{m} P(\mathbf{x} - \mathbf{y}_i, t)a_i(\mathbf{x} - \mathbf{y}_i) - P(\mathbf{x}, t)\sum_{i=1}^{m} a_i(\mathbf{x}) \tag{2.5}$$

While there is no established format for describing stochastically modeled systems mathematically like ODEs for deterministic models, some form of rule representation like in Equation 2.2 is common. Note that while it is fairly straight-forward to derive ODE systems from such rules (e. g., Equation 2.3), the reverse is not true. Consider the ODEs behind Figure 2.1 (left):

$$\begin{array}{lcll} \frac{\mathrm{d}Prey}{\mathrm{d}t} & = & \alpha \cdot Predator - \beta \cdot Predator \cdot Prey & \alpha = 1 \quad \beta = 0.05 \\ \frac{\mathrm{d}Predator}{\mathrm{d}t} & = - & \gamma \cdot Predator + \delta \cdot Predator \cdot Prey & \gamma = 0.5 \quad \delta = 0.02 \end{array} \tag{2.6}$$

Here, the interaction of predator and prey has effects of different magnitude on each population ($\beta \neq \delta$). Intuitively, predator reproduction depends on prey availability, but not every time a predator consumes a prey a new predator is created, so this interaction must be covered by multiple rules (assuming $\beta > \delta$):

$$\begin{array}{rcl} R_1 : Prey & \xrightarrow{\alpha} & Prey + Prey \\ R_{2a} : Prey + Predator & \xrightarrow{\delta} & Predator + Predator \\ R_{2b} : Prey + Predator & \xrightarrow{\beta-\delta} & Predator \\ R_3 : Predator & \xrightarrow{\gamma} & \varnothing. \end{array} \tag{2.7}$$

In the following subsections, we will have a brief look at the most relevant SSA approaches. Overviews or comparisons of several related methods can be found in (Cao, Li, and Petzold 2004; Pahle 2009). In each case, a basic simulation step consists of

generating a random duration τ after which the next event will occur and an index $i \in \{1..m\}$, then advancing the simulation time t by this *sojourn time* τ and executing reaction R_i by changing the state vector by the respective stoichiometry vector. This step is repeated until a desired stop condition, often $t > t_{stop}$ for some user-defined t_{stop}, is fulfilled. The methods differ in how the (τ, i) pairs are derived. .

Direct Method

In the Direct method (Gillespie 1976), the sum of propensities

$$a_0 = \sum_{\alpha=1}^{m} a_\alpha \tag{2.8}$$

is used to calculate the time step

$$\tau = -\frac{1}{a_0} \ln r_1, \tag{2.9}$$

where r_1 is a random number from the uniform distribution on $(0, 1]$ (and hence $-\ln r_1$ is a random number from the exponential distribution with parameter 1, and getting random numbers from an exponential distribution with (rate) parameter λ (its inverse mean) only requires division by this λ).

The reaction is determined by roulette-wheel selection, i.e. finding i such that

$$\sum_{\alpha=1}^{i-1} \frac{a_\alpha}{a_0} \leq r_2 \leq \sum_{\alpha=1}^{i} \frac{a_\alpha}{a_0} \tag{2.10}$$

for another uniformly generated random number $r_2 \in (0, 1]$ (or, equivalently, finding the smallest i that fulfills the right part of the inequality).

First Reaction Method

In the First Reaction method (FRM) (Gillespie 1976), next reaction times $\tau_\alpha = -\frac{1}{a_\alpha} \ln r_\alpha$ are calculated for each reaction independently, requiring generation of m (pseudo-)random numbers. Then the time τ and index i of the next executed reaction are

$$\begin{aligned} \tau &= \min_{\alpha \in 1..m} \tau_\alpha \\ i &= \operatorname*{argmin}_{\alpha \in 1..m} \tau_\alpha \text{ (i. e. } \alpha \text{ for which } \tau_\alpha = \tau). \end{aligned}$$

The result is equivalent to that of the Direct method since for exponentially distributed variables $X_1 \sim Exp(\lambda_1)$ and $X_2 \sim Exp(\lambda_2)$, $\min(X_1, X_2) \sim Exp(\lambda_1 + \lambda_2)$ and hence

$$Exp(a_0) \sim \min_{\alpha \in 1..m} \{Exp(a_\alpha)\}.$$

Compared to the Direct method, more random numbers are generated. The method can be advantageous, however, when reaction frequencies differ by orders of magnitudes. Then, the discrete resolution of floating point numbers in computers may lead to rounding when calculating $\sum \frac{a_\alpha}{a_0}$ and erroneous omissions of slow reactions with a higher index than fast reactions.

Next Reaction Method

In Gillespie's Direct and First Reaction method, each step's time complexity is in the order of number of reactions (all a_i have to be calculated each time).

In Next Reaction method (NRM) by Gibson and Bruck (2000), the First Reaction method is optimized by storing the initially calculated a_α and recalculating, in each step, only those affected by the last step's reaction. For this, it uses a *dependency graph* of which reaction's stoichiometry vector affects which reaction's reactants.

To find the next reaction, it uses an *indexed priority queue* data structure (subsequently also: event queue) that allows retrieval of the minimum element in $O(\ln m)$ instead of $O(m)$ in a simple loop (where m is the number of entries, i. e. of reactions here).

Further optimizations include re-use of random numbers by re-scaling τ_α of affected reactions such that $\tau_{\alpha,new} = \frac{a_{\alpha,old}}{a_{\alpha,new}}(\tau_{\alpha,old} - t) + t$ (except for the reaction actually executed last), where the τ's use an absolute time base, i. e. are relative to the beginning of the simulation. Then, only one (pseudo-)random number r is needed in each step, for recalculating the next firing time of the last executed reaction: $\tau_i = -\frac{1}{a_i} \ln r + t$.

Approximate Solutions: Tau-Leaping

A high copy number of certain species and a high reaction rate constant may lead to one reaction being executed many times in a row before any other reaction happens. It may then be tempting to calculate the number of times this reaction happens, or in fact any group of non-dependent reactions (in the sense of the NRMs dependency graph). However, each single state change may affect rates of other reactions. The basic idea behind *approximate methods* (as opposed to the above, which are *exact* realizations of the CME) is to determine when the state changes due to multiple reactions executed in one step result in only infinitesimal changes in the reaction propensities a_i such that these changes can be ignored. This is usually called the *leap condition*, and the τ-leaping method (Gillespie 2001) is concerned with finding the largest time interval τ at each step for which this condition is fulfilled given a parameter ϵ that quantifies when a propensity change should be considered "effectively infinitesimal". When $\boldsymbol{\lambda}$ is the state change resulting from the reactions in interval τ (not to be confused with the – here irrelevant – exponential distribution parameter usually denoted λ) and a_0 is the sum of reaction parameters a_i again (Equation 2.8), the parameter ϵ's influence is described by

$$|a_i(\mathbf{x} + \boldsymbol{\lambda}) - a_i(\mathbf{x})| \leq \epsilon a_o(\mathbf{x}) \text{ for each } i \in 1...m.$$

Other approximate variants include k_α-leaping (Gillespie 2001), which, instead of a time step τ, determines the maximum number of times each reaction can fire before the leap condition is violated, implicit τ-leaping targeting stiff systems (Rathinam et al. 2003), the slow-scale SSA (Cao, Gillespie, and Petzold 2005), R-leaping (Auger, Chatelain, and Koumoutsakos 2006) and L-leap (Peng and Wang 2007). All these approximate methods have at least one parameter determining the granularity of the approximation, where the aformentioned exact methods are parameter-free.

Further Optimizations

Many implementations of the SSA with optimizations for the general or for special cases exist. In the Optimized Direct method (ODM; Cao, Li, and Petzold 2004), for example, a reaction-dependency graph is used, too, and the model is *pre-simulated* to find frequent reactions. This information is then used to order the reaction such that the computational costs of recalculating the propensity sum can be reduced, especially in multi-scale systems (i.e. with widely differing reaction rates). The Sorting Direct method (McCollum et al. 2006) follows a similar idea but eliminates the pre-simulation required by the ODM by sorting on-the-fly. The Logarithmic Direct method (Li and Petzold 2006) employs binary search on the list of reactions such that the complexity of finding the next reaction is logarithmic in the number of reactions – $O(\ln m)$ – independent of any sorting. As a final example, the Partial-propensity Direct method (Ramaswamy, González-Segredo, and Sbalzarini 2009; Ramaswamy and Sbalzarini 2010) is an optimization whose computational costs scale with the number of species, not the number of reactions, and thus is appropriate for large-scale networks with relatively few species.

2.3 Simulation: Adding Space

The aforementioned methods are all based on the assumption that the system is well-mixed, i. e. that for each molecule the probability of encountering a reaction partner is the same as for all other molecules of the same species. When this is not a reasonable assumption, space must be incorporated into the model. Different approaches of varying complexity, both in terms of modeling and computational execution, exist, which are now briefly introduced (mostly taken from our previous work Bittig and Uhrmacher (2010), incl. Figure 2.2).

2.3.1 Compartments

If one species has different reaction partners in different compartments of a cell, it is necessary to know its quantity in each compartment for proper representation of its reaction dynamics. One can then use a non-spatial modeling approach for the dynamics inside each compartment and add additional terms for the transfer of species between them.

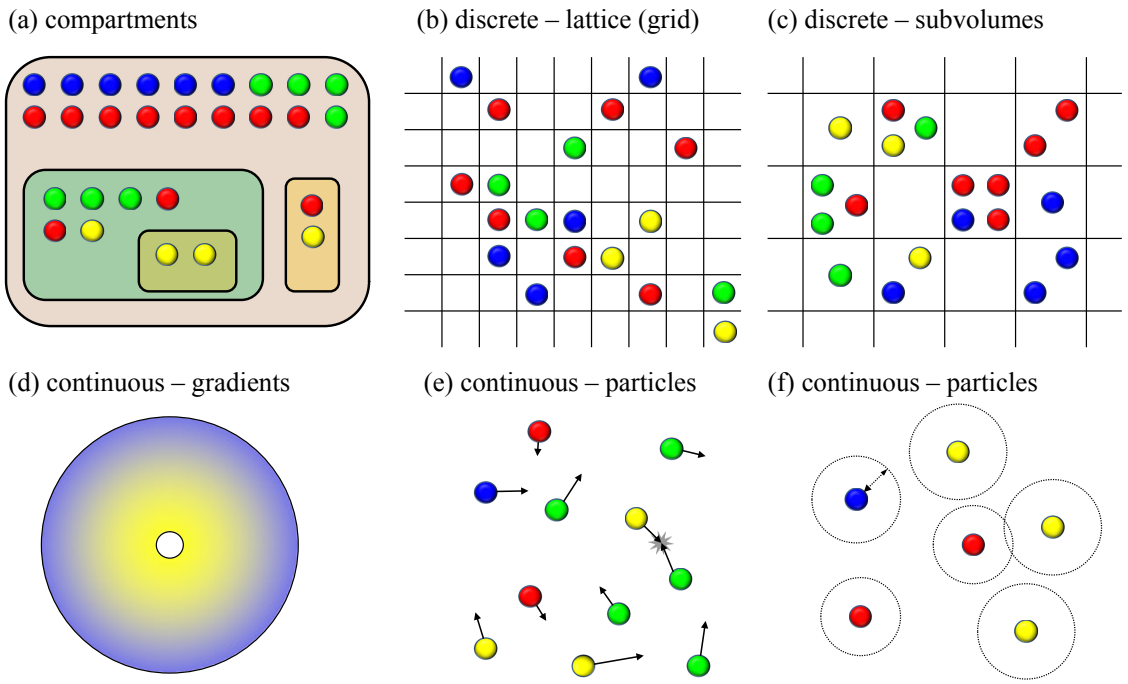


Figure 2.2. Different ways of representing space in a model or a simulation. (a) molecules/species can be in different compartments and interactions happen only within one compartment, apart from transport between compartments that has to be modeled explicitly; (b/c) space is divided into equal blocks of a defined neighborhood relation, usually a lattice of cubes, either (b) small cubes occupied by no more than one particle or (c) subvolumes taking several (up to arbitrary many); (d) model describes not particles, but concentrations and their gradients across space via appropriate boundary conditions; (e/f) particles are associated with position and motion in space, collisions between particles trigger reactions. Individiual-based, continuous-space motion may be modeled following the physical interaction between moving bodies (Molecular Dynamics) or by a fine sampling of random trajectories exhibited by larger particles colliding with surrounding solvent molecules (Brownian Dynamics) (e). In non-crowded environments, position changes can be sampled more coarsely such that particles cannot, in each time step, move beyond a sphere of influence chosen such that each particle can interact with at most one other, lowering computational costs (GFRD) (f).

This most basic, qualitative approach to incorporating (discrete) spatial features does not need to represent actual molecule positions. Compartments may be nested, i. e. each compartment may contain one or more sub-compartments, as depicted in Figure 2.2 a. However, the simple distinction of, for example, nucleus and cytosol would also qualify here. Typically, the compartmental approach of representing space is used at the level of pathways and interaction networks up to cells and tissues. They are less common for molecules and protein complexes, although examples do exist (Maus 2008).

2.3.2 Continuous Space: Particles

For most realistic modeling of molecular motions in space, each particle is represented individually using rules arising from fundamental physics with coordinates and a trajectory in a continuous space. A many-body problem with only computationally expensive solutions or approximations arises. The most fundamental approach, *molecular dynamics* (MD; Adcock and McCammon 2006, Figure 2.2 e), tackles this by numerically solving Newton's laws, which is feasible only for very small scales both in time and space, as computation effort required is proportional to the number of possible particle interactions. *Dissipative particle dynamics* (DPD) is a coarse-grained approximation of MD (Heyes et al. 2004), but does not permit biochemical interactions and is thus not suitable for simulation of biochemical systems.

A non-deterministic approach, i. e. one representing random noise in particle movements, is that of *Brownian dynamics*, relying on the Langevin equation (Ermak and McCammon 1978; Northrup, Allison, and McCammon 1984). Like in molecular dynamics, a continuous differential equation system has to be solved numerically, and it is also computationally expensive, especially the more crowded the environment is.

Brownian motion of particles moved by collisions with (much smaller molecules of) a solvent can be approximated by displacing particles in time steps Δt by $\sqrt{2D\Delta t}\xi$, where ξ is a vector of normally distributed random numbers with mean 0 and variance 1 (also called Smoluchowski dynamics; Andrews and Bray 2004). The diffusion constant is derived from the Einstein relation $D = k_B T/\gamma$, where T is the absolute temperature, k_B the Boltzmann constant and γ is a (species-dependent) friction coefficient, alternatively given as mobility $\mu = \frac{1}{\gamma}$.

Assuming that, when choosing time steps of sufficiently small size, no more than two molecules react at the same time, the many-body problem can be reduced to two-body problems, for which an analytical solution exists. *Green's function reaction dynamics* (GFRD; van Zon and ten Wolde 2005b, Figure 2.2 f) thus approximates the solution of Brownian dynamics, turning it into a discrete event type problem. These computations are faster especially when distances between molecules are not too small (allowing larger time steps).

2.3.3 Continuous Space: Gradients

Whereas ordinary differential equations describe time-dependency of one variable (here usually species concentration) on other variables of the same type, partial differential equations (PDEs) allow description of dependency on more than one variable, e. g., spatial position besides time. However, individual particles are not represented here, only concentration gradients across space. This has been applied to microbiological phenomena like the behavior of receptors on a membrane (Haugh 2002), but also to whole cells (Kholodenko 2006). Numerical integration of the PDEs will focus on approximating the solution for a discrete number of points in the usually continuous domain. This is known as *mesh generation* – the finer the mesh (Slepchenko et al. 2003), the more accurate the approximation, but the higher the computational costs. From the modelers point of view, this approach thus offers a continuous representation of space (as depicted in Figure 2.2 d), but the simulation traditionally treats space as an irregular grid generated on the fly (the mesh; cf. Takahashi, Arjunan, and Tomita 2005, fig. 1 e).

PDE-based simulations are deterministic and thus do not cover stochastic effects from random noise. Noise terms could be added, yielding stochastic partial differential equations (SPDEs), whose applications so far, however, seem to lie outside the realm of biological simulation.

2.3.4 Discrete Space: Single-occupancy Lattice

Whereas the previous approaches represent spatial aspects by real-valued coordinates, space can also be discretized by splitting it into regions considered separately, i. e. projecting a lattice onto it with certain assumptions about each lattice element. *Cellular automata* (Gardner 1970) are the most prominent representative of this group. The lattice here consists of appropriately small uniform blocks, each of which may be occupied by one or no model entity (Figure 2.2 b). These entities may be proteins (Kier et al. 1996) for intra-cellular processes, cells in a tissue or organ (Moreira and Deutsch 2002; Deutsch and Dormann 2004) up to individual organisms in population dynamics (Dewdney 1984).

Cellular automata of the Lattice Gas CA variety usually employed in such simulation takes place in (usually equidistant) discrete time steps. Movement of particles is realized as propagation to a nearby block in the lattice (the original definition is about living and dying at each grid cell in response to the neighbors' state, not movement). Thus one block's state depends only on the previous state of its neighbors and itself. When two particles are to move into the same block, special treatment of a collision or reaction type event is triggered.

These approaches have the potential to drastically reduce computational complexity compared to continuous-space individual-based simulation, but require the block size to reflect average entity distance and average entity diameter. It moves small fractions of space itself into the focus of interest, i. e. their occupancy or non-occupancy by a key player of cellular dynamics. This micro-scale approach thus does not cope well with differences in molecule numbers and sizes (i. e., multiple scales), although it can be

extended or incorporated into methods that do so, for example by adding smaller entities of which many can be located in one grid cell (Dynamic CA; Wishart et al. 2005) or larger entities occupying multiple grid cells and potentially encompassing grid-cell-sized molecules (Haack et al. 2013), or by allowing first-order reaction events at random times according to the SSA/Gillespie's algorithm (Spatiocyte; Arjunan and Tomita 2010).

A further extension, the cellular Potts model, considers (biological) cells or tissue components that occupy multiple neighboring lattice sites and may grow into free adjacent sites or push other cells/tissue out of them (Alber et al. 2002).

2.3.5 Discrete Space: Subvolumes (Spatial Gillespie, RDME)

The aforementioned problem of coping with varying numbers can be overcome by using a more coarse-grained lattice and allowing several molecules in each lattice element, i. e. in a subvolume (Figure 2.2 c). Differing spatial dimensions of species are still not covered, as molecules are considered to be dimensionless entities.

This can be implemented by appropriate adaptations of cellular automata (Shimizu, Aksenov, and Bray 2003), or by extending the SSA spatially, considering the molecules to be distributed uniformly in each lattice cell and letting them react according to Gillespie's algorithm, but also diffuse into neighboring subvolumes with diffusion event times exponentially distributed like the reaction events. The behavior of such simulations, also known as Spatial Gillespie, are thus governed by the reaction-diffusion master equation (RDME) (Stundzia and Lumsden 1996), an extension of the CME (Equation 2.5) to a state set $\mathbf{X} = \{X_j, \mathbf{r}\}$ that contains species (indexed by j) in subvolumes (indexed by $\mathbf{r}$) (Gardiner et al. 1976; Nicolis and Prigogine 1977; Baras and Malek Mansour 1996), although special attention must be paid in the limit of small subvolumes for the method to exhibit consistent microscopic behavior (Fange et al. 2010). A popular example of an RDME approach is the Next Subvolume method (Elf, Doncic, and Ehrenberg 2003; Elf and Ehrenberg 2004).

These major approaches to spatial simulation offer varying granularity in the approximation of the physical processes. From the computationally very expensive molecular dynamics to the rather elementary discrete grid of cellular automata that allows much faster simulation, the main trade-off is between accuracy and calculation time, making each approach more or less fitting for different types of phenomena. The PDE approach sticks out as it does not cover individual particles but rather distribution gradients, the purely compartmental approach deviates from the others in that it does not describe motion in the continuous space or a discretization thereof, but only movements between a finite number of compartments. The applicability of approaches depending on model properties is illustrated in Figure 2.3.

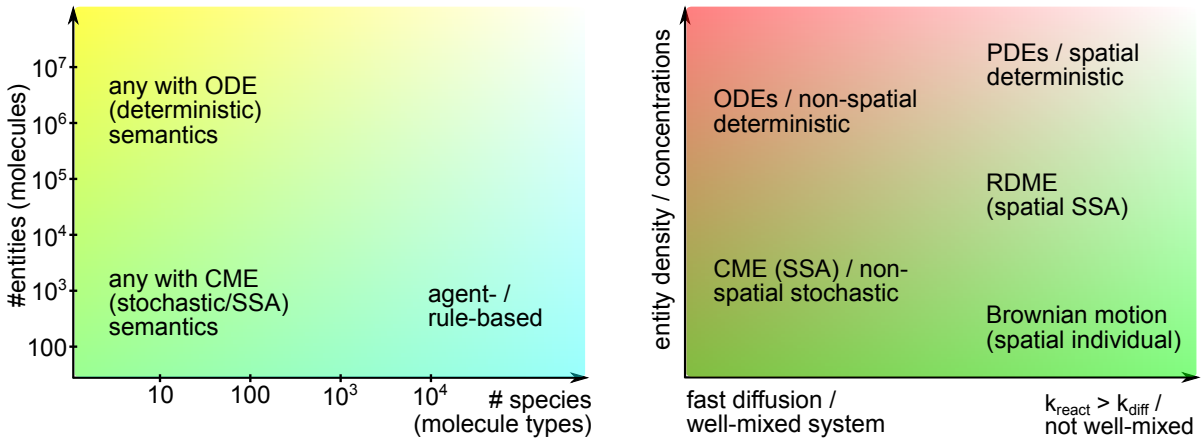


Figure 2.3. Applicability of modeling and simulation approaches in relation to system properties.

2.3.6 Reaction Rates in Well-mixed Systems and in Space

Reaction Kinetics

The speed of a chemical reaction in a well-mixed environment is proportional to the amount of each available reactant (Wilhelmy 1850). The proportionality constant is usually called *rate constant*, as the reaction rate is then simply the product of this constant and the concentration of each reactant raised to their stoichiometric coefficient's power if several of the same reactant are needed (e. g., for a reaction $A + 2\,B \rightarrow \ldots$ and rate constant k, the actual rate would be $k[A][B]^2$).[1] For this product to be a dimensionless number, the units of the rate constants differ between reactions of different *order*, i. e. different number of required reactants.

These simple reaction kinetics, known as *mass action*, can also be applied to intracellular reactions (and form the basis of the examples given previously in section 2.2; compare, for example, the ODE terms in Equation 2.3). However, abstractions and hidden assumptions in cell biology may require more complex descriptions. For example, consider a protein-catalyzed reaction where the enzyme binds reversibly to the substrate, $E+S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_3} E+P$ (which is a condensed version of Equation 2.2). If the total amount of enzyme is constant and so is the amount (or concentration) of the enzyme-substate complex (*quasi steady-state assumption*), the overall rate of product formation can be described by $v = V_{max}[S]/(K_M + [S])$ where $V_{max} = k_3[E_{tot}] = k_3([E] + [ES])$ is the *maximum rate* and $K_M = (k_1 + k_3)/k_2$ the *Michaelis constant*. These *Michaelis-Menten*

[1] Note that in stochastic simulation, particle numbers have to be converted to concentrations first as this product encompasses the likelihood of particles to be close enough for reaction, which is volume-dependent. Note also that when two reactants of the same species are required, one would usually include $\binom{\#B}{2}V^{-2} = \frac{1}{2}\frac{\#B}{V}\frac{\#B-1}{V}$, i. e. the number of possibly reacting pairs as factor instead of $[B]^2$, so the rate constant has to be adjusted by a factor of two for similar behavior.

kinetics, one of several enzyme kinetic approaches, apply to a single reaction with two parameters that subsume three mass-action reactions and four parameters (three rates and total enzyme concentration).

Other reasons to use different kinetics may include the desire to fit a model to experimental data where a detailed mechanism is unknown, or where one suspects crowding effects to play a role (mass action kinetics require the assumption of free diffusion). *Power-law models* (Vera et al. 2007) are an extension of mass-action kinetics designed for this, where in the rate product $k \prod_{j=1}^{n} [S_j]^{g_j}$ the exponents g_j need not correspond to the (integer) stoichiometric coefficients of the reactants, but may instead be real numbers (usually in the range of the common stoichiometric coefficients $0 \ldots 2$, but in general power-law models, negative values are allowed and used to cover inhibitory effects).

Macroscopic and Microscopic Rates

For reactions involving multiple reactants, the actual rate depends on, and the rate constants used above comprise, the likelihood of reactants to get into sufficient proximity, which is determined by their movement and the available volume, and their likelihood to react when close enough. In individual-based *spatial* simulation, movement is an explicit part of the simulation and if reasonable estimates of rate constants in well-mixed conditions are known (*macroscopic rates*), they need to be related to reaction parameters needed for spatial simulation.

The maximum possible reaction rate constant for particles of two species A and B with radius r_A and r_B diffusing randomly with diffusion constants D_A and D_B, respectively, is $k_D = 4\pi(r_A+r_B)(D_A+D_B)$ (von Smoluchowski 1917). When regarding particles as hard spheres, one may be tempted to use the ratio $\frac{k_a}{k_D}$ of the macroscopic (association) rate and this *diffusion-limited rate constant* as probability of a reaction in case of a collision. However, the actually observed rate may deviate from the desired rate depending on crowding and position update step size as a non-reactive collision will usually leave particles closer than average and thus more likely to react subsequently (Noyes 1956). The macroscopic rate relates to the diffusion-limited rate according to $\frac{1}{k_a} = \frac{1}{k_D} + \frac{1}{k_{micro}}$, or $k_a = \frac{k_D k_{micro}}{k_D + k_{micro}}$ (Collins and Kimball 1949) where the microscopic rate k_{micro} leads to the probability of an A and B less than $r_A + r_B$ apart reacting in a time step Δt: $P = \frac{3 k_{micro} \Delta t}{4\pi(r_A+r_B)^3}$. This approach relies on point-based particles that *can* be closer together than the sum of their radii (e. g., Klann and Koeppl 2012) and tracking of the time particles spend close to each other, or using sufficiently short *fixed* time steps Δt (such that $P \ll 1$).

An alternative is to ignore particles' individual radii entirely and only consider interaction radii, i. e. the choosing a reaction-specific value in place of $r_A + r_B$ such that when two respective particles (represented by their center points) are closer than this value, the reaction is executed directly as in Smoldyn (Andrews and Bray 2004; Andrews et al. 2015) or with a some probability. Effects of step-wise sampling of a Brownian trajectory are relevant as "collisions" (i. e. reactive proximity) may be missed with larger steps.

Listing 2.1. SBML reprentation of a single reaction of the predator-prey model

```
<reaction id="predation" reversible="false">
  <listOfReactants>
    <speciesReference species="Prey" stoichiometry="1"/>
    <speciesReference species="Predator" stoichiometry="1"/>
  </listOfReactants>
  <listOfProducts>
    <speciesReference species="Predator" stoichiometry="2"/>
  </listOfProducts>
  <kineticLaw>
    <math xmlns="http://www.w3.org/1998/Math/MathML"> <apply>
      <times/> <ci> Prey </ci> <ci> Predator </ci> <ci> delta </ci>
    </apply></math>
    <listOfParameters>
      <parameter id="delta" name="delta" value="0.02" units="substance"/>
    </listOfParameters>
  </kineticLaw>
</reaction>
```

2.4 Modeling Formalisms and Languages

Orthogonal to the problem of how to represent a system's properties in a model is the question of how to describe or specify the model itself. Simply coding the model's behavior (and thus, model and simulator) in a computer programming language is not uncommon. However, having a formal model description independent of the simulator is preferable as it can improve understandability of the model by separating this concern from the computer implementation and also aid debugging (by allowing an easier distinction whether unexpected model behavior is due to bad model assumptions or faulty software implementation). Some modeling formalisms also aid formal reasoning about a model's behavior, e. g., state reachability analysis.

Standardized model description formats are also important for model reuse, composition and exchange between tools. For non-spatial simulation, the Systems Biology Markup Language (SBML) has become the de facto standard for storing and exchanging models targeted at both deterministic and stochastic simulation. It is not designed to be human-readable and thus not suitable for model specification (see Listing 2.1), which is done in many popular tools via a graphical user interface (e. g., COPASI – Hoops et al. 2006 – or CellDesigner –Funahashi et al. 2007; Funahashi et al. 2008). However, a "shorthand" notation exists, where the description resembles some rule-based approaches. Capabilities for spatial or multi-level models are limited, only fixed compartments are supported (although drafts exists for "packages", i. e. extensions of the SBML "core", that would add multi-state and multi-compartment species, and spatial processes). Establishing a standard for multi-level, multi-scale modeling is still an open

challenge (de Back et al. 2015).

The following sections contain a brief overview of selected model description approaches.

2.4.1 Differential Equations

With differential equations, there is no clear separation between formalism and interpretation, as the term comprises both a well-defined way to specify the equations (syntax) and their mathematical meaning (semantics). While it may be argued that differential equations describe a model in the *language of mathematics*, the actual input format for the respective tools and the visual presentation can differ widely (hence the rise of SBML, for example). Since abstractions regarding the reaction dynamics have to be made when modeling on the population level, the reaction kinetics, e. g., mass action, Michaelis-Menten or detailed enzyme kinetics, are thus incorporated into the equations' structure, i. e. the formal part.

There are other modeling formalisms with ODE semantics, i. e. while the model is specified in some other manner, the respective simulation amounts to ODE integration again. Sometimes these semantics are the results of adaptation of an existing modeling formalism to the continuous realm, e. g., continuous petri nets (Recalde, Haddad, and Silva 2007), sometimes they are just alternatives and approximations of other, often stochastic semantics, while some were created specifically to allow easier or more general description of reaction rules.

2.4.2 Process Algebras

In computer science, process calculi have long been used for modeling concurrent systems formally. They provide syntactic constructs to describe processes, usually computations performed on some device, and algebraic rules (hence the synonymous term process algebras) allowing formal reasoning about the descriptions. For more than a decade they are applied to biological systems. A few variants include some form of spatial representation, e. g., the ambient calculus for modeling concurrent processes on mobile devices organized in a network of variable topology (Cardelli and Gordon 2000). Others have been adapted to take spatial properties or other special features of biological systems into account.

The approach is often termed *molecules-as-computation*, where model entities are individual particles or species described in terms of their interaction capabilities. While the origin of process algebras is the analysis of concurrent processes (e. g., running on a computer) communicating with each other, reacting particles can also be described (Regev and Shapiro 2004). The π-calculus (Milner 1999) and specifically its stochastic extension (Priami 1995) serve as the basis for many.

Particles of one species are considered processes of the same kind running in parallel (an initial configuration of processes, i. e. initial amounts of each species, has to be

given). Reactions consist, broadly speaking, of processes sending or receiving messages via dedicated channels, indicating, respectively, willingness to react or availability of a reaction partner, and continuing as certain different processes after successful communication (i. e. a reaction).

Spatial extensions of these process algebras were mostly devised with compartments in mind, only few deal with actual coordinates assigned to processes. Process algebras that cover compartmentalization, nested or not, but no further spatial attributes, and thus assume homogeneous distribution of species within a compartment, include BioAmbients (Regev et al. 2004), Beta Binders (Priami and Quaglia 2005), Brane Calculi (Cardelli 2005), and others (John, Lhoussaine, and Niehren 2009).

Bio-PEPA (Ciocchetta and Hillston 2008) is a process-algebra based modeling language for which a variant with explicit compartments is also available (Ciocchetta and Guerriero 2009). It follows, in its authors' words, a "*processes as species* and not *processes as molecules*" abstraction, i. e. the model entities are abstractions of molecule species and the model describes changes in the amount of each species. Reactions are still not represented explicitly, only by shared channel names. Aside from stochastic simulation it is also possible to convert species amounts to concentrations and derive ODEs approximating the stochastic solutions (Galpin 2010).

The attributed π calculus $\pi(\mathcal{L})$ with priorities (John et al. 2010) also allows definition of an arbitrarily large (but finite) number of parameterized compartments. The parameters allow establishing a neighborhood relation and facilitate definition of a function for movement between neighbors, although these attributes can be of any type and at the simulation level it is not known whether they actually represent coordinates. In addition to the *processes as molecules* view common with the π calculus, here a *processes as reactions* approach is also possible, i. e. processes change attributes representing species amounts (or concentrations) in line with the actual reactions.

Neither of these approaches allows representation of continuous space. In Space-π (also: SpacePi; John, Ewald, and Uhrmacher 2008), processes are equipped with actual coordinates in a two- or three-dimensional space, direction (i. e. a movement function) and radii to indicate the range of their communication potential. Processes are still able to send and receive messages, but only within the given radius instead of the global space or the whole compartment, ensuring that only entities close to each other can react. The formalism thus has hybrid semantics, combining continuous motion of particles with discrete events like collisions and reactions. While the original description of SpacePi covers only rather simple forms of stochastic and directed motion, Brownian motion can also be covered (Haack, Leye, and Uhrmacher 2010).

Whereas entities in Space-π are circular or spherical, Shape Calculus (Bartocci et al. 2010a, b) was a proposed formal approach with molecules with a more complex 3D structure in mind (which did not reach maturity in form of a usable simulator). The modeled entities are 3D shapes moving in space, communication channels represent potential binding sites, and while a binding reaction yields a new process like in the other calculi, the new process' name is not an abstract one but still contains the original components

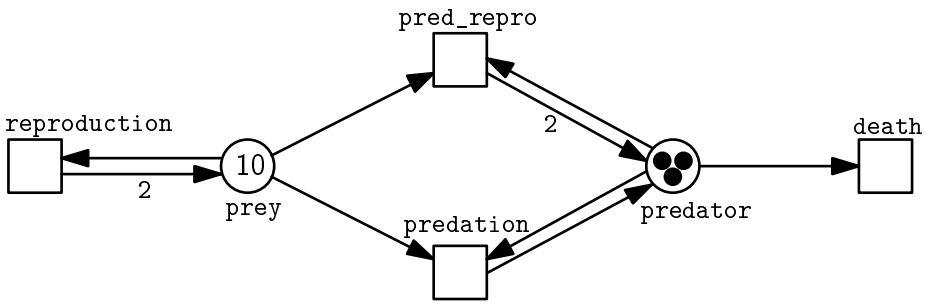


Figure 2.4. Petri net representation of a Lotka-Volterra-Model. Places/states are round, transitions/reactions rectangular. Marking (i. e. tokens) on places indicated by numbers or small filled circles. The transitions `predation` and `pred_repro` differ only in the weight of their outgoing edge, i. e. the number of tokens produced (which is 1 if not explicitly given; cf. R_{2a} and R_{2b} of Equation 2.7).

and the binding site they are linked with. Movement of shapes necessitates the application of sophisticated (and computationally expensive) collision detection techniques as investigated in the field of computer graphics.

2.4.3 Rule-based Languages

Rule-based modeling languages do not focus on the species themselves, but rather the interactions and conversions between them given by explicit rules (Hlavacek et al. 2006). Many, e. g., κ ("Kappa", also known as the κ-calculus; Danos et al. 2007; Danos et al. 2009) and BioNetGen (Blinov et al. 2004; Faeder, Blinov, and Hlavacek 2009), were designed to address combinatorial explosion associated with *multi-state species*, i. e. many different variants of a protein being able to participate in reactions with another protein of which also many different subtypes exist, e. g., MAP-kinases and MAPKKs, and deal with this by establishing generic and concrete agents and rules inspired by object-oriented programming. Approaches that allow representation of at least compartmental space include cBNGL (Harris, Hogg, and Faeder 2009), the Language for Biological Systems (LBS; Pedersen and Plotkin 2008; Pedersen and Plotkin 2010) and ML-Rules (Maus, Rybacki, and Uhrmacher 2011; Maus 2013).

Rules may be represented in text-based form (e. g., equations 2.2 and 2.7 in section 2.1) or in a graph-based format (Faeder, Blinov, and Hlavacek 2009). Details of the above approaches will be discussed in the next chapter since the key approach developed for this thesis is also rule-based.

2.4.4 Petri Nets

Petri nets are an approach that originally gained popularity in computer science and was later adapted for biological system (although they were arguably invented for represent-

ing chemical reactions; Petri and Reisig 2008). Formally, a petri net consists of *places* and *transitions* with directed edges between them (Figure 2.4). Places may be occupied by a number of *tokens*, and transitions *fire* by consuming tokens from all places with incoming edges to the respective transition and producing tokens on places connected by outgoing edges. One of the first applications of Petri nets was modeling concurrent processes on computers. Starting from there, a variety of methods for formal analysis was developed, e. g., for exploring which states were reachable given an initial *marking*, i. e. an assignment of token amounts to the places.

There is an intuitive correspondence of species and reactions as used in reaction networks to places and transitions, so Petri nets are also used for simulation of biological systems, especially for exploring the structure of larger networks (Sackmann, Heiner, and Koch 2006; Heiner, Gilbert, and Donaldson 2008).

The exact semantics of a Petri net model depend on the variant. There are continuous Petri nets, where markings are interpreted as continuous values and transitions "fire" continuously, consuming and producing according to an associated rate, which essentially means ODE semantics, and stochastic Petri nets, where transition firing order is determined by expontially distributed random numbers also depending on transition rate parameters, leading to SSA semantics.

With colored nets (Jensen 1981; Gilbert et al. 2013), tokens are associated with distinguishing properties and reactions may fire only for certain "color" combinations of tokens, essentially introducing the features of an attributed formalism to Petri nets.

2.4.5 Others

Cellular Automata (CA) are a discrete, step-wise modeling approach based on a large lattice (grid) of identical cells. A cell and its neighbors current state determines the cell's next state. In one of the basic and most well-known examples, Conway's Game of Life, there are only two possible states (*live* and *dead*) and very simple rules, however, complex patterns can result (Gardner 1970). They have been used to describe diverse biological phenomena (Materi and Wishart 2007), for example liquid solutions containing enzymes and substrates exhibiting known enzyme kinetics in the simulation (e. g., Michaelis-Menten; Kier et al. 1996) or formation of patterns (Deutsch and Dormann 2004), e. g., in tumor growth (Moreira and Deutsch 2002).

For simulation of intracellular dynamics, a variant called *Dynamic Cellular Automata* (DCA) has been developed (Wishart et al. 2005), taking in aspects of agent-based modeling. An explicit notion of time is added. Choosing an appropriately small lattice element size close to that of an average protein and a time step size so small that molecules do not travel farther than one block at a time, macromolecules (e. g., proteins and RNA) are indeed modeled spatially akin to the original Cellular Automata. DCA also allow small molecules, of which several can be located inside one cell (unlike macromolecules), and non-mobile "supermolecules" to model membranes (and DNA, interestingly). With these and other extensions of cellular automata (e. g., Haack et al. 2013), however, one

again leaves the territory of formal models and gets close to ad-hoc implementations in the authors' programming language of choice.

The discrete event systems specification *DEVS* is another modeling approach used for hierarchical (Degenring, Röhl, and Uhrmacher 2004; Uhrmacher et al. 2007; Maus et al. 2008) and even for spatial simulation of biological systems (Goldstein and Wainer 2009). However, in the later case the laws of continuous movement as well as the model itself are specified as DEVS and executed by the discrete event simulation machine.

Similarly to DEVS, *statecharts* (Harel 1987), whose origin lies in software engineering, support a hierarchical modeling of cell biological systems (Holcombe and Bell 1998; Kam, Cohen, and Harel 2002), but in a more visual way. They have even been applied to cover basic spatial phenomena involving the fate of adjacent cells and the interplay with gene regulation within them (Fisher et al. 2005). Like DEVS, they, too, do not support an easy modeling of biochemical reactions. *Biocharts* (Kugler, Larjo, and Harel 2010) are a visual framework for representing "what a biological system does (specification) how it does it (mechanism) and systematically compare to data characterizing system behavior (experiments)" (Kugler 2013).

Formalisms especially suitable for hierarchical modeling are also discussed in more detail in Maus (2013, ch. 5).

2.4.6 Spatial Formalisms

Formalisms with numeric attributes, e. g., some rule-based approaches like ML-Rules, the aforementioned Attributed π, or colored Petri nets, allow explicit encoding of spatial positions as an attribute, e. g., a subvolume index for RDME-based simulation. Spatial simulation is then possible by two adaptations. First, adding an appropriate condition concerning this attribute to rules, communications, or transitions affecting more than one entity, i. e. requiring entities to be in the same subvolume. Second, adding movement, i. e. first-order changes to the spatial attribute that reflect a changing position, e. g., diffusion into neighboring subvolumes. This way, the models need to be extended quite a bit for the addition of the relatively intuitive concepts of "spatial proximity" and "movement", making them inelegant and rather verbose.

In discrete event based formalisms with CTMC semantics, using attributes for spatial properties does not fit well with continuous-space coordinates and Brownian motion (i. e. random diffusion where the distance traveled is proportional to the square root of time passed) or directed movement (where a constant speed is difficult to express in a framework of event intervals with exponentially distributed time).

Table 2.1 provides a brief overview of targeted approaches, or adaptations of other approaches, with capabilities for expressing space in a less verbose manner. There are also approaches that use a formalism, mostly a rule-based language, for reactions, and their own format for spatial properties like system or particle geometry (e. g., Smoldyn; Andrews 2016 ch. 11, or SRSim; Gruenert et al. 2010).

Similar overviews with a focus on spatial *simulation* tools can be found in Arjunan

Table 2.1. Modeling formalisms and their capabilities regarding spatial modeling.

modeling formalism	model description format	explicit compartments	dynamic cell structures	complex geometries	space representation	temporal evolution
Cellular Automata	(custom)	-	-	-	lattice[1]	fixed time steps
Bio-PEPA (Ciocchetta and Hillston 2009)	calculus	o	-	-	(qualitative)	discr. ev. (CTMC)/ cont. (ODEs)
Space π (John, Ewald, and Uhrmacher 2008)	calculus	-	-	-	continuous	continuous (step-wise approx.)
cBNGL (Harris, Hogg, and Faeder 2009)	rule-based	o	-	-	(qualitative)	discrete event (CTMC)
spatial κ (Sorokina et al. 2013)	rule-based	$*$	-	o	lattice	discrete event (CTMC)
CSMMR (Oury and Plotkin 2013)	rule-based	o	o	-	(qualitative*)	discrete event (CTMC)
ML-Rules (Maus, Rybacki, and Uhrmacher 2011)	rule-based	o	o	-	(qualitative*)	discrete event (CTMC)
ML-Space (this work)	rule-based	o	o	-	lattice & cont.	discr. ev. (CTMC)+ cont. (step-wise a.)
PDEs	equations	-	-	-	continuous/mesh	continuous (PDEs)

Entries: o: explicitly supported by formalism. $*$: supported with limitations, e. g., by explicitly describing relevant structures in the model. -: not supported.
Columns: *dynamic cell structures*: ability to change/create compartments at run time, e. g., for describing a virus docking to a cell, releasing its content into the host cell. *complex geometries*: supports shapes other than circles/spheres or rectangles/cuboids. *space representation*: "(qualitative)" denotes compartments-only approaches (with homogeneous distributions inside these), * here marks where lattice structures can be explicitly encoded via attributes in the model; lattice[1] denotes single-occupancy lattice, otherwise lattice cells contain populations of particles.

and Tomita (2010, Table 1) or Burrage et al. (2011, Table 2.1).

3 ML-Space: A Rule-based Modeling Language in Space

In this chapter, basic concepts for expressing spatial properties in rule-based languages, including ML-Space, will be introduced step-by-step first, followed by a more thorough look at ML-Space' syntax and semantics as far as the rules are concerned.

An algorithmic (pseudocode-based) format will be used rather than formal (operational) semantics, as the spatial constraints would be difficult to incorporate and result in a dense complex description not very suitable for an introduction to the method. ML-Space will support subvolume-(RDME-) and individual-based simulation, requiring distinction of entities based on what type they are and where they are, making the rule application algorithms alone necessarily more complex than those of non-spatial languages (cf. Maus 2013, fig. 6.4 for ML-Rules's basic algorithm in pseudocode or Warnke, Helms, and Uhrmacher 2015 for formal semantics).

The simulation's spatial aspects themselves are left for the next chapter.

3.1 Introducing Attributes

$\texttt{C + O}_2 \rightarrow \texttt{CO}_2$ is a common way to describe one common chemical reaction. It can be interpreted as describing the conversion of two individual molecules – a carbon atom C and an oxygen molecule $\texttt{O}_2$ identified by the atoms it consists of – into another molecule – carbon dioxide $\texttt{CO}_2$. It can also be viewed as postulating the ability of any carbon atom to react, together with some oxygen molecule, to carbon dioxide. The distinction between these interpretations – a transformation of concrete instances of molecules, or the reaction pattern that any molecules of a given type can follow – is of little relevance in basic inorganic chemistry, where molecules occur in bulk and distinction of individuals is neither possible nor reasonable.

However, in organic chemistry involving much larger molecules, there are reactions for which only small parts of the molecules are relevant. It is common to omit the irrelevant *r*est of the molecule and leave a placeholder R (with superscript if several different placeholders are used), e. g., $\texttt{RCOOH + R}^2\texttt{OH} \leftrightharpoons \texttt{RCOOR}^2\texttt{+ H}_2\texttt{O}$ (a process known as Fischer-Speier esterification (Fischer and Speier 1895) – a carboxylic acid and an alcohol reacting to an ester in presence of a dehydrating agent not included in the description). An interesting question for computer simulation of such reactions is then the matching of actually present entities to the patterns specified in these reactions.

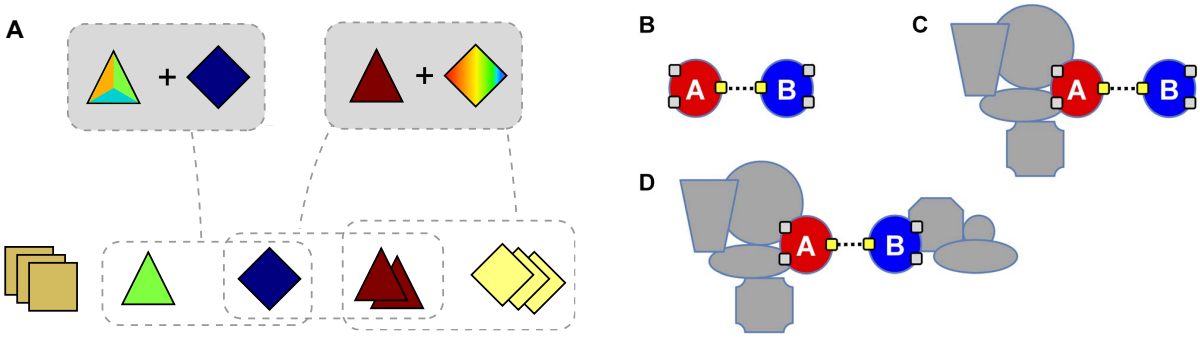


Figure 3.1. A: Reaction rules (top) may apply to entities (bottom) of the same species (illustrated by shape) with an attribute value from a given set (illustrated by color) or to all entities of a species regardless of attribute values (second rule, second reactant); based on Maus, Rybacki, and Uhrmacher (2011, fig. 1). B-D: A seemingly simple reaction rule for binding between two entities may in fact affect several other bound entities that become part of a bigger complex; from Hogg et al. (2014, fig. 2)

In cell biology, the entities of interest are usually macromolecules where even the modification sites have *names* because their actual chemical composition is complex and irrelevant for the result. Common modifications include binding of proteins (i. e. macromolecules) to form arbitrary large protein complexes or smaller modifications like exchanges of phosphate groups or binding of small molecules that may not even be included explicitly in a model. The key point here is that modification of one site need not depend on other sites' modification states (henceforth also called *configuration*), and a useful description format for modification reactions should take this into account and not require separate specifications of one reaction for each possible configuration.

The view of molecules as laid out above follows a pattern of abstractions made in object-oriented programming (OOP). Similar to the categorization of individuals into species in biology, in OOP items of the same category are considered *instances* of the same *class*, with the distinguishing properties called *members* or *fields*, whose *values* may differ between instances. Cell biological modeling formalisms, especially rule-based ones like BioNetGen (Faeder et al. 2005), Kappa (Danos et al. 2007) and ML-Rules (Maus, Rybacki, and Uhrmacher 2011) address this by distinguishing general molecule types (we will use the term *species* here) with *attributes* representing modification sites that can have different *values*. (We will let attributes have names just like fields in OOP, and will usually refer to them by `name:value` pairs.)

A key addition of cell-biological modeling formalisms are abstractions of the changes to and interactions between entities of the same or different species/classes – the rules. These interaction patterns address the problem of combinatorial explosion in model specification by only including those attributes of involved entities directly relevant to the interaction and omitting the others – "don't care, don't write" – which means they may be applicable to entities with different configurations (see also Figure 3.1; an overview

Table 3.1. Classes-instances distinction and terminology of similar concepts in various formalisms.

concept / name in...	category	property	concrete entity	property value	interaction pattern	concrete interaction
OOP	class or type	member or field	instance or object	value	n/a	
biology (taxonomy)	species	n/a	individual	(misc.)	n/a	
Attrib. π	(process definition)	(attribute)	(process)	attribute	(channel)	communi-cation
colored Petri nets	(place)	(color)	token	color	transition	(activated transition)
BioNetGen	molecule type	component	seed species	value	reaction rule	reaction
ML-Rules	entity	attribute	species	value	rule schema	rule (or rule instance)
ML-Space	species	attribute	entity (or individual)	value	rule	(rule match)

n/a: not available (no generalized abstraction). (...): related, but not equivalent concept.

of the terminology used to refer to the OOP-related concepts and interaction patterns in different rule-based approaches is given in Table 3.1).

For example, the mitogen-activated protein kinase 1 (MAPK1) is a protein, i. e. a chain of amino acids, that can be phosphorylated at a threonine at position 183 and a tyrosine at position 185 in the chain, usually abbreviated T183 and Y185, respectively. For our purposes, the *species* `MAPK` has two attributes which each can have two different values: `MAPK1(T183:{p,u},Y185:{p,u})`. Phosphorylation of one site – by a MAPK kinase, `MKK` – need not depend on the state of the other site, allowing specification of T183 phosphorylation by a single rule `MAPK(T183==u)+ MKK ->MAPK(T183=p)+ MKK` instead of two (for `MAPK(T183==u,Y185==p)` and `MAPK(T183==u,Y185==u)`, respectively) or many more if either involved entity had more attributes.

The example rule follows ML-Space' concrete syntax, using the `==` operator to test for equality, i. e. that the attribute whose name appears left of it has the value to the right, and `=` for assignment, i. e. the respective attribute will have the subsequent value after the rule was applied. (This is analogous to their use in common programming languages such as C++ and Java.)

3.2 Using Attributes for Spatial Properties

In the following section, we will look at how attributes are used, especially for spatially relevant properties, in ML-Space and two closely related approaches.

3.2.1 Typed Attributes

In the MAPK example above, if the distinction between the two phosphorylation sites is not relevant, but only the *number* of present phosphorylations, one can simplify them to a single attribute with numerical value `MAPK(np:{0,1,2})` and a single modification rule for either site `MAPK(np<2)+ MKK ->MAPK(np+=1)+ MKK` (with comparison operator `<` and increment-by assignment operator `+=`). This introduces a *numeric type* for attribute values to the description language, with the possibility of performing common computations.

The BioNetGen language (BNGL) offers no such numeric attributes, but has *qualitative types* where all possible values of an attribute are given with the molecule type definition, e. g., `MAPK1(t183~u~p,y185~u~p)` . The other type of attributes BNGL allows is used to express binding sites for other species, which will be discussed later (subsection 3.2.5).

In ML-Rules, attribute values may be numeric or qualitative (i. e. strings, effectively), but entity type definitions contain only the number of attributes with no name and no information on its type, e. g., `MAPK1(2)`. The type of the attributes' values is only implicit from the initial state definition and rules.

3.2.2 From Compartments Attributes to Organizational Levels

Biological functions may involve processes in different biological compartments. For example, transcription factor proteins regulate processing of DNA segments (gene *transcription* to mRNA) in the nucleus, but proteins generally are produced outside the nucleus in the cytosol (*translation* from mRNA). In simple modeling formalisms, entities in different compartments are simply represented as different species, e. g.,

```
                gene -> gene + mRNA_nucleus
        mRNA_nucleus -> mRNA_cytosol
        mRNA_cytosol -> mRNA_cytosol + Protein_cytosol
```

Note that genes occur only in the nucleus.

With an attributed language, the compartment can be encoded as a qualitative attribute, making it clearer that there is only one type of mRNA involved.

```
                gene -> gene + mRNA(location:nucleus)
mRNA(location:nucleus) -> mRNA(location:cytosol)
mRNA(location:cytosol) -> mRNA(location:cytosol)+ Protein(location:cytosol)
```

Strictly speaking, however, location is not a property of the species or molecules themselves. Additionally, the hierarchy of the compartments (the *nesting*, i. e. that the nucleus is surrounded by the cytosol) does not become apparent from the model description. ML-Rules (Maus, Rybacki, and Uhrmacher 2011) was developed to address this and treats compartments as first-class objects, i. e. with attributes and semantics just like other model entities. Our language ML-Space is based on ML-Rules and follows the same approach.

```
                                gene -> gene + mRNA
Cell[Nucleus[mRNA]] -> Cell[mRNA + Nucleus]
                        Cell[mRNA] -> Cell[mRNA + Protein]
```

Here, `Cell[Nucleus[...]]` makes it clear that the nucleus is expected to be inside the cell and to contain other entities. `Cell[mRNA]` refers to mRNA in the cytosol, i. e. the part of the cell that is not the nucleus here. (Note that the second rule above is not actually useful in ML-Rules for lack of a *rest solution* discussed later in subsection 3.5.1). Because all model entities can be contained in and can contain other model entities, the hierarchical structure of compartments is dynamic, so one could specify a rule for the nucleus including its content to leave the cell entirely (not a realistic biological proposition).

In compartmental BioNetGen (cBNGL; Harris, Hogg, and Faeder 2009), a static hierarchy of compartments is defined up front and names of biological compartments are mentioned once in front of rules that only apply in them or next to the reactants.

```
Cell     3  vol_cell          #three-dimensional top-level (outermost) compartment
Nucleus  3  vol_nuc   Cell    #three-dimensional compartment inside Cell

        gene -> gene + mRNA                    #universal reaction rule
mRNA@Nucleus -> mRNA@Cell                      #transport rule
   mRNA@Cell -> mRNA@Cell+ Protein@Cell        #scope-restricted rule
```

This language supports both three- and two-dimensional compartments in the same model, the latter for explicit representation of membrane surfaces as boundaries between three-dimensional compartments. This is relevant for the calculation of reaction rates from (mass action) rate constants (omitted above). ML-Rules effectively allows this, too, by always requiring the reaction rate calculation to be given explicitly (possibly including the volume of the surrounding compartment given as an attribute value). In ML-Space, entities are positioned in actual 2D or 3D space, which makes several aspects of rule application and rate evaluation different from the above approaches where space is not an explicit consideration (aside from the given qualitative, i. e. compartmental part), so compartments of different dimensions in the same model is not (yet) supported.

For a full ML-Space model, one always needs to define the volume and shape of the system in which the model interactions take place. This is done by defining an immobile,

large entity that contains all the others. This entity can also have attributes that change via first-order reactions or along with reactions inside it.

3.2.3 Attributes for Discretized Space

As outlined previously (subsection 2.4.6 "Spatial Formalisms"), with numerical attributes one can express mesoscopic (i. e. RDME-based) spatial simulation (subsection 2.3.5) *explicitly in the model* by adding a location attribute in form of a subvolume index and limiting all other reactions to entities in the same subvolume. For (a single-compartment) example, consider

```
mRNA(svIndex<max) -> mRNA(svIndex+=1)@D
  mRNA(svIndex>0) -> mRNA(svIndex-=1)@D
  mRNA(n=svIndex) -> mRNA + Protein(svIndex=n)@r
```

The first two rules model diffusion from one subvolume to a neighbor, assuming that adjacent subvolumes have numerically adjacent indices. The final rule is a translation reaction as before, where `n=svIndex` assigns the current value of `mRNA`'s `svIndex` attribute (which is left unchanged on the right) to a local variable `n`, whose value is then assigned to the newly created `Protein`'s attribute of the same name.

With compartments as first-class model entities like in ML-Rules, subvolumes can also be made explicit entities (with an `index` attribute so the contained model entities do not need one):[1]

```
SV(n=index)[mRNA]+ SV(index==n+1)[]   -> SV[] + SV[mRNA]   @ D
SV(n=index)[mRNA]+ SV(index==n-1)[]   -> SV[] + SV[mRNA]   @ D
                mRNA                  -> mRNA + Protein    @ r
```

This approach requires no adaptations of the usual reaction rules compared to a non-spatial, non-compartmental approach – when each `mRNA` is located in some `SV`, but the latter is not explicitly mentioned, all relevant rule entities are supposed to be in the same compartment.

The above approaches to encoding space via explicit subvolumes (or subvolume attributes) in the model has major downsides. First, a known diffusion constant D would not actually be a useful rate as shown above, but would have to be adapted to sub-

[1] Here, the local variable `n` is used already on the left rule side to assert that the other involved `SV` is a neighbor. Explicit boundary checks (like "`index<max`") are not necessary this way because when the first reactant pattern (first rule) is matched to the subvolume with highest index, there cannot be any matching subvolume for the second reactant pattern.

This is ML-Space-like syntax again for consistency with other examples here, although the "diffusion" reactions would not be valid in an ML-Space model (where reactions across several non-nested compartments are more diffcult to handle than in not-explicitly-spatial ML-Rules). For entities of the same species on both sides of a reaction, the order is assumed to be the same, i. e. the first entity on the left would be associated with the first entity on the right.

volume size (to be exact, to be divided by the current subvolume side length's square, or more generally for subvolumes of rectangular/cuboidal shape but different sizes, the side length and the distance of centers of diffusion source and target; (Bernstein 2005)), as a smaller subvolume grid corresponds to a finer sampling of space where movement of a diffusing entity is observed more often. (Interestingly, at constant *concentration* of entities, the lower amount of entities in smaller subvolumes exactly cancels this effect out in a way that the number of diffusions per time between a pair of subvolumes is size-independent.) Second, and more importantly, the above examples show two rules for diffusion – in one dimension. Twice or thrice as many are needed for diffusion on a 2D or 3D grid, and twice as many again if periodic boundary conditions are used (and even more for a *von Neumann* instead of *Moore* neighborhood).

Rule-based languages were developed precisely to avoid a large number of rules to express a single conceptual change (here, diffusion). *Spatial Kappa* (Sorokina et al. 2013) is an extension of the κ language (which is very similar to BNGL) to a subvolume-based spatial domain with a notion of (fixed, not hierarchically structured) compartments. Diffusion is specified via rules that can reference the geometry to allow some species to move only along a certain layer representing a membrane and others freely along or across it. Our language ML-Space addresses diffusion by special interpretation of any attribute named `diffusion`. Its value will be interpreted as the usual diffusion coefficient (with length squared over time as unit) and for entities to be simulated in subvolumes handled appropriately.

3.2.4 Attributes for Continuous Space

ML-Space started as an attempt to extend ML-Rules into space, where entities containing other entities are a key feature. In a spatial approach, any entity containing something should have a size (larger than zero). The derivation of the Chemical Master equation, and subsequently the RDME, does not involve particle's extension, i. e. there is no notion of excluded volume. For a consistent representation of a hierarchy of nested spatial entities, each entity should either be contained in another or not overlap it at all. These compartments thus have unique positions (whether in continuous space or a discretized version thereof), giving them distinct properties and making population-based simulation inapplicable.

Individual-based simulation of entities with continuous-space coordinates will be difficult to express in an existing rule-based framework without special spatial constructs. For movement, one could add position attributes and update them by rules, so Brownian motion behavior could be achieved at least in the long run (e. g., by fixed distance, random direction steps in equidistant or exponentially distributed time steps). Spatial proximity of reacting particles could be ensured by conditions involving the particles' position attribute values. However, avoiding collisions, i. e. making particles move into space not occupied by other, would be very difficult to express. Additionally, this way of adding space via the model specification would again lead to more rules and need-

Table 3.2. Special attributes with spatial interpretation.

attribute	value type	required?	changeable in rule?
position	vector	only for top-level entity (otherwise randomly in surrounding entity)	no
size	real number	yes	(pending implementation)
shape	keyword	if size> 0.0	no
diffusion	real number	no (default: 0)	yes
drift	vector	no (default: $\vec{0}$)	
- velocity	real number	no	yes
- direction	real number(s)	no	yes (2D only so far)

Drift is directed movement that can alternatively be specified via its components *velocity* and *direction* (in case only one changes at a time). For some shapes, additional attributes may be required (e. g., an *aspect ratio* for rectangles/cuboids – the only applicable case implemented so far).

lessly complicated rules that are much harder to understand than the concept they are supposed to express.

In ML-Space, the named attributes allow specification of spatially relevant properties (like diffusion as above, but also shape and size) of entities quite easily. Table 3.2 shows attributes with spatial interpretation and the respective expected value types. The handling of movement and non-overlap is then left to the simulator without the need for further specification in the model. One key difference to the subvolume-based spatial approach is that the collision detection establishes proximity, which is conceptually part of the rate for second-order reactions (higher-order reactions should be broken down into second-order reactions with intermediate products as three or more particles are unlikely to collide at the same point). While some approaches handle this by letting particles closer than separately determined interaction radii react with the macroscopic rates (cf. section 2.3.6), in ML-Space in second-order rules involving spatial entities (i. e. those with size > 0), the rate expression is interpreted to be a probability of reaction in case of collision. (While interaction radii of some kind could in principle be added to the ML-Space language via another attribute with spatial interpretation or a rule modifier, the chosen approach simply turned out to be sufficient for the applications so far.) Note that second-order reactions between entities in subvolumes as in the previous section are still interpreted and timed as in the SSA, only when spatial entities are involved the reactions become collision-triggered.

A slightly different approach is pursued in SRSim (Gruenert et al. 2010), a tool that connects BioNetGen and a Molecular Dynamics simulator, LAMMPS (Plimpton 1995). The model species and reactions are defined in the (non-compartmental) BNGL format and supplemented by xml-based definition of relevant particle properties like size and

mass. Applications of SRSim are geared more towards physically accurate simulation of macromolecule formation and the interactions of very few macromolecules, i. e. towards a smaller physical (and time) scale than ML-Space.

3.2.5 Binding

BioNetGen and κ primary focus is on binding between entities (as exemplified by most BNGL example models that use attributes with values from a fixed set – cf. subsection 3.2.1 – using these to express some form of binding, e. g., of a phosphate group not explicitly present as model entity[2]). For a simple example, consider a receptor protein that can dimerize, which is expressed by a binding site attribute: `R(d)`. Binding is expressed in a rule by (an exclamation mark and) an identifier as attribute value that is common to both entities bound:

```
R(d) + R(d) -> R(d!1).R(d!1)
```

The advantage of explicit bindings between existing entities over a separate dimer species `R + R ->D` is, of course, that the `R` retain their identity and attribute values (if there are any other attributes). If they have other binding sites, previously bound entities would stay bound and form part of a larger complex (as in Figure 3.1 right part). The above rule specifies binding independent of the state of any other binding site of `R`, whereas `R(l,d)+ R(d) -> R(l,d!1).R(d!1)` would require a binding site `l` of the first `R` to be free (and stay free) and `R(l!+,d)+ R(d) -> R(l!+,d!1).R(d!1)` would require it to have something bound. Unbinding, in any case, would be expressed by exchanging both sides of the arrow (in fact, `<->` can be used in BNGL for reversible reactions, hence even attributes whose state does not change must be mentioned on both sides of a rule).

ML-Rules as initially published (Maus, Rybacki, and Uhrmacher 2011) did not have explicit treatment of bindings, but has a ν ("new") operator (a concept taken from private channels in the π calculus) that generates a (numeric) value not previously assigned to the respective attribute of any species, which then serves as identification for the generated dimer. The binding and unbinding then conceptually work as follows:

```
R(d==0)+ R(d==0)-> (νx) R(d==x)+ R(d==x)
R((x=d)!=0)+ R(d==x)-> R(d==0)+ R(d==0)
```

For the second (i. e. unbinding) reaction, the bound entities need to be matched by the value of their `d` attribute. (The term `(x=d)!=0` is used here to mean "assign the value of attribute `d` to local variable `x` and test that it is not `0`", which here is the value indicating a free binding site. This syntax is again closer to ML-Space than actual ML-Rules.) When treating these attributes like all numeric attributes, matching this rule requires processing all present pairs of `R`s, and dimers cannot be grouped either for population-based simulation as their values of `d` all differ. A version of the ML-Rules simulator that

[2] Example(s) at bionetgen.org/index.php/BioNetGen_Tutorial#molecule_types

explicitly accounts for attributes that express binding and storing bound entities explicitly has since been developed, which is an order of magnitude faster than the original simulator.

Binding as discussed so far does not have spatial implications, it only creates an (implicit) graph of which entity is connected to which. In spatial simulation, bindings have implications about how the involved entities are positioned relative to each other. One approach to avoid dealing with this is to represent a complex in space by a sphere with a volume corresponding to the combined volume of its parts, and keep the graph representation of actual bindings internal to each complex (Klann et al. 2013). It has been applied to MAPK dynamics in presence of scaffolding proteins, but is not suitable if the geometric structure of the bound entities itself is to be investigated.

In SRSim, the aforementioned xml-based spatial information supplementing the BNGL model also includes binding sites' relative position to each other on the surface of each particle (assumed to be spherical) and forces arising from bonds and volume-exclusion effects, which are also considered in Molecular Dynamics simulation.

In ML-Space, binding between entities was a feature added for a study of actin filaments, long chains of actin molecules that provide scaffolding for the cell. Representation of actual spatial arrangement was thus required. We chose to syntactically separate the binding sites from the other attributes, as the "value" assigned to them is in fact another entity, which is conceptually different from the other types of values. We also use special keywords to identify the state of a binding site, `occupied` (or, shorter, `occ`) and `free` on the left rule side (like, in BNGL, `!+` and a binding site name without qualifications, respectively) and binding site actions (changes) on the right rule side, `release` and `bind`.

```
R()<d:free> + R()<d:free>    -> R()<d:bind>.R()<d:bind>
R()<d:occupied>              -> R()<d:release>
```

The unbinding reaction is specified like a first order reaction (in fact, for scheduling purposes it is one), but also affects the bound entity in that it also loses its binding partner. Instead of simply declaring that a binding site must be occupied, one may also require a binding partner of a certain species and also with some given attribute values by inserting the reactant pattern in place of the `occupied` keyword (which can be regarded as a universally matching reactant pattern), e. g., `R()<d:R()> ->R()<d:release>`. (Note that the binding in the first rule above is sufficiently specified by the `bind` keyword, the dot separating bound entities in ML-Space is syntactic sugar, unlike in BNGL. It was originally intended to be enforced to allow bound entities on the left rule side without specifying the site at which they were bound, but this was not useful for any study using ML-Space so far. Using a unique identifier instead of keywords to allow identification of several bindings was also considered, but not required so far.)
Section 3.5.2 contains further examples and a discussion of limitations of this approach.

3.2.6 Spatial Abstraction Levels Example

ML-Space is designed to simulate entities with spatial positions and allows the modeler to focus on species and their interactions without the need to worry about the detail of the spatial behavior. However, sometimes a fine-grained spatial resolution may not be desired for all kinds of entities in a model.

Consider a protein that can bind several copies of a nucleotide, or, for a larger scale example, a species of cellular organelles `O` that can bind several copies of the same protein `P` on their surface.

A direct translation of this description using explicit binding sites as in the previous section would require the number of every possible binding site to be named explicitly and a rule for the change of each in ML-Space. (This is because binding sites were originally introduced to allow expressing the geometry of larger structures. Multiple equally treated binding sites are another feature that was not needed for any application so far but whose addition to ML-Space should not be too difficult.)

If `P` has no state other than its binding status that needs to be kept, one can describe its binding or unbinding as consumption or production by `O`, which stores the number of bound `P` as a numeric attribute (similar to the phosphorylation example in subsection 3.2.1, with a constant `max` for the maximum number of bound `P`):

```
O(boundP<max)+ P()-> O(boundP+=1)@ pBind.
```

The `O`s may then be simulated as small entities with volume and positions in continuous space, or as populations of dimensionless entities in certain virtual subdivisions. Which of the two approaches is used in ML-Space simulation then only depends on the defined size of entities of species `O`, i. e. whether the size is larger than zero or exactly zero. Note again that for the simulator, a binding *probability* in case of collision is relevant, not a macroscopic rate.

If one can also assume a spatially homogeneous distribution of `P`, one can leave out the explicit species `P` and instead model their amount as an attribute of the surrounding entity, here the `Cell`. Recruitment of `P` by `O` is then a first-order reaction with a rate calculated from the expected collision frequency (which depends on the system volume; cf. section 2.3.6) and this surrounding entity's attribute:

```
Cell(n=availableP)[O(boundP<max)]->Cell(availableP=n-1)[O(boundP+=1)]@rBind*n.
```

Here, the `Cell[...]` on both sides gives the *context*, i. e. surrounding entity in which the rule applies, which means it would no longer apply in other compartments if any were present. The (first-order, mass-action) reaction rate is proportional to the number of `P`, which is not the number of the reactant on the left hand side of the reaction and hence explicitly included in the rate expression given.

When `P` is so abundant that the change to its total is not significantly affected by the

binding reactions, one may as well use a global constant `nP` for its amount:

```
0(boundP<max)->0(boundP+=1)@rBind*nP.
```

Thus, attributes offer a way to incorporate non-spatial behavior to a certain extent.

3.3 ML-Space' Language Syntax

A full ML-Space model is a self-contained description of all information necessary to simulate the modeled system. Not included are parameters relevant to the simulation only, but not to the described model system, e. g., the granularity of the spatial resolution. Table 3.3 contains a simplified version of the ML-Space syntax (see appendix A.1 for the full EBNF). Listing 3.1 contains an example model of a transcription factor protein that can bind to the gene encoding it and prevents its transcription and thus its own production. Binding is not represented via an explicit binding site but rather an attribute of the gene, representing its repression state.

3.3.1 Constants

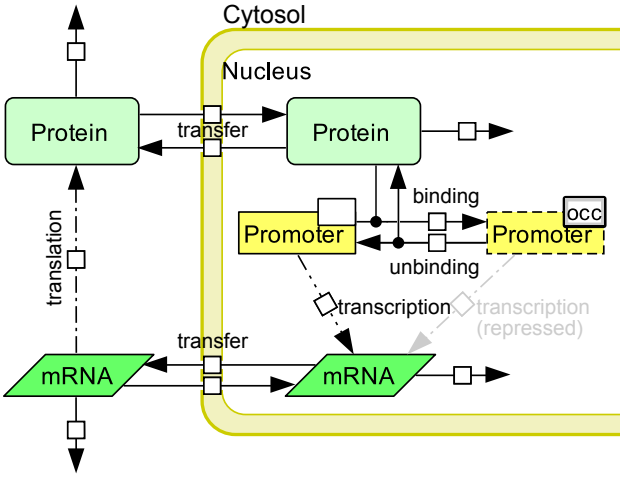
An ML-Space model usually starts with definition of some numeric *constants* that can be used in every place where numbers are expected, e. g., for entity sizes and amounts or for reaction rates. Defining these separately via initially given constants separates the model parameters from the rest of the model description. The experimental framework in which ML-Space models will be simulated also allows constants' values to be replaced (overridden) by externally given ones, e. g., to allow parameter scan experiments without requiring a separate model file with manual changes for each run.

3.3.2 Species Definitions

Species definitions follow the constants section. The types of all entities used in the model are defined here along with the names of their attributes and default values of these. One of these default values will be used when an entity of the species is supposed to be produced (in the initial state definition or on the right hand side of a rule) and not all attribute values are explicitly given. It is not strictly enforced that no other values are ever assigned to the attribute in any rule (but we will assume just that for the examples in this chapter) except when the default size is 0. This interpretation was simply most convenient in the modeling efforts with ML-Space so far, but could be changed in the implementation rather easily. The size attribute is an exception because the model reader/parser distinguishes species that have a spatial extension and those that do not. The latter are designated for subvolume-(RDME-) based simulation and hence population-based treatment and this distinction is made on a per-species basis.

Binding capabilities were added to ML-Space initially for the purpose of modeling actin filaments, which consist of straight chains of actin molecules possibly with branches

Listing 3.1. ML-Space example model of a gene regulatory network like the Hes1 oscillator system (Sturrock et al. 2013) shown schematically on the right. Double slashes // start a comment, i. e. the symbol and the remainder of the line are ignored when the model is read for simulation. The second-order reaction rate $1.66\frac{\mu m^3}{min}$ in line 20 is per interacting pair, calculated from the concentration-based value $\frac{10^9}{M \cdot min}$ by dividing by Avogadro's number and converting the volume units to μm^3.




```
1  D = 0.6 // molecular diffusion; unit: μm²/min
2  rCell = 7.5; rNucleus = 3; // radii, unit: μm
3
4  Cell(shape:ball,size:4/3*pi*rCell^3,position:(0,0,0));
5  Nucleus(shape:ball,size:4/3*pi*rNucleus^3,position:(0,0,0));
6  Gene(size:0,position:(0,0,0),site:{"free","occ"});
7  Protein(size:0,diffusion:D);
8  mRNA(size:0,diffusion:D);
9
10 1 Cell[1 Nucleus[1 Gene + 10 mRNA] + 60 Protein];
11
12 Nucleus + Protein -> Nucleus[Protein] @ p=1 // transfer into nucleus
13 Nucleus[Protein] -> Nucleus + Protein @ p=1 // transfer out of nucleus
14 Nucleus + mRNA -> Nucleus[mRNA] @ p=1
15 Nucleus[mRNA] -> Nucleus + mRNA @ p=1
16
17 Gene(site=="free") -> Gene + mRNA @ 3 // transcription, unit 1/min
18 Gene(site=="occ") -> Gene + mRNA @ 0.1 // repressed transcription
19 // repression and repressor release:
20 Gene(site=="free") + Protein -> Gene(site="occ") @ r=1.66 // μm³/min
21 Gene(site=="occ") -> Gene(site="free") + Protein @ 0.1 // 1/min
22
23 Cell[mRNA] -> Cell[mRNA + Protein] @ 1 // translation; 1/min
24 mRNA ->    @ 0.015 //  mRNA  degradation; 1/min
25 Protein -> @ 0.043 // Protein degradation; 1/min
```

Table 3.3. Sketch of the ML-Space syntax.

Construct	Grammar rule
Model	M ::= $C^+ S^+ I\ R^+$
Constant	C ::= ident '=' numexpr
Species definition	S ::= $\text{ident}_{species}$ '(' ('(' $\text{ident}_{attribute}$ ':' range)* ')' '<' ('(' $\text{ident}_{bindingsite}$ ':' ('(' numexpr | 'any' ')')* '>'
Initial state	I ::= (numexpr EP '[' I* ']')+ ';'
Reaction rule	R ::= (ident_{rule} ':')? RL '->' RR '@' RE
Rule left hand side	RL ::= ES^+ | ES '[' ES* ']' ('+' ES)?
Rule right hand side	RR ::= EP* | EP '[' EP* ']' ('+' EP)?
Rate expression	RE ::= ('r=' | 'p=')? numexpr
Entity pattern (substrate)	ES ::= $\text{ident}_{species}$ '(' '(' (AM|AV))* ')' '<' ('(' $\text{ident}_{bindingsite}$ ':' ('FREE' | 'OCC' | ES))* '>'
Attribute match expression	AM ::= $\text{ident}_{attribute}$ ((OpC numexpr) | ('in' range))
Attr. ex. with variable	AV ::= '(' 'ident' $_{localvar}$ '=' 'ident' $_{attribute}$ ')' (OpC numexpr)?
Entity pattern (product)	EP ::= $\text{ident}_{species}$ '(' '(' ($\text{ident}_{attribute}$ OpA numexpr)* ')'
Comparison operator	OpC ::= '<' | '<=' | '==' | '>=' | '>'
Assignment operator	OpA ::= '=' | '-=' | '+=' | '/=' | '*='

Items in quotation marks are elements of the concrete syntax. X | Y denotes "either X or Y", X? denotes 0 or 1 occurrences of X, X* 0 or more, X^+ 1 or more. For the latter cases, separators between repeated occurrences of the same syntactic construct have been omitted (e. g., '+' between entities on the same side of a rule, ',' between attribute-range-/attribute-value-pairs, or optional semicolons between constants, species definitions and reaction rules, respectively). Also, parenthesis pairs not enclosing any content, e. g., for entities without specified attributes, are actually optional. Syntactic constructs omitted above include *for* loops in the initial state definition for several similar (e. g., equidistantly placed) entities. See A.1: The ML-Space Language's Full Grammar for a complete EBNF.

*numexpr*essions are mathematical terms containing numbers, previously defined constants and, in reaction rules, local variable identifiers, parentheses for grouping and the common operation symbols (+,−,∗,/ and ^ for exponentiation). Ternary operations known from various programming languages for use as if-then-else expressions are also supported. A *range* is an interval with one numeric expression each for the lower and upper bound or a set of explicitly given values.

with fixed angles relative to the main filament. Binding sites in species definitions are thus given as *name:angle* pairs separate from the qualitative and quantitative attributes. When "looking" from one entity's center at two binding partner's centers, the angle between the implied vectors should correspond to the difference of the values given in the species definition. The angles are thus interpreted relative to each other and it is usually reasonable to use a value of 0 for the first defined one.

3.3.3 Initial State

The *initial state* definition that follows consists of entities preceded by their amount and followed by their content (in brackets, `[]`), which (recursively) follows the same pattern. Only entities with a non-zero size can have a content. The top-level entity defines the dimensions of the simulated system and must have a fixed position. The number of coordinates of the position vector specifies the spatial dimensions of the system, i. e. whether it is 2D or 3D. In the example (Listing 3.1 line 10) the position is given as default attribute at the species definition, knowing that there will be only one entity of type `Cell`. While given initial positions (centers) of all entities must be chosen such that the entities do not overlap, entities at different organizational levels can have the same coordinates (in the example, `Cell` and `Nucleus` are concentric spheres). It is possible in principle to have several top-level entities (at different positions) but since they cannot interact due to being immobile there is no advantage over running separate simulations for each of them.

When an attribute of a species is not mentioned in the initial state definition and an interval or several values are defined in the species definition, uniform random sampling is employed (e. g., for the initial repression state in Listing 3.1 line 10).

3.3.4 Reaction Rules

The final and crucial part are the *rules*, each consisting of entity patterns to match on the left and entity modification patterns to the right of an arrow, followed by a rate expression (omitted in most examples in previous sections). Several aspects of rules can be distinguished by syntax features.

- If one side consists of an entity pattern inside another and the other side consists of two patterns for the same two species on the same level, we have a *transfer rule* describing the inner entity crossing the boundary of the outer one (Listing 3.1 lines 12-15).
- If both sides consists of one or more entity pattern(s) inside another and this surrounding entity is of the same species on both sides, we have a *rule with context* (line 23).
- If there is no nesting on either side, we have *universal rule* applicable wherever the entities matching the reactant patterns can occur (all others in the example).

Other useful rule distinctions between non-transfer rules can be made by their order.

1. Zero-order rules (none in example) must have a context, i. e. ML-Space requires explicit mention in which compartments they can happen.[3]
2. First-order rules are universal or context rules with one entity pattern on the left (in a context – Listing 3.1 line 23 – or without one). On the right, there may be a pattern of the same species (possibly with changes to the original entity), a different one (replacement), none (consumption/degradation) or several (production), all in the same context if there was one on the left.
3. Higher-order (≥ 2) rules between dimensionless entities have several entity patterns of species with default size 0 on the left (again with context or without; line 17). These three rule types so far are *time-triggered* and their rate expression, when evaluated and multiplied with the number of applicable reactants, gives the reaction propensity used in the classical SSA.
4. Higher-order rules involving at least one entity with spatial extensions must involve *exactly two* entities, as they are *collision-triggered*, i. e. the spatial proximity required arises from their movement. Here, the rate expression's value will be interpreted as probability that the matching entities react when they collide. Transfer rules are also collision-triggered and a rule for transfer into something differs syntactically from a second-order rule only on the right hand (i. e. product) side.

Since reactions in space require spatial proximity, we do not allow reactions of entities that are located in different compartments (and hence separated by a barrier, e. g., a membrane) and there may be only one level of nesting on each side of a rule. Unlike ML-Rules, we do not allow transfers across two entity boundaries at once like in `Cell[Endosome[P]]->Cell[]+ P` (which includes destruction of one entity), or any other rule affecting entities two or more organizational levels apart.

3.3.5 Differences to ML-Rules

While ML-Space is inspired by ML-Rules, certain differences emerged from necessity and partly convenience. These are (as partly mentioned previously):

Named attributes

Adding spatial attributes to species increases their *arity* (attribute number). Since spatial attributes are fundamental properties for ML-Space' simulation and since at least some of them are common to all entities, it is natural to write them first in the species definition, while they are not usually relevant in the reaction rules. It would therefore be much more inconventient in ML-Space to require placeholder variables for

[3] This is because zero-order rules happening at *all* organizational levels were considered unrealistic, and to make explicit that no rule can happen outside the top-level compartment, i. e. the overall system dimensions are fixed throughout a simulation of an ML-Space model. It also helps avoiding accidentally allowing nested entities of the same species (which is still possible if made explicit, e. g., `S[] -> S[S]`, which may happen if an interval to pick random values from is given as default size for `S`).

each attribute as it is in ML-Rules, so ML-Space adopted the "don't care don't write" approach of referencing used attributes by name and omitting the others as used in κ or BNGL.

Having named attributes also allows us to specify conditions on attribute values of the reactants directly instead of relying on if-then-else constructs in the rate expressions.

Rules crossing at most one level

In ML-Rules, rules can completely rearrange the hierarchical composition of the *solution* (system). Entities in ML-Rules then should not be considered to move but to be consumed and created (another reason why requiring placeholder variables for all attributes of rule reactants makes things easier in ML-Rules, but not in ML-Space). In ML-Space' spatial simulation, particles are tracked individually or via diffusional jumps between neighboring subvolumes. Transferring model entities from one place to a completely different one in the system (by position or place in the organizational hierarchy) is then not a realistic proposition. ML-Space' rules were therefore restricted to cross at most one boundary. (Note that it is still possible for a particle to cross multiple boundaries in one *simulation step*, but more than one *rule* needs to be applied for this.)

Mass-action rates by default

Due to the previous point, most interactions in ML-Space are *local*. Two colliding particles should therefore not be aware how many similar particles are in their surrounding (apart from the particle density's influence on collision frequency, which is already accounted for). It was therefore not reasonable to allow reactant amounts to be bound to variables (like the `n` in ML-Rules's `C(x,y,z):n`) that can be used on the right hand rule side or the rate expression. This simplifies ML-Space' rate expressions and makes ML-Space a bit less flexible in this regard (but also avoids some pitfalls; cf. subsection 3.5.4). Note that rate dependence on entity counts can still be expressed by incorporating an attribute value of the surrounding (context) entity and changing that value whenever the respective entity count changes.

Rest solution

Due to the previous two points, the equivalent to ML-Rules's rest solution is always implicit in ML-Space' rules: a rule regarding the interaction of some entities applies regardless of whether there are other entities in the same compartment. This does not allow modeling of reactions like "a compartment and *everything in it* dissappear" or "two compartments merge into one containing the content of both of them". While the former could easily be added (but can already be handled via multiple rules), the latter would require repositioning particles in ways that are not obvious, or dynamic shapes for the new merged compartment's boundary.

Bindings

Bindings can be expressed in ML-Rules by assigning attributes a unique id as a value – entities with the same unique id as *some* attribute value are considered bound. In ML-Space, where attributes can have a specific type, bindings are special: their "attribute" value is essentially a link to another entity's (which has a link back). It increases model readability significantly by making that property explicit.

Additionally, for an ML-Space application, binding sites with explicit angles were needed. This information needed to be provided in the species definition, for which a syntactic constructs not necessary for any other attributes had to be created. Thus, ML-Space defines binding sites separately from other attributes.

To convert an ML-Rules model to an ML-Space model, some effort is therefore unavoidable. Such a process should probably follow the sections above: adding names for existing attributes (which should be doable via regular expression search-and-replace; it will help to identify attributes used only for binding right here) and adding spatial attributes, transforming each multi-level rule into rules crossing at most one level, simplifying rate expressions or making population effects explicit via a context entity attribute, adding an explicit rest solution to every rule side with a context and finally identify binding reactions and rewrite them with explicit binding sites.

3.4 Semantics Without Space

3.4.1 Initialization and Basic Simulation Steps

The simulation of an ML-Space model proceeds in discrete steps with a continuous time base like other stochastic simulations of rule-based models. In non-spatial simulation, all of those steps deal with timed reactions. In RDME-based simulation, timed events for diffusion between subvolumes are added. In our spatial approach, movement of spatial entities is approximated in discrete time steps as well.

ML-Space uses an *event queue* (i. e. a data structure for efficient storage and retrieval of data with a total order, here an event time; a.k.a. indexed priority queue, *f*uture *e*vent *l*ist, pending event set, and similar names) like the NRM (Gibson and Bruck 2000), but for different reasons. Spatial entities are treated individually as they inevitably differ in at least their position attribute. A population-based treatment of them for timed reactions would require an extra grouping step initially and a recalculation of reactions for two groups when an entity changes. It makes more sense here to calculate timed reaction rates for spatial entities on an individual basis so that a change of one entity requires recalculation of applicable rules for this single entity only. Event times for non-affected entities stay correct and thus should stay on record, hence the event queue.

Also, the RDME-based simulation supported by ML-Space was implemented following the *Next Subvolume method* (NSM; Elf and Ehrenberg 2004, suppl. methods), where the

Algorithm 3.1. Event scheduling for a new spatial entity at simulation time t. Applied to the top-level entity (or entities) at $t = 0$ and then recursively on all others.

```
1  function scheduleEvents(spatial entity e, event queue FEL)
2    d := e.diffusion // value of e's diffusion attribute
3    v := e.velocity // length of e's drift attribute value
4    FEL.enqueue(trigger=e,type=move,time= t + min(l_s/v, l_s^2/d),
         additionalInfo=t) // l_s-mean step length, add.info=time of last move
5    s := sum of rates of applicable reactions
6    FEL.enqueue(trigger=e,type=timed,time= t + (1/s)(− log rand_U(0,1)),*)
7    for each contained spatial entity ce
8      schedule events(ce)
```

"Applicable reactions" are first order reactions where the reactant pattern has the same species as e and zero-order reactions where the context pattern has the same species as e. * When scheduling timed events (line 6), rates (and the respective reaction and the matched entities, as returned by Algorithm 3.3) for each reaction with rate > 0 are stored as additional information for eventual selection of the reaction taking place as in the Direct method (Equation 2.10).

event queue is used to schedule, for each subvolume, the next event originating from it. (Despite the name, NSM uses the Direct method to determine the actual next reactions inside each subvolume.)

The event queue can thus contain three main types of events – a movement step of a spatial entity, a timed reaction of a spatial entity, or an event in a subvolume (which may be a timed diffusion or reaction event), and is filled initially as shown in Algorithm 3.1. The overall simulation loop (including main parts of the spatial steps) is depicted in Figure 3.2. The actual application of reaction rules will be examined in the next subsection.

3.4.2 Matching Rules and Scheduling Events

Matching an Entity to a Pattern

Checking whether a concrete entity in the simulation matches a pattern specified on a left side of a rule consists, intuitively, of testing whether species, attribute values and binding partners of the entity match the name, ranges and entities given in the pattern (see Algorithm 3.2). As already shown in previous examples, this matching may involve storing attribute values of the concrete entity as local variables for subsequent use in matching and modifications. For the former, consider a species with two numeric attributes `a,b` and a rule where these attributes shall have the same value, no matter which: `E(x=a,b==x)`. For this, patterns and their attributes have to be matched in the order they are given in the rule.

Algorithm 3.2. Matching an entity pattern of a rule and a concrete entity.

```
1  function matchEntity(pattern p, entity e, local variables Vars)
2    if p.species != e.species
3      return failure // i.e. exit early
4    for each (attribute name an, match expression expr) pair of p
         // e.g., a<=2, (x=a)<=2, a in [0...2], a==2x
5      value = value of e's an attribute
6      if not matchValue(value,expr,Vars)
7        return failure
8      if attribute match contains variable name x // e.g., x=a, (x=a)<=2
9        add (x,value) to Vars
10   for each (binding site name bn, entity pattern bp) pair of p
11     if not matchEntity(bp, entity bound to e at bn, Vars)
12       return failure
13   return success // changes to Vars are kept
```

The `match expression` *expr* consists of a comparison operator and a numeric expression or the keyword `in` and an interval in the example given above (see Table 3.3 rule AM). The numeric expression may contain variable names which are replaced by the values in *Vars* on evaluation. The model parser performs semantic checks to prevent variable use before definition (binding). For the purposes of binding site matching (line 11), the `free` keyword is supposed to match no entity, only the "unoccupied site" placeholder, while the `occ` keyword matches everything but this placeholder.

Algorithm 3.3. Matching (a left hand side) of a rule. A mapping of actual entities to rule patterns and one for values of local variables are returned for later application of the rule's changes to the entities and for evaluating the rate expression.

```
1  function matchReactionRule(rule r of order n, context entity ce, entities
       e1,...,en)
2    Vars = (name,value) mapping, initially empty
3    Matched = entity list, initially empty
4    if r has context
5      if not matchEntity(r.context, ce, Vars) // may change Vars
6        return failure
7    add ce to Matched
8    for i := 1 to n
9      if not matchEntity(ith reactant pattern in r, ei, Vars) // -"-
10       return failure
11     add ei to Matched
12   return (Matched,Vars)
```

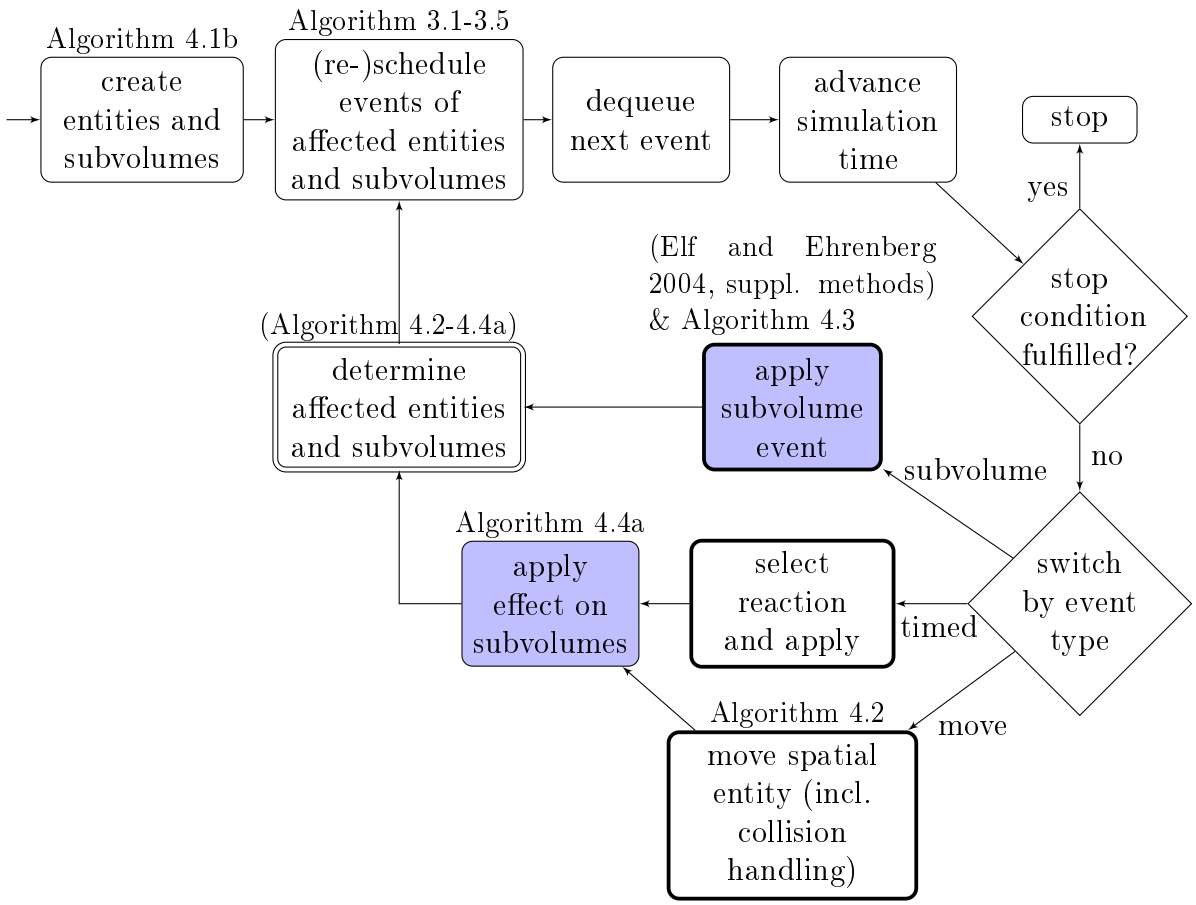


Figure 3.2. Overview of the main simulation loop. Shaded (blue) boxes indicate steps unavailable or skipped in continuous-space-only simulation. Subvolume simulation without dynamic spatial entities, in turn, uses only the "subvolume" path (after the "switch ...").

To check applicability of a rule, the appropriate entities must be matched to the pattern on its left hand side. In Algorithm 3.3, this is detailed for a single permutation of the list of applicable entities. For higher order rules, matching must be attempted for each permutation as, for example, a collision rule `A+B->...` may also apply when a moving `B` collides with `A`.

Order (in terms of sequence) also matters for determining which modification will be applied to which entity if entities of the same species occur on one side of a rule. In

```
E(a==1) + E(a==2) -> E(a=0) + E(a=3) @ ...
```

the entity matched to the first pattern will have its value of `a` set to `0` as a result and the entity matched to the second will end up with `a` set to `3` (same order). In

```
E(a==1) + E(a==2) -> E(a=0) @ ...
```

the entity matched to the second pattern will be consumed (and the other will be modified as above). Ambiguity resolved by considering the sequential order can also arise from patterns of the same species on the right rule side only:

```
E(a==1) -> E(a=0) + E(a=3) @ ...
```

Applying this rule will modify the matched entities attribute to `0` and produce a new entity with attribute value `3`. The general algorithm for matching a rule is shown in Algorithm 3.3.

Rule Instantiation, Network Generation, Matching

Initially, the simulation state corresponds to the initial state given in the model description. In multi-level models like in ML-Rules and ML-Space, it consists of a hierarchy (a tree in the graph-theoretical sense) of entities, which in the former is also called a (nested) *solution* (and the multi-set of all entities on the same level, inside one entity or the top level, is called a sub-solution).

In ML-Rules, identical subtrees are still grouped for population-based treatment by the simulator. Rules can apply to arbitrary large subtrees. The result of matching such a subtree to a rule is called a *rule instance* or an *instantiated rule*, where a concrete entity in a solution is mapped to each reactant pattern in the rule. Since a change somewhere in the tree by one rule might affect all previously evaluated rules, the process of instantiating rules is repeated in each step and no event queue is used (as in the Direct method).

In BioNetGen, with its capabilities to express bindings between entities, the simulation state consists of graph-like structures as well, albeit with a focus on molecular motifs rather than hierarchies. In the original simulator, the rules (with patterns) are actually expanded to reactions (for each concrete entity configuration). The full reaction network can actually be infinite: consider a polymerization where, for example, a styrene monomer binds to the last styrene molecule in a polystyrene chain. This rule would subsume a reaction each for monomer and a "chain" of length 1 (degenerate/initial case), monomer and chain of length 2, 3, and so on. For this reason, the reaction network is generated based on the initially present entities (seed species) and the entities produced from them by the generated reactions, which is repeated until the process terminates or a cutoff is reached. The generated reaction network can then be used, for example, as input to static analysis tools or for ODE simulation (Blinov et al. 2004; Faeder et al. 2005). With *NFsim* (Network-Free Stochastic Simulator; Sneddon, Faeder, and Emonet 2011), however, there is also a simulator that does the network generation on the fly like ML-Rules' instantiation.

In ML-Space, the equivalent to a rule instance or generated reaction would be the *match* consisting of the list of entities matched to a rule's left side (and local variables, and, in addition to what is shown in Algorithm 3.3, the modifications prescribed by the reaction for each entity, the entities to be produced as well as propensity calculated by

Algorithm 3.4. Rule matching for a spatial entity.

```
1  function matchSpatialTimedEvents(entity e, zero order rules R_0, first
       order rules R_1)
2    M := list of (rule match,propensity) pairs, initially empty
3    for each rule r ∈ R_0
4      m := matchReactionRule(r, e) // Algorithm 3.3; e is the context (i.e.
           surrounding) here
5      if m not failure
6        a_i := evaluate(r.rateExpr, m.Vars) * e.size // rule rate
             expression's unit is per time per volume (or per area in 2D)
7        if a_i > 0 add (m,a_i) to M
8    for each rule r ∈ R_1
9      m := matchReactionRule(r, surr(e), e) // e is the reacting entity
10     if m not failure
11       a_i := evaluate(r.rateExpr, m.Vars)
12       if a_i > 0 add (m,a_i) to M
13   return M
```

evaluating the rule's rate expression). For timed reaction, the list of these matches is what is actually stored in the event queue. This event scheduling happens for spatial entities and on populations in a subvolume.

Scheduling for Spatial Entities

Spatial entities in ML-Space are always treated on an individual basis. Rules are thus evaluated and reactions scheduled for each one individually (Algorithm 3.4). For simplicity, an early version was designed such that each timed reaction would affect one spatial entity only to minimize potential event invalidation, i. e. scheduled events in the queue becoming obsolete due to changes to the affected entity. However, upward and downward causation cannot be expressed with this limitation in place. Also, collision-triggered reactions (arising from a move event) can change up to two entities in a way that affects their scheduled reactions, e. g., due to an attribute value change making some first-order rules no longer applicable.

Current ML-Space keeps the approach of regarding timed reactions as centered on one *trigger* entity. This has the advantage of making invalid/obsolete events in the queue identifiable by their trigger (many event queue implementations can be easily expanded with an inverse lookup for efficient removal or *requeue* of specific events at arbitrary positions in the queue). It supports upward causation by allowing attribute changes to a context entity, if present. Consider, for example, an attribute counting the content of a certain entity type changing in response to a change of one such entity:

```
Cell()[P(phos=="yes")] -> Cell(phosPCount-=1)[P(phos="no")] @ rDephos
```

This reaction would be triggered by the individual `P`, but still require re-evaluation of rules applicable to `Cell`. Now imagine a rule inside a context depending on an attribute value thereof, e. g., reproduction depending inversely on the number of present entities:

```
Cell(n=totalPCount)[P] -> Cell(totalPCount+=1)[P+P] @ rRepro*(100-n)
```

This would constitute downward causation. Changes to the context entity's attribute value here affect all contained entities of the given species, meaning that all first-order rules affecting `P`s in this `Cell` (context) have to be re-evaluated (cf. Figure 3.2 last box in loop, with double outline).

Rule Matching in Subvolumes

Reactions relevant for RDME-simulation, i. e. subvolumes, are

- zero-order reactions producing dimensionless entities
- first- and higher order reactions with only dimensionless entities on the left rule side (and also on the right; except for the context, if present).

In ML-Space' hybrid of meso- and microscopic simulation approach, each subvolume is located inside a spatial entity (for reasons laid out in the next chapter), which provides the context entity for context-dependent rules. The subvolume content is stored in pairs of entities and their amounts. The next reaction event in each subvolume is calculated following the Direct method as shown in Algorithm 3.5, taking into account the amounts for each reactant and also subvolume's size (volume) as the volume determines concentrations, which in turn determine collision frequency and thus likelihood to react.

When matching attributed entities to patterns that need not narrow down all the entities' attribute values, special handling is required for cases where reactant patterns overlap, i. e. can match the same entities. For example, matching the rule `E(a==0)+ E(b==1)->` ...in a subvolume that contains `4 E(a=0,b=0)+ 5 E(a=0,b=1)+ 6 E(a=1,b=1)`, there would be 9 entities matching the first pattern and 11 matching the second, but each number includes 5 entities matching both, so the number of possible collisions for the reaction is smaller than $9 \cdot 11$, the product of amounts of entities matching each reactant (it actually is $84 = 4 \cdot 5 + 4 \cdot 6 + 5 \cdot 6 + 5\frac{5-1}{2} = 9 \cdot 11 - 5 \cdot 5 + 5\frac{5-1}{2}$).

3.5 Expressiveness Comparison

3.5.1 Contained Entities: Modification or Replacement

Reactions in ML-Space are events related to individual entities (zero/first order) or interactions with some spatial constraints (collision or same subvolume), i. e. limited locally. While rules may contain consumption and production, they are mostly about changes to existing entities. They affect different organizational levels only in cases of transfer rules or with a context whose attribute changes.

Algorithm 3.5. Rule matching on subvolume, which contains a population of dimensionless entities. The population S (i. e. the subvolume *state* or `content`) is given as set of entity-amount pairs (e, n). $|e \text{ in } S|$ denotes the n for which $(e, n) \in S$, i. e. the amount of e in S (which can be interpreted as multiset or *entity* $\mapsto$ *amount* mapping).

```
1  function matchSubvolumeRule(subvolume sv, rule r)
2    Vars := ∅
3    if r has context and not matchEntity(r.context,surr(sv),Vars)
4      return ∅
5    C := {(Vars, ∅)} // set of pairs of matched variables and multiset of
         contained entities e already matched
6    for each reactant pattern p in r
7      C := matchSubvolumeReactant(p,sv.content,C) // (see below)
8    M := ∅
9    v := sv.size^(1−r.order) // mass-action volume adjustment
10   for each (V, c) in C
11     M := M ∪ {( V, evaluate(r.rateExpr, V) · v · ∏_{e∈sv.content} C(|e in sv.content|, |e in c|) )}
12   return M
```

```
1  function matchSubvolumeReactant(reactant p, subvolume content svc, matches
       so far C_before)
2    C_new := ∅
3    for each (V_b, c_b) in C_before
4      for each e ∈ svc
5        if |e in c_b| = |e in svc|
6          continue with next e // all of this entity already matched here
7        V_new := V_b // create copy because next method may modify it
8        if matchEntity(p, e, V_new) // see Algorithm 3.2
9          c_new := c_b ∪ e
10         C_new := C_new ∪ {(V_new, c_new)}
11   return C_new
```

```
1  function scheduleSubvolumeEvent(subvolume sv, event queue FEL)
2    M := ∅ // for pairs of (rule match, reaction rate)
3    for each subvolume rule r
4      M := M ∪ matchSubvolumeRule(sv, r)
5    D := ∅ // for pair of (entity type, diffusion rate)
6    for each (entity e, quantity n) ∈ sv.content
7      D := D ∪ (e, n · e.diffusion/l^2 · |sv.neighbors|)
8    s := Σ_{(_,r)∈D∪M} r
9    FEL.enqueue(trigger=sv,type=subvolume,time=t + (1/s)(− log rand_{U(0,1)}),M ∪ D)
```

The scheduling (final lines) works analogously to Algorithm 3.1. l is the subvolume side length (line 7, assuming a uniform square grid; more complex subvolume and grid patterns can be handled by the ML-Space RDME-based simulator, but not yet defined via the modeling language).

There are no spatial constraints in ML-Rules and BNGL, and the view of the system state as a graph is more important. In the former, a rule schema describes the *replacement* of the part of the hierarchy/tree that matches the pattern on the left with entities defined on the right of the rule. In a solution (i. e. system state) of `1 C[2 A + 3 B]` the rule `C[A] -> C[]` would match because the sole `C` has an `A` in it, and on application *all of it* would be replaced by a new, empty `C`, i. e. the other `A` and all `B`s in the former `C` would disappear with it. To keep the content, one needs to explicitly specify the so-called *rest solution*, e. g., `C[A + sol?] -> C[sol?]`. This construct allows reaction like moving the whole content of a compartment out of it, and via the *functions on solutions* extension (Nähring 2014; Nähring et al. 2014; Warnke, Helms, and Uhrmacher 2015) also the splitting of content, e. g., on cell division.

In BioNetGen, where the graphs of interest represent molecule complexes, degradation rules for one molecule also have the effect of degrading a whole complex[4], i. e. `A() -> Trash()` would match any `A` regardless of binding state and would also eliminate all current binding partners. If the rule explicitly mentions one binding partner, interestingly, it would only match if there are no others, i. e. `A().B() -> B()` would not match an `A` bound to a `B` and a `C`. A special keyword can be used to avoid this constraint and delete individual molecules possibly inside a complex which itself is kept (but most likely broken up): `A() -> Trash() rate DeleteMolecules`.

ML-Space instead is based on the intuition that an event in one spatial region should affect only the immediate vicinity. Thus, rules can only affect the entities matching a pattern explicitly mentioned, as the entities for side effects similar to the above might be far away spatially. The only exception are releases of bonds, which may be specified for one entity without mentioning the binding partner itself (which, however, is in the immediate vicinity by definition).

For lack of an explicit rest solution, ML-Space cannot express splitting up content of one entity. However, the example of cell division would lead to further complications with respect to spatial handling, either requiring odd shapes (e. g., half-spheres) that gradually change (to smaller spheres) or moving the content around when the surrounding entity changes (e. g., when one sphere is replaced by two others with equal total volume). When an entity with content is deleted in ML-Space, the content stays in place and the deleted entity's surrounding would become the content's new surrounding entity. In other words, the degradation would be interpreted as dissolution of the boundary only. (Attempted degradation of the top-level entity would lead to an error.)

3.5.2 Binding: Entity or Complex View

With ML-Space allowing no higher than second-order reactions between spatial entities, only one bond – between the colliding entities – can be established in one reaction, while in BioNetGen there is no principal limitation preventing establishing multiple bonds between entities pulled from the (well-mixed) population at once.

[4] Source: bionetgen.org/index.php?title=BNGManual:Delete_a_Molecule

In BioNetGen, a reaction affecting bound entities is viewed as a reaction *of the complex* and its definition, e. g., `A(a!1).B(b!1)->...` for species `A` and `B`, each with a binding site of the same name, is considered a single pattern. In ML-Space, such a reaction is still viewed as referring primarily to one single entity and the bound one is specified as binding partner to be matched, e. g., `A()<a:B()> ->...` or `B()<b:A()> ->...`. This makes it easier to schedule reactions along with other first-order reactions affecting either `A` or `B`.

ML-Space' syntax extends the "don't care – don't write" approach to binding sites as those entities mentioned only for matching binding partners do not have to be repeated on the right of the rule (unlike in BNGL, or with ML-Rules's ν workaround). A disadvantage is that they cannot actually be mentioned on the right even if this is desired, i. e. these entities are unavailable for immediate modification.

For example, if the `A` and `B` as above each had a phosphorylation site `ps`, the state of two molecules' site can be changed at the same time in BNGL:

```
A(ps~u,a!1).B(ps~u,b!1) -> A(ps~p,a!1).B(ps~p,b!1)
```

In ML-Space, this cannot be equivalently expressed, but a workaround with a second rule to be applied immediately after the first one can be used:

```
A(ps==u)<a:B(ps==u)> -> A(ps=p) @ ...
B(ps==u)<b:A(ps==p)> -> B(ps=p) @ Infinity
```

Here, `B(ps==u)<b:A(ps==p)>` is considered an indication for one half of the desired reaction having taken place already, and the other half is completed by the *confluent* (infinite rate) reaction. This is not as elegant as the BNGL format and gets even less so when `B(ps==u)<b:A(ps==p)>` is in fact an allowed state (where an attribute indicating "incomplete pair phosphorylation" would have to be added to species `A` and its state set in the first rule and reset in a third one).

Also, in the ML-Space expression `A()<a:B()>` the name of `B`'s binding site at which `A` is bound is not and cannot be specified, but this is not a severe limitation as two entities can have only one bond between them.

The bound entities can effectively also be nested, making matching of arbitrarily large complexes possible, although not necessarily desirable. For example, if styrene monomers `S` can bind other styrenes `left` and `right`, the following rules would all match polymer chains of length three.

```
1 S()<right:S()<right:S()>> -> ...
2 S()<left:free, right:S()<right:S()<right:free>>> -> ...
3 S()<left:S(), right:S()> -> ...
4 S()<left:S()<left:free>, right:S()<right:free>> -> ...
```

To be exact, the first two rules would match the first `S` in the chain of three while the latter two rules would match the middle one, and rule 1 and 3 would match any such `S` in a three molecule segment of an arbitrary large chain while the even-numbered rules would match only chains of length exactly three (requiring the `left` binding site of the

leftmost and the `right` site of the rightmost molecule to be `free`).

3.5.3 Symmetric Second-Order Rules and Instantiation

Consider a second-order reaction that requires two entities whose attribute values may or may not differ. Care must be taken regarding the mass action rate of the reaction. As example for a species (i. e. molecule type in BNGL) `S(a:{0,1})`, the rule `S() + S() -> ... @ r` sums up three reactions (Faeder, Blinov, and Hlavacek 2009, note 29):

```
S(a==0) + S(a==0) -> ... @ ?
S(a==0) + S(a==1) -> ... @ ?
S(a==1) + S(a==1) -> ... @ ?
```

If n_0 and n_1 are the amounts of reactants with the respective value of attribute `a`, then there are $n_0 n_1$ possible collisions for the second reaction to happen, but only $\frac{n_0(n_0-1)}{2}$ for the first and $\frac{n_1(n_1-1)}{2}$ for the third. BNG automatically corrects the rate of the symmetric reactions to `r/2` when expanding the rule to reactions. However, this is only appropriate if the modeler did not consider the putative symmetry of the original rule and did not adjust the rate accordingly already. In that case, it would be more appropriate to adjust the unsymmetrical (middle) reaction's rate by factor 2.

The latter would effectively happen in ML-Rules and ML-Space, where, in presence of entities of both configurations, for the unsymmetrical case the rule would be instantiated/matched twice:

```
S(a==0) + S(a==1) -> ... @ r
S(a==1) + S(a==0) -> ... @ r
```

(although in ML-Rules entity counts have to be incorporated explicitly into the rate, which may lead to the following pitfall).

3.5.4 Non-linear Dependencies on Amounts

In ML-Space, rate expressions are assumed to give the mass action kinetic rate and the rate constant is multiplied with the amount of matching reactant(s) when calculating the time to the next reaction. In ML-Rules, arbitrary expressions are allowed and the reactant amounts must be specified explicitly. Consider a species with at least one attribute, for simplicity `S(a:{0,1})` again. In the rule

```
S():n -> ... @ f(#n)
```

the `:n` is supposed to capture the (amount of) matching reactants, which can then be used in the rate expression, e. g., `@ r*#n` for mass action kinetics. However, in ML-Rules such a rule schema is instantiated to several rules matching specific reactant configurations, effectively behaving like

```
S(a==0):n0 -> ... @ f(#n0)
S(a==1):n1 -> ... @ f(#n1)
```

Now if `f` is a non-linear function of `#n`, e.g. involving saturation or squaring, the sum of rates of the instantiated rules differs from what the modeler would expect from the single rule (schema)[5].

Remembering the example from the previous section, if the collision of two `S` changes only one of them, i. e. `S():s1 + S():s2 ->S()+ ... @r/2*#s1*#s2`, it may be tempting to omit the pattern occurring on both sides since its amount should also be captured by the remaining one, shortening the rule to `S():s ->... @r*#s*(#s-1)/2`, but this is correct only if `S` has no attributes that can differ between entities in the same solution.

BioNetGen initially allowed only mass-action kinetics, but later Michaelis-Menten kinetics were added and eventually arbitrary functions[6]. Species/patterns whose amounts shall be used as variables in functions must be specified separately (i. e. outside of rules) as observables. (In JAMES II-based simulators, observation is a concern separate from the actual simulation steps and thus model properties to be observed are not usually mixed with the model itself.) It is thus the modeler's responsibility to correctly define the amount used in the function, and even possible to use amounts of not directly related entities.

In ML-Space, similar entity counts can be expressed as attributes of the surrounding entity (as seen in subsection 3.2.6 and 3.4.2), whose changes need to be explicitly included in rules, making it also the modeler's responsibility to specify what is counted.

3.5.5 Collisions from Inside vs. Transfer Out

As mentioned in subsection 3.3.4, by looking at the left side of a rule alone, one cannot necessarily distinguish transfer and reactions. For example,

```
E(a==0) + C -> E(a=1) + C @ ...
E(a==0) + C -> C[E(a=1)] @ ...
```

are rule with the same left hand side, (1) for a reaction rule for an attribute change of an `E` that collides with `C` and (2) for a transfer rule for an `E` entering `C`, including an attribute change. Both rules need to be evaluated on collision. In contrast, a single entity inside another on the left hand side can start a transfer rule or a first order reaction rule with context:

```
C[E(a==1)] -> C + E(a=0) @ ...
```

[5] F. Haack and A. M. Uhrmacher, personal communication. The actual example involved cells with a cell cycle state attribute, among others, and a rule concerning proliferating cells whose rate depended non-linearly on the total number of proliferating cells. For an ML-Space model, however, entire cells with a complex internal state make unlikely candidates for representation as dimensionless (i.e. size 0) entities in spatial simulation.

[6] bionetgen.org/index.php/BNGManual:Functions; thanks to Tobias Helms for discussion of ML-Rules's behavior in equivalent cases.

```
C[E(a==1)] -> C[E(a=0)] @ ...
```

The latter is a timed rule, while the former, a transfer out of something, can be applied only when the entity to be transfered moves against the boundary of its surrounding entity. The syntactic patterns shown – nesting on one side but not the other means transfer, nesting on both sides gives a context and the actual reaction happens inside it – was chosen simply because it was deemed the most intuitive way to express these concepts.

However, this choice leaves no easy way to express a (non-transfer) reaction changing an entity that moves against its surrounding entity's boundary, i. e. a collision from inside. While handling of such reactions could easily be added to the simulator, a syntactic element to make the rules distinguishable from context reaction rules would have to be chosen first. This is another example of an edge case requiring special handling that was not needed for ML-Space' applications so far.

Note that absorbing boundaries can still be expressed, albeit not very elegantly, by surrounding everything with an (explicitly) all-consuming entity:

```
E(...)                                    // example model entity
Cell(shape:...,size:...,position:(0,0,0)) // desired top-level entity
Absorber(...) // actual top-level entity, larger than Cell by at least
    twice E's diameter in each direction, same position as Cell
1 Absorber[1 Cell[n E + ...]];

Cell[E] -> Cell + E @ p=1             // E crosses "absorbing" boundary...
Absorber[E] -> Absorber[] @ r=Inifinity //...and is destroyed outside in
    the immediately following step
```

4 Hybrid Spatial Simulation

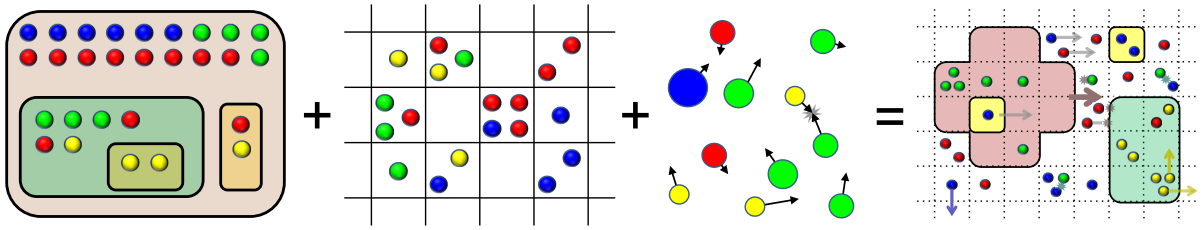


Figure 4.1. Illustration of spatial aspects (as in Figure 2.2) combined by ML-Space.

As we saw in the previous chapter, the ML-Space modeling language introduces spatial aspects to rule-based languages' relatively intuitive approach of reaction pattern descriptions. In describing syntax and rule evaluation semantics, so far we mostly disregarded actual spatial behavior. However, spatial constraints can be quite relevant, as some reactions may alter the spatial positions of entities and fail when there is no space at the target, e. g., for binding with given relative positions to already bound entities or transfer across a boundary with other entities already on the other side. After briefly revisiting and expanding on the main simulation loop, this chapter will be about the dynamics of individual-based simulation in continuous space of entities with a size, and possibly hierarchical nesting.

The ML-Space language is intended to separate the model of the actual system as much as possible from the detail of the simulation approach, and thus, as stressed before, applicable to both individual-based and subvolume-(RDME-)based simulation. After a brief description of how ML-Space handles subvolume simulation alone, the other main aspect of this chapter will follow: a *hybrid* method bringing subvolumes and multi-level individual-based simulation together.

The developed approach is hybrid with respect to species, which can occur anywhere in the simulated space but must be either of the dimensionless type (in subvolumes) or spatial (with extensions and hence individuals). A different approach bringing these two together is the *Two Regime method* (Flegg, Chapman, and Erban 2011), which is hybrid with respect to space, i. e. all entities would be simulated as part of subvolume populations in one part of the space and as individuals (still dimensionless, however) in the other.

4.1 Main Loop and Event Types

ML-Space is implemented in James II (Himmelspach and Uhrmacher 2007; Ewald et al. 2010), a Java-based framework for modeling and simulation with a focus on discrete event methods. The events to process are stored in an event queue (future event list) and the main loop of the simulation simply consists of taking the event with the least time stamp from the event queue and handling it according to its type possibly resulting in new events being scheduled (shown in Algorithm 4.1a and before as flowchart in Figure 3.2). The initialization involves placing spatial entities and creating subvolumes

Algorithm 4.1a. Overview of symbols/abbreviations used in pseudo-code (top) and simulation main loop (bottom).

FEL	future event list (event queue)
ev	event from FEL
t_{ev}	time ev is scheduled to happen
ent, ce, se, e	model/simulation entities (usually: se – entity surrounding another relevant entity, ce – colliding entity)
ent.x	value of ent's attribute x
$surr(ent)$	entity surrounding ent
$space(ent)$	space occupied by ent
sv	subvolume (grid cell containing dimensionless entities)
$surr(sv)$	entity to which sv belongs
$Changes$	collection of entity moves, attribute and subvolume state changes during current step

Furthermore, capital initial letters usually denote sets or lists (except diffusion constant D) while symbol names in small letters denote single objects.

```
1  FEL,E,SV := initialize(model.initialState,...) // see Algorithm 4.1b
2  (t_ev,ev):= FEL.removeFirst
3  advance simulation time to t_ev
4  while not stop condition fulfilled
5    if ev.type = move // of spatial entity
6      Changes_space := moveEvent(E,ev.trigger,ev.additionalInfo)
7      Changes_sv := subvolumeEffects(SV,Changes_space)
8    else if ev.type = timedReaction // of a spatial entity
9      Changes_space := zeroOrFirstOrderEvent(E,ev.trigger,ev.additionalInfo)
10     Changes_sv := subvolumeEffects(SV,Changes_space)
11   else  // ev is subvolume event
12     (Changes_space,Changes_sv) :=
           subvolumeEvent(SV,ev.trigger,ev.additionalInfo)
13   FEL := rescheduleEvents(FEL,ev,Changes_space,Changes_sv)
14   (t_ev,ev):= FEL.removeFirst
15   advance simulation time to t_ev
```

Algorithm 4.1b. Initialization of spatial entities, subvolumes and events.

```
1  function initialize(initial model state I, subvolume side length l_sv)
2    TLE := top-level spatial entities in model.initialState
3    E := ∅
4    for each se in TLE
5      E := E ∪ placeContainedEntities(se,I)
6    d_small := diameter of smallest spatial entity in model
7    l_sv := min(l_sv, d_small/√dim) // dim ∈ {2,3} (dimensions); ensures that each
       spatial entity contains at least 1 subvolume
8    SV := createSubvolumes(l_sv,E,model.initialState)
9    scheduleEvents(TLE,FEL)            // Algorithm 3.1
10   scheduleSubvolumeEvents(SV,FEL)    // Algorithm 3.5
11   return FEL, E, SV
```

```
1  function placeContainedEntities(entity se, initial state I)
2    E := ∅
3    for each spatial entity e in se as specified in I
4      if e.position is defined
5        check that e is completely inside se
6        check that e overlaps no entity in E
7      else
8        place e randomly inside se not overlapping any element of E
9      add e to E
10   for each spatial entity e in E
11     E := E ∪ placeContainedEntites(e,I)
12   return E
```

```
1  function createSubvolumes(length l, spatial entities E, initial state I)
2    b := bounding box of top-level entities in E
3    SV := split b into equal boxes with all sides ≤ l
4    for each subvolume sv in SV
5      e := smallest spatial entity in E that contains sv.center
6      if no such e exists    // top-level entity may be round
7        remove sv from SV
8      else
9        sv.surroundingEntity := e
10   for each spatial entity e in E
11     C := dimensionless entities in e as specified in I
12     distribute C randomly over all subvolumes surrounded by e
13   return SV
```

Some data structures used internally are simplified above. For example, spatial entities are not stored in a set E but rather a tree-like structure for efficient lookup of top-level entities and containment relations (including which nodes are leaves, for `createSubvolumes`' line 5). ML-Space also maintains a mapping of spatial entities to subvolumes belonging to each (final line).

for hybrid simulation and is shown in Algorithm 4.1b.

The discrete events, for our purposes, are of one of the following types:

1. position update (i. e. move) of a spatial entity, which may trigger second-order reactions between spatial entities (Algorithm 4.1a, line 6),
2. first or zero-order reaction involving a spatial entity (line 9))
3. subvolume event (i. e. reaction or diffusion between subvolumes; line 12)

Simulation ends when a certain stop condition is fulfilled. Using the JAMES II framework's capabilities, this may relate to the execution of a fixed number of steps, the passing of a given amount of real (i. e. wall clock) time, model-related criteria like the amount of some species reaching a steady state, but most often the passing of a given amount of simulation time. (For the latter case, the simulation time is updated and the stop condition checked before the actual execution of the event as otherwise the simulation would go one step too far and the final simulation state would be marginally off the desired result.)

4.2 Brownian Motion and (Multi-Level) Interactions in Continuous Space

As in other stochastic particle-based simulation, move events of spatial entities consist of updating their position by a random vector and, if this results in a collision, finding applicable reactions between the colliding entities (Algorithm 4.2). In case of failure, a new move attempt is made, up to a fixed number of times (usually, four, unless a reaction was matched but not executed due to having a probability <1). In case of successful reactions, the moving entity is moved only far enough to touch the reaction partner, such that there is no overlap at the end of each step. Nested entities, e. g., the content of a mitochondria or proteins inside a lipid raft, are moved along, staying in the same relative position to its surrounding (`moveAttempt` line 2).

Reactions between spatial entities may include

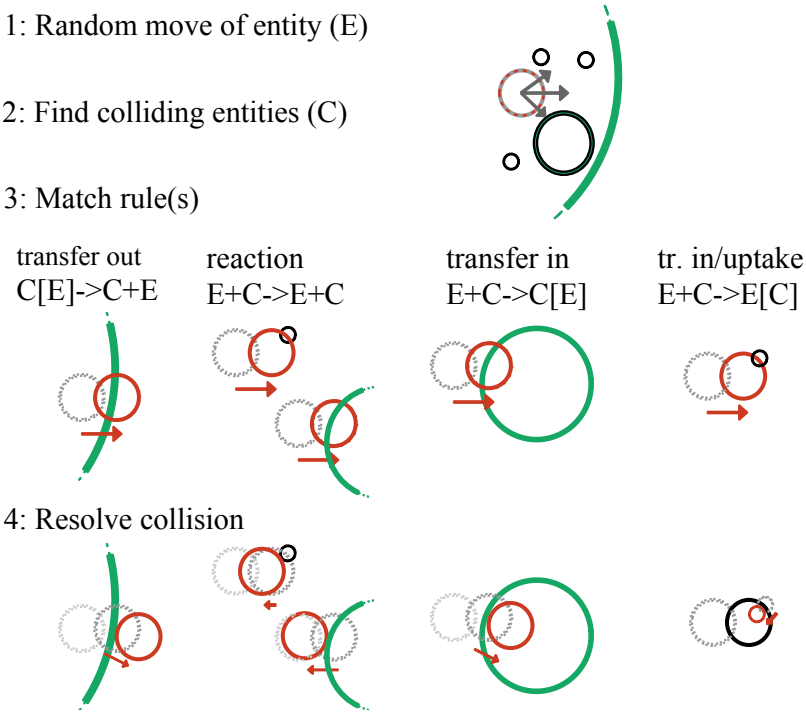
- changes of attribute values (e. g., gain or loss of a phosphorylation), but also
- changes in hierarchical composition, i. e. an entity leaving the compartment that surrounded it or entering another, i. e. a transfer, as well as
- degradation (i. e. disappearance) and creation of spatial entities.

In case of transfers into (or out) of a compartment with a distinct boundary, e. g., cellular organelles, the transferred entity is moved far enough towards (away) from the center of the entered (left) compartment that it only touches its boundary but does not overlap it. This may result in collisions (i. e. overlap) with other entities it did not originally collide with, so the algorithm for dealing with collisions including matching and applying rules has to be applied recursively until arriving at a situation without or with unresolvable collisions (see Figure 4.2).

If a move resulted in collision with several entities, the resolution is attempted first with the largest thereof – if the moving entity enters a larger one, this will resolve

previous collisions with entities outside the entered one. If the moving entity is larger than the ones it collides with, an applicable rule must be found for each of those (e. g., a moving lipid raft taking up several receptor proteins on its way) or the move attempt fails (`moveAttempt` line 33 ff). Newly created spatial entities are positioned in place of consumed ones, if possible, or near the colliding entity (i. e. touching but not overlapping it), as the originally moving entity may have moved further away from the original collision site due to the aforementioned recursive calls. Creating the new entities before the collision resolution is impractical as (a) collision resolution then might lead to further

Algorithm 4.2a. Move of spatial entities: Illustration of general approach (top; (red) arrows indicate move of the entity above it; previous position with dotted (gray) and subsequent position solid (red) outline) and loop for multiple attempts (below).




```
1  function moveEvent(spatial entities E, moving entity ent, time of last
       move t_lm)
2    Δt := t_sim − t_lm
3    D := ent.diffusion
4    attemptCount := 0
5    do
6      attemptCount := attemptCount + 1
7      u⃗ := randomVector(D, Δt)
8      Changes := moveAttempt(ent, u⃗, E)
9    while changes = ∅ and attemptCount < threshold
10 return (model.s, changes)
```

Algorithm 4.2b. Move of spatial entity: single attempt with collision detection.

```
1  function moveAttempt(entity to move ent, move vector u⃗, other entities E)
2    Changes := move(ent, E, u⃗)  // update position of entity and all contained
         ones
3    if space(ent) ⊈ space(surr(ent))
4      // collision with surrounding boundary: transfer out or failure
5      Change_tr := transferOutAttempt(ent, surr(ent), E)
6      if change_tr ≠ ∅
7        return Changes ∪ Change_tr
8      else
9        undo(Changes, ent, E)
10       return ∅
11
12   Col := collisions(ent,entities in E also in surr(ent))
13   if Col = ∅ // no collision: move successful
14     return Changes
15
16   Col := sortBySizeDescending(Col)
17   ce := Col.first
18   // collision with other spatial entity: try same-level reaction first:
19   Change_col := reactionAttempt(ent, u⃗, ce, E)
20   if Change_col ≠ ∅
21     return Changes ∪ Change_col
22
23   // second, try transfer of moving entity into larger one
24   for each ce in Col with ce.size > ent.size
25     Change_tr := transferInAttempt(ent, ce, E)
26     if Change_tr ≠ ∅
27       return Changes ∪ Change_tr
28   if ent.size ≤ Col.first.size
29     undo(Changes, ent, E)
30     return ∅
31
32   // if only smaller colliding entities, try transfering these into ent
33   do
34     ce := Col.removeFirst
35     (Change_tr := transferInAttempt(ce, ent, E) // note different order
36     if Change_tr ≠ ∅
37       Changes := Changes ∪ Change_tr
38   while Col ≠ ∅ and Change_tr ≠ ∅
39   if Col = ∅ // all others incorporated
40     return Changes
41   else
42     undo(Changes)
43     return ∅
```

Algorithm 4.2c. Move of spatial entity: collision-triggered reaction handling.

```
function reactionAttempt(entity ent,move u⃗, colliding spatial entity ce,
    other entities E)
  R_a := ∅
  for r in CollisionRules
    (M,V) :=matchReactionRule(r, surr(ent), ent, ce) // Algorithm 3.3
    if rand_U(0,1) <evaluate(r.rateExpr,V)
      R_a :=R_a ∪ rule
  R_a := randomizeOrder(R_a)
  while R_a ≠ ∅ // try to apply rules
    r := R_a.removeFirst
    Changes := applyChanges(r, M)
    if Changes does not involve degradation of ent or ce
      v⃗ := correctPosReact(ent, u⃗, ce) // resolve overlap from collision
      Change_corr := moveAttempt(ent, v⃗, E) // recursive call
      if Change_corr = ∅
        undo(Changes)
        continue while loop with next rule (if any)
      Changes := Changes ∪ Change_corr
    if r.toCreate = ∅
      return Changes
    else
      Change_create := createEntities(r.toCreate, ce, E)
      if Change_create ≠ ∅
        return Changes ∪ Change_create
      else
        undo(Changes)
  return ∅ // after trying every applicable rule: reaction failed
```

```
function transferInAttempt(entity ent, colliding spatial entity ce, other
    entities E)
  for each r in TransferInRules
    (M,V) :=matchTransferRule(r, ent, ce) // analoguous to Algorithm 3.3
    if rand_U(0,1) <evaluate(r.rateExpr,V)
      Changes := applyChanges(r, M)
      v⃗ := correctPosTransIn(ent, ce) // to move ent completely into ce
      Change_corr := moveAttempt(ent, v⃗, E) // recursive call
      if Change_corr ≠ ∅
        return Changes ∪ Change_corr
      else
        undo(Changes)
  return ∅ // after trying every applicable rule: transfer failed
```

Transfer out attempts are handled analogously. Resolving a collision (`correctPos...`) here means finding a vector in the opposite direction of the recent move that resolves the overlap, while for transfers into (out of) another it means finding a suitable move vector towards (away from) the other entity's center.

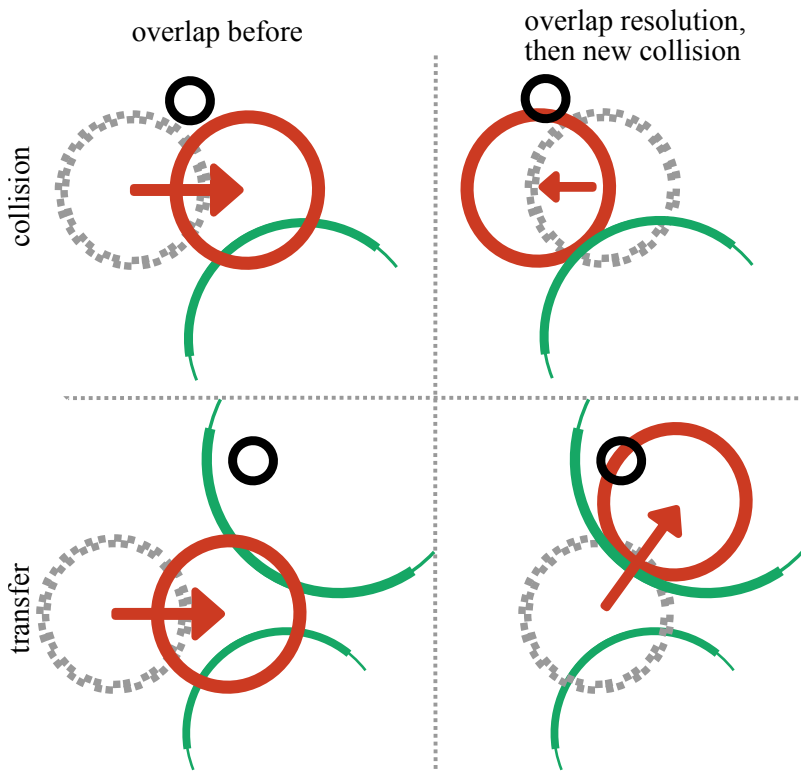


Figure 4.2. Collision resolution after a reaction or transfer may result in collisions with entities that the moving entity did not previously collide with. Hence, the move resolving the original collision triggers a recursive call to the move attempt handling (Algorithm 4.2c, `reactionAttempt` line 13 and `transferInAttempt` line 7).

collisions with just produced entities and (b) the recursive collision resolution may free up space needed to place the new entities.

If a triggered spatial change results in an unresolvable collision, all previous changes in the same step are rolled back and the original move attempt fails (lines 10, 30 and 43). Therefore, information on all changes is kept until successful completion of the step or definitive failure. In hybrid simulation, this "rollback information" is also used for determining consequences on the subvolume level and for correct updating of reaction and move events for all affected entities, i. e. re-evaluation of first-order events of, and zero-order events in, entities whose attributes changed.

4.2.1 Reactive Sites and Reaction Probabilities for Interacting Particles

In actual cells, whether two proteins that can interact actually do so depends on their proximity and their orientation, as such interactions usually depend only on certain

subunits or subsequences of the amino acid chain of each one and the respective regions may not face each other in an appropriate manner for a reaction.

For interactions in well-mixed environments (i. e. in non-spatial or subvolume-based simulation), the likelihood of particles getting close enough *and* having the right orientation for reacting is comprised in the reaction rate constant k. For continuous-coordinate based spatial simulation, when interaction radii and/or interaction probabilities for point-based (dimensionless) or spherical particles are derived from the well-mixed rates, the limited interaction area on each entities' surface is already accounted for, too (cf. section 2.3.6; the derivation of reaction probabilities after collisions indicated there, however, are not directly applicable to ML-Space, where particles have hard boundaries, i. e. are not allowed to come closer than the sum of their radii, and collisions are transient occurrences, i. e. not associated with any time step or duration Δt).

For spherical particles, the probability of interaction when close enough can be considered proportional to the ratio of the reacting regions relative to the total surface of each if the rotational orientation of each is random (unless the reactive regions have to be aligned relative to each other in a certain way). If this proportion is known and rotational diffusion[1] is assumed to be sufficiently fast, the orientation of spherical particles can be ignored in simulation without loss of physical accuracy.[2]

This is no longer true for particles of other shapes: consider a U-shaped particle, where collision with a reactive site in the "valley" is much less likely than one with a side at either end. Entity shapes in ML-Space are so far limited to spheres (circles in 2D) and axis aligned boxes (i. e. cuboids, rectangles in 2D). The latter were needed to allow rectangular system dimensions, but since system dimensions are specified via a model entity that is like all other compartments, cuboids can be used for any of them. However, cuboids in ML-Space, mobile or not, will always have the same orientation throughout the simulation and reactive collisions happen with the same probability no matter where on the cuboid surface the reaction took place. Entities with more elaborate shapes and distinct reactive regions, however, can be represented by spatial entities bound to each other, i. e. complexes.

[1] Throughout this work, the term diffusion is primarily used to refer to translational diffusion, i. e. change of position, not rotational diffusion, i. e. change of orientation.

[2] In fluids with low Reynolds number (which gives the ratio of inertial to viscous or friction forces), the mobility μ of the diffusion coefficient $D = k_B T \mu$ (cf. subsection 2.3.2) is the inverse of the drag coefficient $\zeta = 6\pi\eta r$ for translational diffusion (Stokes-Einstein equation; η is the viscosity of the medium and r the particle's radius) and $\zeta_\mathrm{r} = 8\pi\eta r^3$ for rotational diffusion. With mean squared displacement over time in d dimensions given by $\langle x^2 \rangle = 2dDt$ and mean squared angular deviation for rotational diffusion about a single axis being $\langle \theta^2 \rangle = 2D_\mathrm{r}t$, dividing and canceling terms common in both ζ and ζ_r eventually leads to $\frac{\langle x^2 \rangle}{\langle \theta^2 \rangle} = \frac{4d}{3}r^2$, which indicates that, for example, a particle in three dimensions moves a distance equal to its diameter in the time it rotates by 1 rad on average. For non-idealized conditions, the relationship will be less clear-cut, and additional complexity is added if particles are not exactly spherical – they will rotate faster around axes along which they are longer, meaning that they have different diffusion constants along different dimensions.

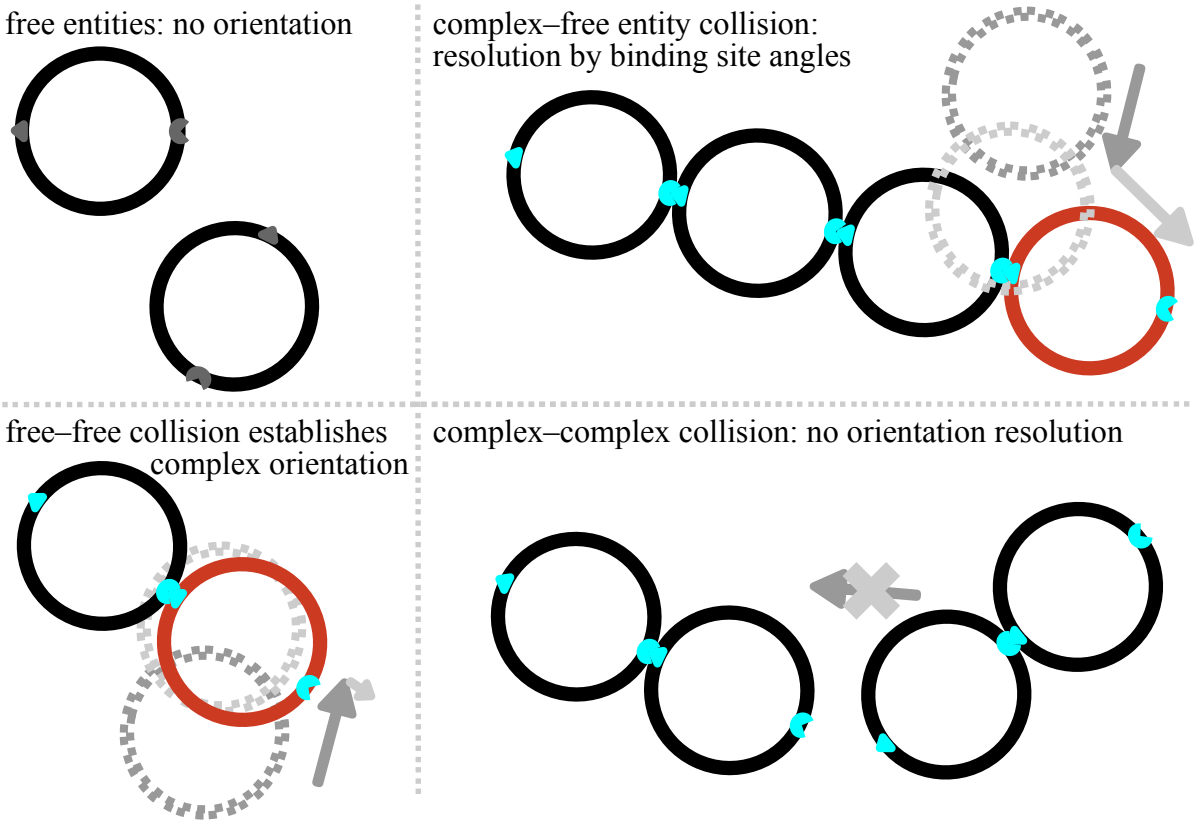


Figure 4.3. Complex formation for entities with two binding sites on opposite sides, e. g., `E<top:0,bottom:180°>`. Gray arrow arrows indicate a move (attempt) of the adjacent entity (old position with darker dotted outline, resulting intermediate position light gray), light gray arrows indicate additional collision resolution.

4.2.2 Entity Complexes / Binding

With explicit bindings between entities, bound entities can still participate in reactions independent of their binding status. However, as there is no object representing the whole complex itself, reactions are local to a subunit of it, i. e. a single entity.

ML-Space' binding capabilities and binding sites with fixed angles were developed for an application involving actin molecules forming long straight chains, *filaments* (section 5.4). Binding sites with angles can be given in the model description, and bindings can be established by collision-triggered reactions (which means that for each entity the occupied binding site and a link to the bound entity will be stored; included, but not explicitly mentioned in Algorithm 4.2c `reactionAttempt` line 10). When the binding entities do not have any other binding partners (i. e. are *free*), they are simply left where they are after the original overlap has been resolved previously. The orientation of the complex is thus established by the direction from which the original collision occurred (Figure 4.3 bottom left).

When a new bond is to be established between a free entity and an entity of the complex, the collision resolution (Algorithm 4.2c `reactionAttempt` line 12) does not move the free entity to the closest non-overlapping position but places it according to the binding site angles: when seen from the center of an entity with multiple binding partners, the angle between the vectors to two bound entities must correspond to the difference of the angles specified for the respective binding sites (Figure 4.3 top right).

For a later application, mitochondrial networks (section 5.3), we used ML-Space' binding capabilities to represent fusion (and fission) of mitochondria, where alignment in fixed angles was not reasonable, so the capability to bind entities at arbitrary angles (in the direction from which they approached) was added. However, in neither case was it necessary to account for mobility of complexes (filaments do not move, and fused mitochondria movement, if existing at all, was neglected). Movement of complexes is possible nevertheless. By default (i. e. unless diffusion constants are explicitly set in rules) a complex' diffusion constant is the reciprocal of the sum of reciprocals of the diffusion attribute values of the entities that make up the complex, i. e.

$$D_{Complex}^{-1} = \sum_{ent \in Complex} ent.\texttt{diffusion}^{-1},$$

which is the expected behavior if diffusion were solely a function of the size (mass) of each particle.

No rotation is implemented so far in ML-Space, i. e. the orientation of a complex does not change. This was appropriate for the filament studies, but limits the usefulness of mobile complexes. Neither is binding between two colliding complexes supported, as consistent handling of these cases may involve changing the orientation of one complex, and hence the position of all its entities, even those not directly part of the reaction-triggering collision. Also, angles (other than 180°, i. e. straight chains) are currently only supported for 2D. Angles in three dimensions would require minor changes in the language's grammar (more than one number is required for specifying them) and some implementation for correct positioning of spheres and possibly other shapes.

However, for detailed studies of complex formation at molecular detail, other approaches will be more suitable, e. g., SRSim (Gruenert et al. 2010) which supports more flexible angles and also takes forces involved into account.

4.2.3 Regions: Soft Entity Boundaries

Compartments, i. e. spatial entities containing other entities, so far have *hard boundaries*, i. e. other entities are either completely inside or outside them. For the same application that brought binding into ML-Space, we also wanted to express spatial areas where different properties hold than in others (specifically, actin filaments growing differently in regions where cells were in contact with a surface structure underneath it than where they were not). Using compartments as before to model these regions was inconvenient, as in the modeled system part there is no membrane or anything corresponding to the

hard boundary that would need to be crossed. We therefore added entities with soft boundaries, called *regions*, which means that other, smaller entities may overlap their boundary after the end of a simulation step. As for the language, this is done via an additional spatial attribute `boundaries` (cf. Table 3.2) with a keyword as value, either `soft` or `hard` (the default used if the attribute is not specified).

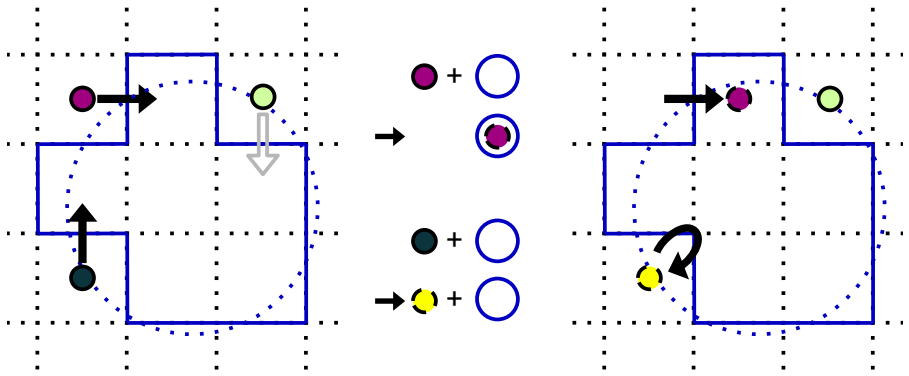
Transfer rules still need to be given to specify whether an entity can move into or out of a region. Transfer attempts, i. e. whether an entity moves out of or into a region (w.r.t. the entity hierarchy), are then determined by whether its *center* moves across the region's boundary. The same criterion is used to determine collisions that may trigger second-order reactions: a moving entity's center must move into the region, it is not enough for an entity to move such that its shape overlaps a part of it (cf. Algorithm 4.2b line 12). After a successful transfer into or out of, or a reaction with a region, no overlap resolution is performed (i. e. Algorithm 4.2c `reactionAttempt` line 12 ff. and `transferInAttempt` line 6 ff. are skipped).

Since the regions were originally devised to represent static areas, support for spatial interactions between is limited. Regions, unlike compartments (hard-bounded entities), may not overlap other regions, unless one is contained within another to begin with. Otherwise, for entities situated in the space where two regions overlap, it would be ambiguous to which they belong. Also, regions will take their content (i. e. all entities with center inside them) with them, which may lead to entities on their boundary colliding with other entities outside them. These will not result in successful reactions as collision-triggered reactions can only be between the current event's moving entity itself and another one, not triggered by an entity that was just moved along.

4.3 Mesoscopic Reaction-Diffusion Simulation with Non-moving Boundaries

One of ML-Space' initial goals was to develop a language that is not limited to a particular spatial simulation method. Our corresponding simulator thus can be used for reaction-diffusion simulation, too. Then, the simulated system (defined by one immobile spatial entity with shape and non-zero size) is subdivided by a regular lattice where each lattice site (subvolume) contains zero to many dimensionless entities (i. e. of size zero). Positions of entities are thus not known exactly in continuous space, only with precision up to subvolume size. Therefore, it does not make sense to distinguish between entities of the same species and with equal attribute values in the same subvolume. Thus, the simulator treats those groups of entities as populations rather than individuals. Diffusion takes place as jumps between neighboring subvolumes and reactions are performed by the classical stochastic simulation algorithm, treating each subvolume as a well-stirred system. This part of the ML-Space simulator was realized in analogy to the Next Sub-

Algorithm 4.3. Diffusion of an entity from a subvolume to a neighbor belonging to a different spatial entity (dotted circle/ solid subvolume outline) can have three different outcomes: transfer (with possible attribute changes for both participants; rule above the illustration's center), reaction (with attribute changes, consumption or replacement of diffusing entity, but no actual change in subvolume; rule indicated below center), or no effect, i. e. effectively a reflective boundary collision.



```
1  function boundaryCollision(origin subvolume sv_o, target sv_t,entity e)
2    if surr(sv_o) = surr(surr(sv_t))
3      R := TransferInRules // e.g. E + S -> S[E]
4      ent_boundary := surr(sv_t)
5    else if surr(surr(sv_o)) = surr(sv_t)
6      R := TransferOutRules // e.g. S[E] -> S + E
7      ent_boundary := surr(sv_o)
8    else return ∅ // no transfer at all if several entity boundaries needed
       to be crossed in the same step
9    for each r in R
10     (M,V) := matchTranferRule(r, ent_boundary, e) // as in Algorithm 3.3
11     if rand_U(0,1) <evaluate(r.rateExpr,V)
12       Change_surr := applyChanges(r, M, ent_boundary)
13       e_changed := applyChanges(r, M, e)
14       sv_o.content := sv_o.content \ {e}
15       sv_t.content := sv_t.content ∪ {e_changed}
16       return Change_surr ∪ {sv_o, sv_t}) // transfer in/out success
17   if surr(sv_o).size >= surr(sv_t).size // try reaction without transfer
18     ce := surr(sv_t) // colliding spatial entity (reaction partner)
19     for each r in CollisionRules
20       (M,V) :=matchReactionRule(r, surr(sv_o), e, ce) // Algorithm 3.3
21       if rand_U(0,1) <evaluate(r.rateExpr,V)
22         Change_surr := applyChanges(r, M, ce)
23         e_changed := applyChanges(r, M, e)
24         sv_o.content := sv_o.content ∪ {e_changed} ∪ createEntities(r.toCreate) \
             {e}
25         return Change_surr ∪ {sv_o}) // reaction success
26   return ∅ // failure: no matching rule found
```

volume method (Elf and Ehrenberg 2004, suppl. methods) with minor changes.[3]

The main addition here is that subvolumes may belong to different biological entities, e. g., the (cytosol of a) cell or nucleus of a cell. In the ML-Space model, these may be defined like spatial (i. e. non-zero size) entities for continuous-space simulation. Each subvolume is then assigned to the spatial entity (called *surrounding entity* subsequently) that includes its centre and which is lowest in the nesting hierarchy. When a diffusion involves two subvolumes, sv_o, sv_t, belonging to different surroundings, the diffusing particle e is considered to collide with their boundary and a reaction, uptake etc. may take place (Algorithm 4.3 `boundaryCollision`; cf. Elf and Ehrenberg 2004, suppl. methods, step 9 b/c).

The size of the lattice must thus be chosen such that the circumscribed sphere (or circle) of each subvolume is smaller than the smallest spatial entity to ensure that at least one subvolume belongs to each entity. For spherical entities, this means that a subvolume's diagonal must be smaller than the respective sphere's diameter (Algorithm 4.1b `initialize` line 6).

In other words, all species in ML-Space must either be modeled as dimensionless entities covered by population-based simulation or as spatial entities larger than a subvolume. There cannot be spatial entities that occupy a part of a single subvolume and take up space otherwise available to the subvolume's population. Direct interaction of spatial and population-based entities happens only through the latters' diffusion attempts.

4.4 Hybrid Spatial Entities and RDME Simulation

The key contribution of ML-Space is a combination of reaction-diffusion simulation with the Brownian motion simulation of spatial, possibly nested entities. Spatial events (moves and first/zero-order reactions of spatial entities) and non-spatial events (i. e. triggered at the subvolume level) are now scheduled in the same event queue. In each simulation step, the next event is dispatched to and handled by the respective part of the simulator.

Moving spatial entities in our approach takes their content with them. This does not only apply to nested spatial entities, but also to the content of the subvolumes that belong to the entity. Therefore, the number of subvolumes belonging to (or overlapped by) a spatial entity should not change by a move. For this, we move spatial entities now only in steps corresponding to the grid size in each dimension (see illustration in Algorithm 4.4a; cf. Algorithm 4.2 `move` line 7) and vary the time between movement events accordingly, e. g., by recording position updates in a "virtual position" in continuous space and changing each entity's actual position only when it deviates by more than half a grid cell side length in any direction.

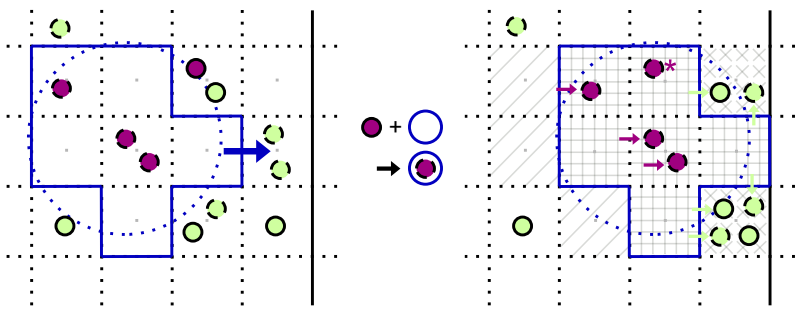
[3] Most notably, a random number $r \in \mathcal{U}[a, b]$ used only for a binary decision (i. e. to test whether it is above a certain threshold, $r > t$ or $r \leq t$) can be rescaled an re-used (i. e. $r_{next} = (r - t)\frac{b-a}{b-t} + a$ or $(r - a)\frac{b-a}{t-a} + a$). This is not done in ML-Space as random number generation was never found to be a significant part of the overall calculation time.

When a spatial entity moves onto a subvolume that did not previously belong to it, this is treated as if *all* non-spatial entities within this subvolume collided with the moving entity (i. e. by invoking `boundaryCollision` from Algorithm 4.3 with effectively the same source and target). Entities of the subvolume that cannot enter or cannot react with the moving spatial entity are "pushed" into the closest neighboring subvolumes that still belong to the original surrounding entity. This is outlined in Algorithm 4.4a. After that, reaction and diffusion events need to be rescheduled for subvolumes in moved spatial entities (marked with grid hatching in Algorithm 4.4a illustration, right), but also those outside into which non-spatial entities were pushed (x-mark hatching) and those from which a spatial entity moved away (diagonal hatching; since these are now empty, only zero-order reactions in their new surrounding entity must be evaluated).

Subvolume events can have only limited effects on the spatial entity level, namely if such an event involves a diffusion attempt between subvolumes with different surrounding entities. If the involved spatial entity's attribute values were changed in the process (which can happen already in the subvolumes-with-fixed-boundaries simulator of the previous section, cf. Algorithm 4.3 line 12), its respective first- and zero-order reaction events must be recalculated. This is done only after all direct effects of an event on the spatial entity and subvolume level have been applied (as shown in the flowchart of Figure 3.2), with the changed model components (spatial entities, subvolumes) being collected on the way (cf. *Changes* in all pseudocode snippets).

The architecture of the ML-Space does not allow subvolume events to trigger, for example, the direct creation of spatial entities, as the part that handles continuous space always comes before the subvolume part when handling a single event. A similar effect like the creation of a spatial entity `S` from dimensionless entities `N1`, `N2` in a context `C`, i. e. `C[N1 + N2]-> C[N1 + N2 + S]@r`, could be achieved with a helper attribute of the context, here `createS`, and an additional, immediate (*confluent*) rule,

Algorithm 4.4a. A move of a spatial entity results in content of newly reached subvolumes being either taken up (if there is an applicable reaction rule – dark filled circle marked * in illustration) or being pushed away (bright cricles/arrows). The content of the spatial entity is moved along (dark arrows).



```
1  function subvolumeEffects(subvolumes SV, spatial Changes)
2    // displacement of NSM entities by newly created entities
3    for each ent in Changes.createdSpatial
4      for sv_o in occupiedSubvolumes(ent, SV)
5        Changes := Changes ∪ pushContentToCloseSVs(sv_o, surr(ent))
6
7    for each (ent, u⃗) in Changes.moves
8      // first, treatment of subvolumes newly occupied by ent
9      SV_o := occupiedSVsBeforeMove(ent, SV, −u⃗)
10     SV_n := occupiedSubvolumes(ent, SV) \ SV_o
11     for each sv_n in SV_n
12       sv_tmp := new sv(sv_n.content, surr(ent)) // dummy sv to store old
             content and previous surrounding entity
13       sv_n.content := ∅
14       for each e in sv_tmp.content // try uptake of previous sv content
15         Changes := Changes ∪ boundaryCollision(sv_tmp, sv_n, e) // code 4.3
16       Changes := Changes ∪ pushContentToCloseSVs(sv_tmp, surr(ent), u⃗)
17     // then, moving content of others along with ent
18     for sv_o in SV_o
19       sv_t := moveTarget(sv_o, u⃗)
20       sv_t.content := sv_t.content ∪ sv_o.content
21       sv_o.content := ∅
22       Changes := Changes ∪ {sv_t}
23
24   // dimensionless entities created by spatial rules
25   for each (e_created, e_near, e_surr) in Changes.createdDimensionless
26     sv_t := random SV in e_surr close to e_near
27     sv_t.content := sv_t.content ∪ {e_created}
28     Changes := Changes ∪ {sv_t}
```

For moving subvolume content along with a moving spatial entity (line 18), ML-Space sorts subvolumes in a queue to start with those subvolumes next to those newly occupied by the entity (line 10), continuing with subvolumes next to these, and so on, to avoid erroneously moving content into subvolumes not yet updated (and thus mixing old and new content). `pushContentToCloseSVs` first finds, for a given subvolume, the closest subvolume(s) in the given surrounding entity and in the given direction, if present as third parameter (where proximity is measured by their center's distance and direction by scalar product of the center-to.center and given vector; still there may be several "close" subvolumes by this definition). Then, it distributes the original subvolume's content equally among these (with random placement of marginal entities).

```
C[N1+N2] -> C(createS+=1)[N1+N2] @ r
C(createS>1)[] -> C(createS-=1)[S] @ Infinity
```

However, the newly created S would be placed randomly inside C, not necessarily near the location of the original reaction, i. e. the subvolume containing the relevant `N1` and `N2`.

4.5 Implementation

4.5.1 Foundations

The ML-Space simulator was implemented as part of the JAMES II framework (Himmelspach and Uhrmacher 2007; Himmelspach 2007; Ewald et al. 2010) in *The Java Programming language*, mostly while 1.7 (a.k.a. Java 7; Gosling et al. 2013) was the current version, but has since been migrated to Java 8. *ANTLR, ANother Tool for Language Recognition, v3* (Parr 2007) was helpful for generating the parser that translates models in the ML-Space language to Java code that serves as input for the simulator.

The JAMES II framework was developed to aid the implementation of new simulation algorithms with focus on proper separation of concerns, here the specification of the model and the actual code of the simulator, and to help facilitate re-use of components needed in many different types of simulation via a plug-in based architecture. There are, as of 2016, more than 100 plug-in types in the core alone, not all relevant to ML-Space. Relevant components include methods to generate random numbers from a given distribution (as pseudo-random number generators usually deliver only numbers uniformly distributed on the interval $[0, 1)$) and event queues, also known as future event lists or sets (Rönngren and Ayani 1997; Brown 1988).

For the simulation execution mechanism of JAMES II, only a method for single simulation steps needs to be implemented in a simulator-specific way, the loop for executing the simulation step-wise and checking whether a stop condition is fulfilled is already part of the framework (cf. Algorithm 4.1a line 4).[4]

4.5.2 Main Architecture

JAMES II provides the basic interface (actually a base class) for the simulator and a whole framework to run simulations. ML-Space consists of implementations of a common base class (see Figure 4.4) for the continuous space simulator, for the subvolume simulator, and for the hybrid simulator that contains one instance of each of the former, acting like a coordinator (similar to Jeschke and Uhrmacher 2008, although the coordinated components there were not made for stand-alone use).

[4] This may sound like it limits JAMES II to discrete-event or discrete step-wise formalisms, but it should be noted that, for example, numerically solving differential equations also involves small, discrete steps for approximating points on a continuous curve that can usually not be given explicitly.

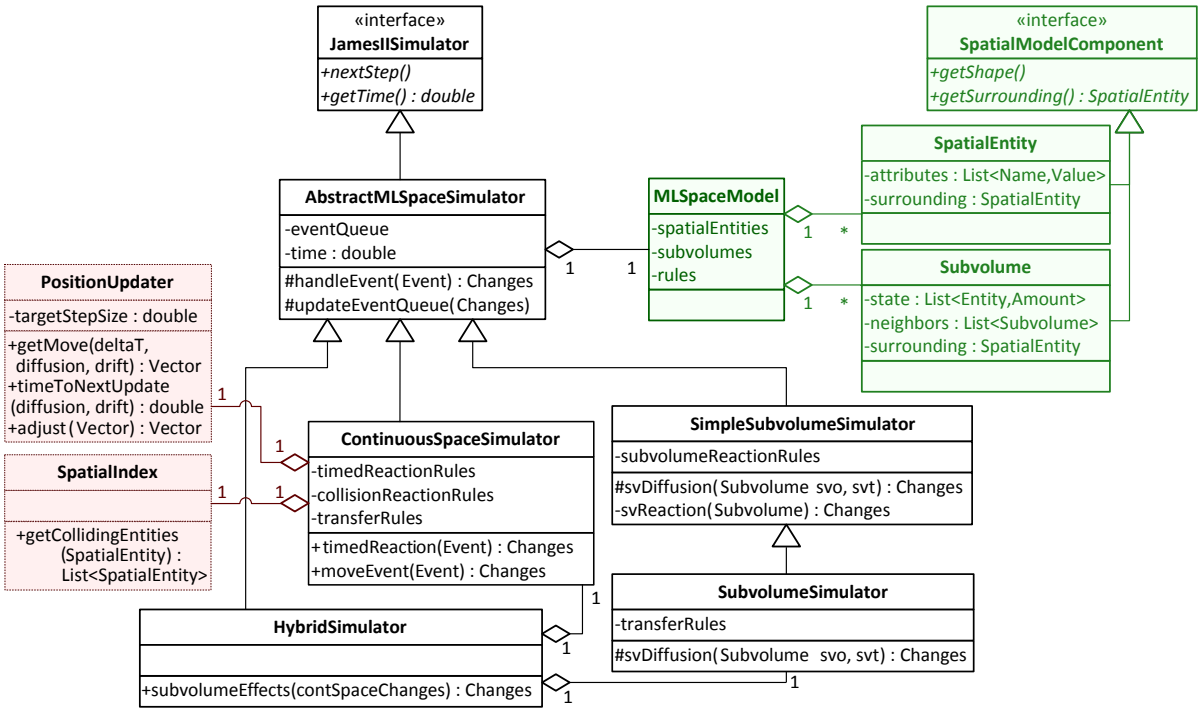


Figure 4.4. UML diagram of the main components of ML-Space' implementation (simplified). Non-shaded (white): classes of the simulator implementation. Shaded (red) left: plug-ins, i. e. exchangeable subunits of a part of the simulator. Shaded (green) top right: classes related to the model representation.

The subvolume simulator is actually implemented via two classes, one simply reproducing the Next Subvolume method of Elf and Ehrenberg (2004) ignoring spatial entities, and one that overrides the diffusion handling method, including a check whether the involved subvolumes belong to different spatial entities and handling it according to Algorithm 4.3. (In the actual code, the continuous space simulator is split into two classes as well, one handling zero- and first-order reactions and event scheduling, and one extending this handling the actual spatial aspects, but this separation was mainly made to reduce code complexity per class, not from a conceptual point related to the simulation algorithm. Also, for historic reasons, the term *processor* is used synonymously with *simulator* in JAMES II, including for ML-Space, unlike shown in Figure 4.4).

The hybrid simulator is *composed* of a continuous-space and a subvolume simulator, i. e. contains an instance of each. This pattern allows for exchange of each part, e. g., replacing the current subvolume simulator by one employing spatial τ-leaping (Jeschke, Ewald, and Uhrmacher 2011) or by a (deterministic) PDE-approximating scheme (e. g., as in Kossow et al. 2015).

Observation, e. g., recording of entity count trajectories or complex sizes, but also graphical output, is handled via separate classes that are notified of changes by the sim-

ulator after the simulation step has completed (i. e., here, passed the *Changes* structure created in the respective step).

4.5.3 Plug-Ins: Customizable Simulator Parts

Several tasks to be executed during the simulation can be performed in different ways. To accommodate this, JAMES II contains a plug-in and dependency injection system as well as implementations for tasks common to many different simulation approaches, e. g., generation of (pseudo-) random numbers for stochastic simulation or transformation of these random numbers, customarily uniformly distributed on $[0, 1)$, to a different random distribution, e. g., the Gaussian distribution (see Table 4.1; for other transformations, e. g., from $\mathcal{U}(0, 1)$ to $\mathcal{U}(a, b)$ or to an exponential distribution, there is only one reasonable way).

Event Queues

As mentioned, ML-Space schedules event using a priority queue data structure, a.k.a. future event list/set or queue. It stores elements (events) along with a numerical value (priority, here: time stamp) and allows efficient retrieval of the highest priority (least time stamp) element (dequeue). JAMES II offers a wide variety of event queue implementations (most of them developed when its focus was still on simulating DEVS models). While some versions that can run dequeue and subsequent enqueue operations in amortized constant time ($O(1)$, cf. Brown 1988), the rescheduling of existing events (requeue) that we usually need with each collision and subvolume diffusion event is in $O(\log n)$ for requeue-optimized implementations (and in $O(n)$ otherwise, where n is the number of contained events; requeue-optimization here usually means adding an inverse lookup from events to their previously scheduled time to subsequently remove them from whichever data structure they are stored in). In our ML-Space experiments, we saw little difference in simulation speed when exchanging the event queue as long as it was requeue-optimized. In particular, we do not see a significant performance difference between event queues with amortized $O(1)$ dequeue and those with $O(\log n)$ dequeue operations.

Position Updates

The handling of vectors, i. e. positions and displacements, to make the simulator implementation independent of whether the actual model uses two- or three-dimensional coordinates and whether periodic boundary conditions are used or not, is another aspects for which generalized interfaces with several implementations exist (but here, different implementations for different behavior).

ML-Space employs the *strategy pattern*, i. e. different implementations for achieving one kind of behavior, for random vectors, i. e. move steps in ML-Space, for example. Creating d-dimensional vectors with lengths following a given normal distribution and

Table 4.1. Selected JAMES II plug-in types and implementations relevant for ML-Space simulation. Implementations selected by default (and used for most applications of chapter 5) in bold. The first three types are part of the JAMES II framework, the latter two are specific to ML-Space.

Plug-in type	Implementation	Reference/Remarks
(Pseudo-) random number generator	**Mersenne Twister**	Matsumoto and Nishimura (1998)
	Linear Congruential	fast, long established, but not very good (Park and Miller 1988), Java default
	... many more, not all included public version of JAMES II e. g., WELL (Panneton, L'Ecuyer, and Matsumoto 2006)	
Normal (Gaussian) distribution	**Polar method**	Marsaglia and Bray (1964)
	Box-Muller	Box and Muller (1958)
	(CDF inversion)	(approximating Φ^{-1} is either inaccurate or slow, cf. Thomas et al. 2007)
Event queue	**Calendar queue**	Brown (1988, requeue-optimized version)
	TreeMap-based	own impl., sometimes marginally faster
	Priority queue	Java impl., not requeue-optimized
	... many more, not all included public version of JAMES II e. g., MList (Goh and Thng 2003)	
Position updater	**scaled uniform** (continuous space)	$x_1 \dots x_d \in \mathcal{U}(-1, 1)$, vector scaled to length from $\mathcal{N}(0, 2dD\Delta t)$
	independent (cont.)	$x_1 \dots x_d \in \mathcal{N}(0, 2D\Delta t)$
	polar/spherical (continuous space)	length from $\mathcal{N}(0, 2dD\Delta t)$, angle from $\mathcal{U}(0, 2\pi)$ (if $d = 3$, another angle $\phi \sim \cos^{-1}\mathcal{U}(-1, 1)$), conversion to Cartesian coordinates
	discrete step *(hybrid)*	steps in line with subvolume grid size, e. g., by wrapping a continuous space updater and applying updates only when a threshold distance is exceeded
Spatial index	**uniform grid**	(see text)

random direction (cf. subsection 2.3.2) can be achieved generating d normally distributed (pseudo-) random numbers or d uniform and one normal random number, possibly at different computational costs, although in practice we find the time difference to be in the single digits percentages.

The continuous-space simulator updates entity positions in fixed time steps by default. Determining a reasonable time step, i. e. the time in which a particle with given diffusion and drift (omitted in the last column of Table 4.1) would move a pre-defined distance on average, is also a responsibility of the position update method. This is an extension point that will be relevant if one wants to extend ML-Space by GFRD-like behavior, where the position updates happen not in fixed-time, same-mean-displacement steps but are flexible with distance and time depending on the proximity of reaction partners (van Zon and ten Wolde 2005a, see also subsection 2.3.2). It also allows us to vary the (then no longer fixed) time steps slightly to avoid many particles with the same diffusion constant moving at the exact same time steps, alternating with groups of particles with a different diffusion constant.

In hybrid simulation, on the other hand, spatial entities move only in steps corresponding to the subvolume grid. Without this, the amount of subvolumes belonging to an entity (determined by whether the entity's shape includes a subvolume's center) could change due to a move, requiring special handling of the dimensionless entities contained in the spatial entity (if at all possible in a realistic way). Discrete steps are achieved by internally keeping track of an entity's continuous-space position but returning only discrete steps to the simulator, which may mean null vector "updates" as long as an entity does not stray too much from its previous position. This approach requires us to also discretize vectors generated by the simulator, e. g., those for resolving collisions (Algorithm 4.2a line 12), i. e. *adjust* it (which, for the continuous-space position updaters, is the identity operation).

An unfortunate but inevitable result of this adjustment is that the hybrid simulator and bindings between spatial entities do not go well together, as it may introduce small gaps between the (continuous-space) shapes of entities whose overlapped subvolumes partially border each other. It will also interfere with correct binding angles as the grid-related adjustment will very rarely be in the direction required of the binding-angle-related correction determined by the continuous-space simulator.

Collision Detection

Finally, a component crucial for simulation speed is the collision detection. The implementation of a collision check for two concrete entities is the responsibility of the `Shape` class (cf. `SpatialModelComponent` in Figure 4.4), which is straight-forward for two spheres (distance of centers $<$ sum of radii) or box-box interactions and follows well-established methods for box-sphere checks (Arvo 1990). When adding further shapes to the implementation (e. g., ovals, cylinders/rods) it will be necessary to add collision tests for all already implemented shape types. A naïve approach of pairwise collision checks

between all n entities on one organizational level has time complexity $O(n)$ *per move of an entity* and thus does not scale well with the number of particles. *Spatial indexing* or *spatial partitioning* methods are an important field of research in computer graphics and game development and can help here as well. While ML-Space also contains an abstraction (i. e. an interface) that allows implementation of different approaches, so far there is only one: a grid-based approach, using a static, implicit, uniform grid (cf. Ericson 2004, ch. 7.1; not to be confused with the subvolume grid). The continuous space here is subdivided into a number of roughly square/cubic grid cells and for each grid cell, the overlapping entities are recorded. Collision checks then need to be performed only between entities in the grid cell(s) of the entity last moved or added. (Our rigid-body, positive-size entities can belong to several grid cells when they are on a boundary – up to 2^d in d dimensions if smaller than box size. Smoldyn – Andrews and Bray 2004; Andrews 2012 – which uses point-based, dimensionless entities, also employs a grid for detection of particles closer than their respective interaction radius but always needs to check for interaction partners in all neighboring cells, and if an interaction radius exceeds the cell size, some interactions may be missed.)

The collision check effort for each move can be reduced to $\lesssim O(\sqrt{n})$ with this approach if the entity density and sizes are not too inhomogeneous. To see this, take a grid of $\gtrsim \sqrt{n}$ cells and consider how many entities each will contain. With a smaller grid size, chosen such that there is a small constant maximum number of entities in each cell, collision detection for a single moving or created entity can theoretically be performed in constant time ($O(1)$), but the overhead for spatial entities located on the boundary of grid cells increases, especially when there can be larger entities that can fill multiple cells entirely.

Methods that may address inefficiencies of the above collision detection approach for wildly inhomogeneous entity densities or offer general speedup include dynamic grids (quad-/octrees) or BSP-trees (Ericson 2004) or space-filling curves (e. g., Gargantini 1982; Hale and Youngblood 2014), but while micro-optimizations of this part of the simulator are an interesting research sub-topic, they were not a priority so far.

4.5.4 Experimental Framework and Simulation Setup

The aforementioned simulator and its components are the means to simulate an ML-Space model one single time. When running a simulation multiple times with the aim of performing model parameter scans, component tests (e. g., to determine the effect of the chosen spatial step size on the results) or simply averaging multiple simulation run results – always an important aspect in stochastic approaches – we speak of a simulation experiment.

Parts of what Kappa and BioNetGen include in the model is comprised in the experiment setup here: the observables. The observation mechanism in JAMES II is relatively independent of the simulator, which would make it less convenient to intertwine observables with the model in the way the former languages do, where a pattern to observe can be specified and the amount of entities matching the pattern can be used during model

Listing 4.1. Example experiment configuration file for the gene regulary network model of Listing 3.1, varying one model parameter (diffusion `D`) and one simulator parameter (subvolume side length l_{sv}, cf. Algorithm 4.1b) for a total of 6 simulation runs.

```
[Overridden model variables]
; Variable name = comma-separated list of values
D = 0.01,0.6

[Simulation parameters]
parallelthreads = 2
simulationendtime = 1200
subvolumesize = 0.25,0.5,1

[Observation]
observationtargets = Protein,mRNA IN Nucleus,Cell;Gene=site
; ^ count proteins in nucleus, proteins in cell, mRNA in nucleus and
; mRNA in cell; also record attribute value of "site" for all genes
; every ... sim time units:
snapshotinterval = 2
outputdirectory = Hes1sim
```

simulation directly, e. g., in reaction rate expressions.

The experiment setup also contains settings for the simulator, e. g., target step size for continuous-space position updates and subvolume side lengths. One can also run parameter scans by associating constants specified in a model with one or more values (possibly) different from the one in the model. Doing this for several parameter values will result in all combinations of these parameters being simulated (*full factorial*). (In BNG, this is part of the *actions* also usually included in the model file.)

The simulation setup can be specified in a JAMES II-specific Java file. However, this is somewhat cumbersome and probably unintuitive for ML-Space users without a computational background. The former, but not so much the latter is addressed by SESSL (Simulation Experiment Specification via a Scala Layer; Ewald and Uhrmacher 2014), which aims to generalize setups for various simulation tools. Limited SESSL bindings for ML-Space were used only briefly for parameter optimization experiments. (ML-Rules's SESSL interface is significantly more advanced.) Instead, the current ML-Space simulator accepts a simple configuration file (according to the somewhat standard `.ini/cfg` format, see Listing 4.1 for an example) or the same key-value pairs as command line arguments. (The downloadable version contains a short manual, and the tool displays a list of options when run without argument.)

5 Applications

The initial version of an ML-Space simulator was tested with a model of receptor proteins on a membrane and lipid rafts, which are glycolipoprotein microdomains of the membrane in which proteins may behave differently (Bittig et al. 2011; with subvolume and continuous-space capabilities but not yet hybrid in the current sense).

The first novel biological application were investigations of actin filament formation, specifically which assumptions can explain or contribute to an explanation of different actin filament patterns when growing osteoblasts (bone cells) on titanium surfaces with different patterns (Bittig et al. 2012; Bittig et al. 2014a). For this, explicit bindings sites were added to the ML-Space language (subsection 3.2.5) and handling of fixed binding angles to the simulator (subsection 4.2.2).

These binding capabilities also proved useful in a subsequent application on mitochondrial networks (Bittig et al. 2014b). Here, we build on an existing "'agent-based" model (not following a specific formalism; Patel, Shirihai, and Huang 2013) where mitochondria are abstracted to round individuals that are mobile and form "networks" simply by staying in proximity after collisions, extending it by regulatory proteins relevant for the breaking of these network "bonds".

For our more technical publication on ML-Space (Bittig and Uhrmacher 2016), finally, we extended the aforementioned lipid raft model into a three-dimensional abstract model of proteins and large organelles that slow these proteins down and "sweep" their surrounding when they move. We use this simple model for comparing results and performance of a purely individual-based (continuous space) and a hybrid approach. Also for that publication, we reproduced an established spatial gene regulatory network model, the Hes1 oscillator already used as example in Listing 3.1, achieving broadly similar results to the original RDME-based simulations both in our continuous-space and subvolume-based simulator.

This chapter contains these applications in reverse chronological order, with each section based on (and quoting liberally, mostly without further indication, from) the respective original publication.

5.1 Reproducing an Existing Model: Hes1 Gene Regulatory Network Oscillations

5.1.1 Background

Proteins are produced in the cytosol by translation of an mRNA. mRNA is produced by transcription of DNA in the nucleus. DNA transcription may be regulated by proteins binding near the region to be transcribed, at a promoter site.

Hes1 is an example of a protein that can bind at the promoter site of its own gene, effectively suppressing its own production. Expression levels of Hes1 have been found to oscillate with two hour periodicity in a variety of cell types (Hirata et al. 2002). Malfunction in the Hes1 feedback loop has been implicated in cancer development (Sang, Coller, and Roberts 2008).

Previous non-spatial simulation efforts targeting the Hes1 network, stochastic as well as deterministic, relied on explicitly introduced delays to reproduce the oscillatory behavior. Sturrock et al. (2013) were able to produce oscillations with a spatial model relying on mass-action kinetics only, without delays, obtained with URDME (Drawert, Engblom, and Hellander 2012), a Next Subvolume method implementation on unstructured meshes (which itself is freely available, but requires commercial software – MATLAB® and COMSOL Multiphysics® – to run).

5.1.2 Model and Simulation Setup

The ML-Space representation of the model and its reactions were already used as example in the language chapter – see Listing 3.1 (page 41).

Since the only model entities that can contain other entities, cell and nucleus, are not mobile here, there is no need to use the full hybrid ML-Space simulator (meaning we use either subvolume-based simulation with proteins and mRNA as dimensionless entities in subvolumes belonging to different fixed compartments or continuous-space with proteins and mRNA as mobile individuals). For the sole second-order reaction (protein binding to promoter/gene), the bulk reaction rate was converted to a probability for continuous-space simulation by dividing by the diffusion-limited reaction rate (cf. section 2.3.6) calculated with the proteins' diffusion constant (the gene is immobile) and the sum of arbitrarily chosen protein and gene radii, to which our results were not sensitive. Sturrock et al. (2013) had already found their results to be insensitive to changes to the respective binding rate parameter.

5.1.3 Results

Our ML-Space model can reproduce the original results that the simulated cells show oscillations of differing period (see Figure 5.1) and amplitude. This is consistently observed both in continuous-space and subvolume-based simulations.

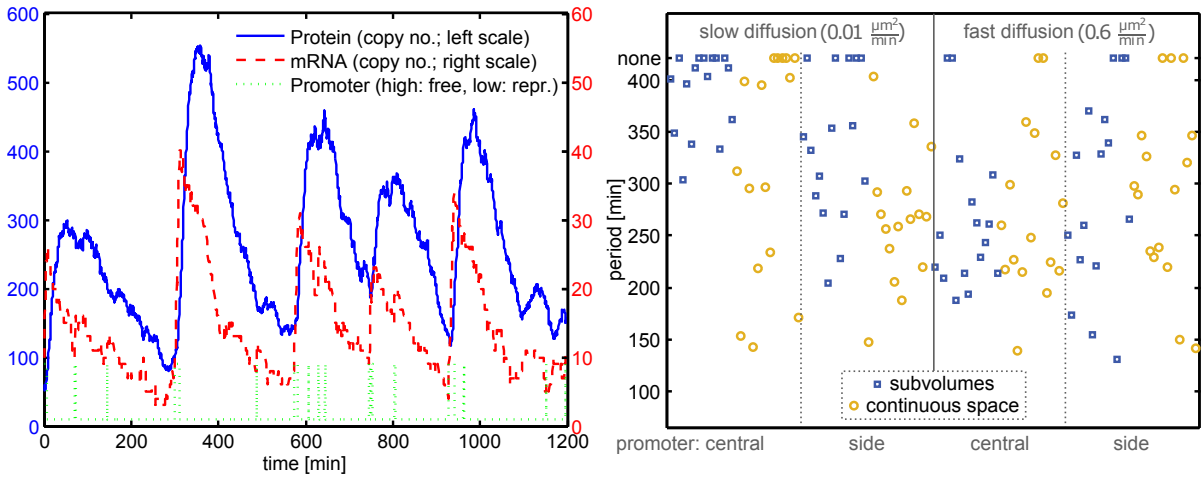


Figure 5.1. Left: Trajectories for a single simulation run (promoter state changes that lasted <0.5 min not included). Right: Periods (of protein numbers) of 128 trajectories, calculated with a Morlet continuous wavelet transform with Gaussian edge elimination using WAVOS (Harang, Bonnet, and Petzold 2012) as in Sturrock et al. (2013, cf. fig. 5 and 4, respectively).

We can also reproduce the dependence of the oscillations on the diffusion coefficient (with slow diffusion, more simulation runs exhibit no stable oscillations – also for continuous-space, side-promoter simulation, even although none happen to be present in the runs randomly selected for Figure 5.1; while the mean period of slow-diffusion runs that exhibit oscillations is higher than those from experiments with fast diffusion, the difference is within either standard deviation) and that the oscillations are robust to changes in the promoter position (which can be far from the nuclear membrane and thus close to the nucleus' center, or near a side of it and thus near the membrane).

5.2 Hybrid and Multi-level: Lipid Rafts and Sweeping Organelles

5.2.1 Biological Motivation

To compare the simulation approaches covered by ML-Space with a model including dynamic compartments, we consider proteins in three dimensions and mobile organelles that can enclose them and slow them down. The example was inspired by lipid rafts on membrane surfaces (Haack et al. 2013; Haack et al. 2015).

5.2.2 The Model

The model, shown in Listing 5.1, consists of only three species (including one to specify the surrounding system) and two rules, for transfer of proteins into and out of the organelles, where the former's diffusion attribute is modified (by a multiplicative factor from $[0, 1]$, which we simply call "slowdown", corresponding to the ratio of protein diffusion coefficient inside to their diffusion coefficient outside organelles). We initialize the model with all proteins outside organelles and observe the ratio that ends up inside organelles, which depends mostly on how much proteins are slowed down in organelles: For no slowdown (factor 1), the ratio should correspond to the fraction of volume covered by organelles. With actual slowdown (factor < 1) the organelles should "catch" a higher fraction of proteins.

Listing 5.1. Sweeping organelles example model.

```
x=100; y=80; z=60; // arbitrary units, e.g. micro-m for length ...
sizeProt = 4/3*PI; diffProt = 2; // ... and min for time
sizeOrg = 4000; diffOrg = diffProt*(sizeProt/sizeOrg)^(1/3);

protAmount = 1000;
slowdown = 0.25; // factor from [0,1]

orgCov = 0.2; // organelle coverage, i.e. ratio of volume
orgAmount = x*y*z*orgCov/sizeOrg;

Cell(diffusion:0, size:x*y*z, shape:cuboid, aspectratio:(x,y,z));
Organelle(shape:sphere, size:sizeOrg, diffusion:diffOrg);
Protein(shape:sphere, size:sizeProt, diffusion:diffProt);

1 Cell(position:(0,0,0)) [protAmount Protein + orgAmount Organelle ];

Protein + Organelle -> Organelle [Protein(diffusion*=slowdown)] @ p=1
Organelle[Protein] -> Protein (diffusion/=slowdown)+Organelle @ p=1
```

5.2.3 Results

We indeed find that the ratio of proteins outside vs. proteins inside organelles approaches the ratio of space outside vs. inside organelles multiplied by the slowdown factor. Put differently, when s is the slowdown factor and c the fraction of space covered by organelles, the ratio of proteins inside organelles (relative to total) is $1/(1 + s(\frac{1}{c} - 1))$. Figure 5.2 contains example time courses for simulations with selected model parameter combinations. With strong slowdown, reaching the predicted steady state value (single marker at $t > 200$) requires more time than shown.

We also find that with mobile organelles, the ratio of proteins inside organelles is

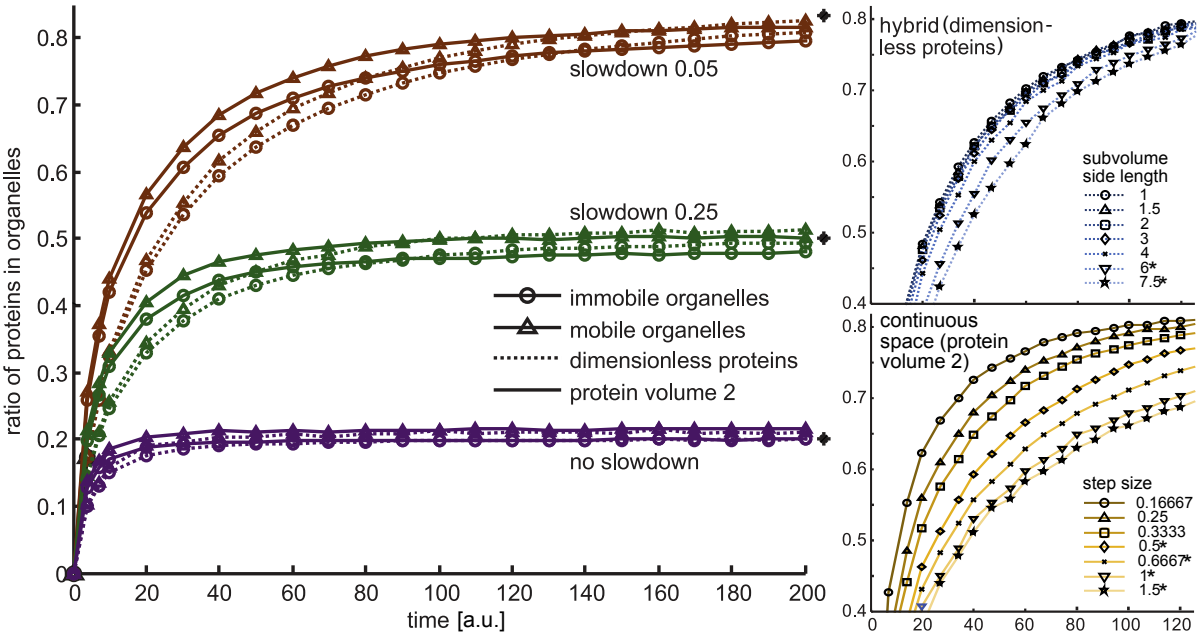


Figure 5.2. Left/main: Uptake of proteins by organelles with different slowdown values. Averages of 84-140 simulation runs each. Theoretical steady state values (for immobile organelles) indicated by black marker after the curves. Right: Enlarged section of the curves for slowdown 0.05 and mobile organelles when distinguishing different spatial resolutions (subvolume side length or mean movement length, respectively). Finer resolutions lead to faster uptake (curves more to the left), an effect that is less pronounced in hybrid simulation (where the curves for the lower values almost coincide). Runs marked with asterisks (*; lower granularity) were excluded from calculation of the average trajectories in the main plot.

slightly larger than with fixed organelles. These simulation results are similar for hybrid and continuous-space-only simulation with otherwise equal model parameters. However, in both cases the granularity of the simulation (subvolume size or move step size, respectively) makes a difference regarding how fast the steady-state ratio of proteins in organelles is approached (Figure 5.2 bottom).

We can explain the difference in continuous space by two considerations. First, the number of collisions between proteins that recently entered the organelles with proteins trying to do the same will be higher. This is because the simulator tries to move all spatial entities in steps of the same average length, so with slowdown in the model (factor < 1), the intervals between updates of (slower) proteins inside organelles are higher than of those (faster ones) outside organelles. At larger step sizes (and thus longer intervals between position updates), proteins that just entered organelles stay near the boundary for longer before eventually moving (in larger steps) further inside the organelles or out again, potentially blocking others in the meantime. Second, if a Brownian trajectory is sampled in larger steps, a brief crossing of a boundary may be missed or a permanent

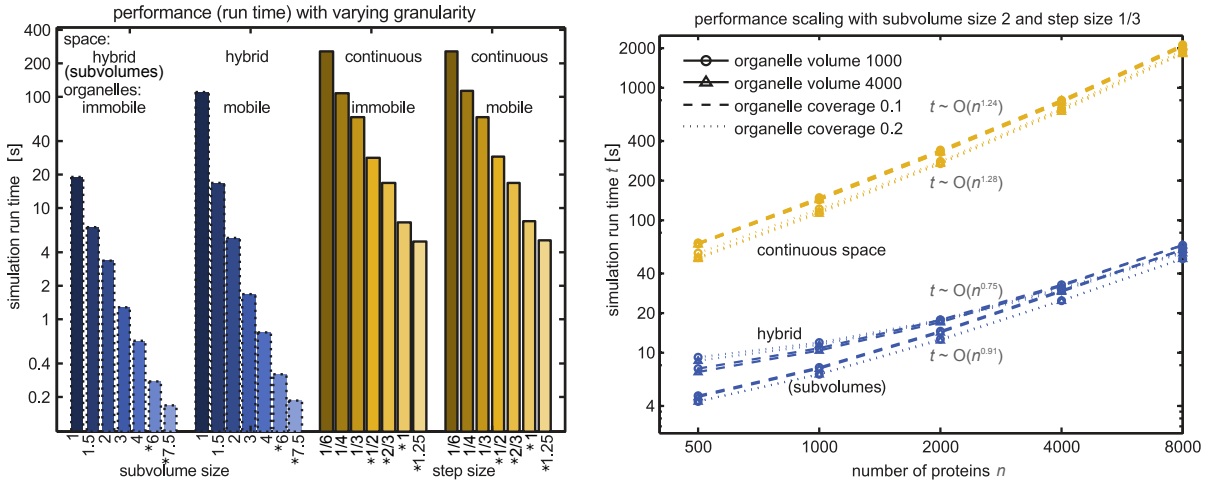


Figure 5.3. Performance experiments using the subvolume-based/hybrid simulator or continuous-space simulation. Left: Varying resolutions (cf. Figure 5.2). Right: Varying the number of simulated entities. Note the logarithmic scale of the quantitative axes. Implied time complexities $O(n^c)$ were calculated from a linear fit of the log-log data points. Model parameters as in Listing 5.1 unless otherwise noted, model time at end of simulation 300 (left) and 600 (right) arbitrary units. Proportions are similar for slowdown values other than 0.25. Performed on an Intel Core i7 X990 3.47GHz with 6 cores (Hyperthreading off, 5 parallel threads used), Windows 7 SP1, Java 7u65. left: 28 replications per bar. right: 4 replications per point.

crossing may be detected later compared to a finer sampling. However, while a fine sampling may seem advantageous in general, in this model the sampling can be *too* fine, since proteins are slowed down quite suddenly on entering the organelle (an unrealistic proposition for a boundary that can be crossed with $p = 1$) and then the next step, which includes the possibility of immediately leaving the organelle again, will be made much later, increasing the time proteins stay inside the organelle.

Whereas the curves for subvolume simulation seem to converge for smaller side length, the continuous space step length has not been decreased far enough to see a similar effect. While this is indeed possible, the curves for the finest two step sizes are above curve for the "best" subvolume run (i. e. protein uptake is faster), indicating the aforementioned effect is already at play here.

As for performance, in continuous space there is almost no difference in run time of simulations with mobile and immobile organelles, whereas the hybrid simulations with mobile organelles are computationally more expensive than the ones where organelles are fixed, i.e. each subvolume belongs to the same spatial entity over the whole course of the simulation (compare left two groups in Figure 5.3, left). Still hybrid simulation is much faster than continuous-space simulation except when comparing a very fine subvolume grid with large step sizes in continuous space (note that proteins with radius 1

in continuous space have roughly the volume of a subvolume with side length 1.6).

Performance Discussion

Our experiments with the sweeping organelles example model show that performance scales sub-linearly with the number of model entities for subvolume-based simulations (if the number of dynamic compartments is constant in the hybrid case) and at least sub-quadratically for continuous-space simulations with the same spatial resolution. The performance-relevant simulator components here are the event queue, as mentioned previously, and, for continuous space, the collision check.

For continuous-space simulations, the number of scheduled events increases linearly with the number of entities, whereas, more important for performance, the effort for handling collisions increases in the order of $\lesssim O(n^{1.5})$ (section 4.5.3: Collision Detection).

For subvolume-based simulations with the same amount of subvolumes, increasing the total number of entities increases the number of entities per subvolume and the number of events per time interval proportionally (but the number of total scheduled events stays constant). Thus, time complexity of the main simulator part is at most linear in the number of entities. Note that approximate methods like τ-leaping (cf. section 2.2.2; spatially extended by Iyengar, Harris, and Clancy 2010 and Jeschke, Ewald, and Uhrmacher 2011) are particularly suitable when particle numbers are large, because then larger approximation steps can be taken.

5.3 Individual-based with Bindings: Mitochondrial Fission

5.3.1 Background

Mitochondria are mobile organelles that exist in living cells as a tubular network. They continuously join the mitochondrial network by fusion and divide by fission events. Mitochondrial fission is mainly regulated by two nuclear-encoded proteins, fission protein 1 (Fis1) and dynamin related protein 1 (Drp1). Mitochondrial dynamics have been shown to be an essential quality control mechanism in order to maintain mitochondrial health. A proxy for mitochondrial health and integrity is the mitochondrial membrane potential (Schmitt, Lenzen, and Baltrusch 2011). Recent wet-lab studies have shown that the mitochondrial membrane potential is disturbed by an imbalance of the mitochondrial fission proteins. It is therefore the objective of this study to develop an in silico prediction model for the influence of Fis1 and Drp1 on mitochondrial spatial structure and health.

We here take an existing model of mitochondrial health maintenance (Patel, Shirihai, and Huang 2013), where mitochondria move in a random direction for random intervals of time (i. e. along not explicitly included microtubules) in an otherwise (for purposes of the model) empty 2D cell. Abstract health units mimic the functional state of mitochondria by representing the membrane potential. Mitochondrial fusion allows mitochondria to exchange components (here: health units) in order to maintain health.

5.3.2 Model and Simulation Setup

Mitochondrial fusion is described in our ML-Space model as binding with probability 1 on collision of moving mitochondria, including exchange of two health units. Fission is a first-order reaction with a given rate. Fused mitochondria become immobile here. For simplicity, checks against exceeding the minimum and maximum for the health attribute's value are omitted here. See appendix A.2 for the full model including rate parameter values.

```
1 Mito()<bs:free> + Mito()<bs:free> -> Mito(velocity=0,health-=2)
      <bs:bind>.Mito(velocity=0,health+=2)<bs:bind> @ p=1
2 Mito()<bs:Mito()> -> Mito()<bs:release> @ rFission
```

Further rules include damage (a first-order reaction lowering health of a mitochondrion), autophagy (consumption of unhealthy mitochondria) and replication (creation of a new, healthy mitochondrion, keeping the total number roughly constant.) We can reproduce basic findings of the original (Patel, Shirihai, and Huang 2013) despite some imprecision in the original model description, e. g., whether fused mitochondria are also subject to damage and possibly autophagy.

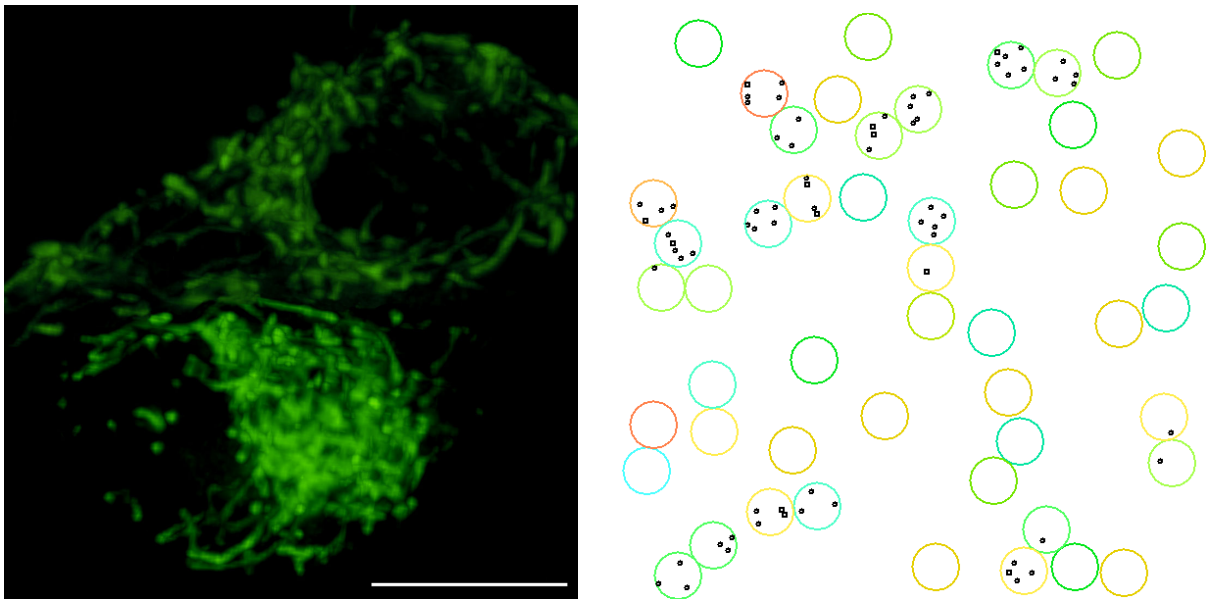


Figure 5.4. Left: Microscopic image of the mitochondrial network in a glucose-responsive MIN6 beta cell. Scale bar 20 µm. Right: Simulation screenshot (cyan/-green: healthy mitochondria, red: damaged; tiny circles/squares: recruited Fis1/Drp1 molecules of fused mitochondria).

More complex variations could include allowing transfers only from the less healthy to the more healthy mitochondrion (a damaged mitochondrion "sacrificing" itself; first reaction below) or only from the healthier to the less healthy one (a "healing" of the less strong; second reaction below; it is not known which proposition is realistic biologically).

```
3 Mito(h=health>=3)<bs:free> + Mito(health in [h...8])<bs:free> ->
      Mito(velocity=0,health-=2)<bs:bind>.Mito(velocity=0,health+=2)<bs:bind>
4 Mito(h=health)<bs:free> + Mito(health in [3...min(h,8)])<bs:free> ->
      Mito(velocity=0,health-=2)<bs:bind>.Mito(velocity=0,health+=2)<bs:bind>
```

The maximum number of health units is 10 in this model, thus with 2 health units exchanged in each fusion reaction, 8 is the maximum attribute value for being on the receiving end of a health unit exchange (3 is the minimum for participation in health exchange and also the threshold for autophagy).

We modified this model by representing the number of bound Fis1 and Drp1 molecules of mitochondria as attributes (see also Figure 5.4) as follows.

```
5 Mito(f=nFis<maxFis)<bs:Mito(nFis<8-f)> -> Mito(nFis+=1) @ rFisRecruit
6 Mito(f=nFis,d:=nDrp)<bs:Mito(nFis>=4-f,nDrp<2-d)> -> Mito(nDrp+=1) @ ...
```

Only fused mitochondria facilitate Fis1 and Drp1 recruitment here and the number of Fis1 (Drp1) per mitochondria pair cannot exceed the fission threshold 8 (2). Addition-

ally, we only allow Drp1 recruitment when a certain number of Fis1 is already bound (above: 4). The actual fission rule becomes more complex then:

```
7 Mito(f=nFis,d=nDrp)<bs:Mito(nFis>=8-f,nDrp>=2-d)>
      -> Mito(nFis=0,nDrp=0)<bs:release> @ Infinity
8 Mito(nFis>0)<bs:free> -> Mito(nFis=0) @ Infinity
9 Mito(nDrp>0)<bs:free> -> Mito(nDrp=0) @ Infinity
```

The nFis and nDrp attributes of the first mitochondrion are set to 0 in the process, indicating release of the previously bound Fis1 and Drp1. The final two reactions are needed to "release" the remaining Fis1 and Drp1 entities from the formerly fused mitochondrion.

With the above rules, Fis1 and Drp1 are essentially treated as ubiquitous and the recruitment rate constant choices are the only limitations to fusion. We also simulated a slightly more realistic model where Fis1 and Drp1 numbers are limited (but constant) and recruitment thus happens slower if there are already many fused mitochondria that have some Fis1 and/or Drp1 bound, but not enough for a fission event. This can be expressed ML-Space by attribute values for Fis1 and Drp1 numbers in the surrounding entity, i.e. the cell, and rules where the context (and its attributes) are explicitly mentioned (omitted due to spatial constrains).

5.3.3 Results and Outlook

Related wet-lab experiments have shown that cells with reduced Fis1 or Drp1 expressions exhibited a significantly lower membrane potential and a heterogenic mitochondrial network (Reinhardt et al. 2013).

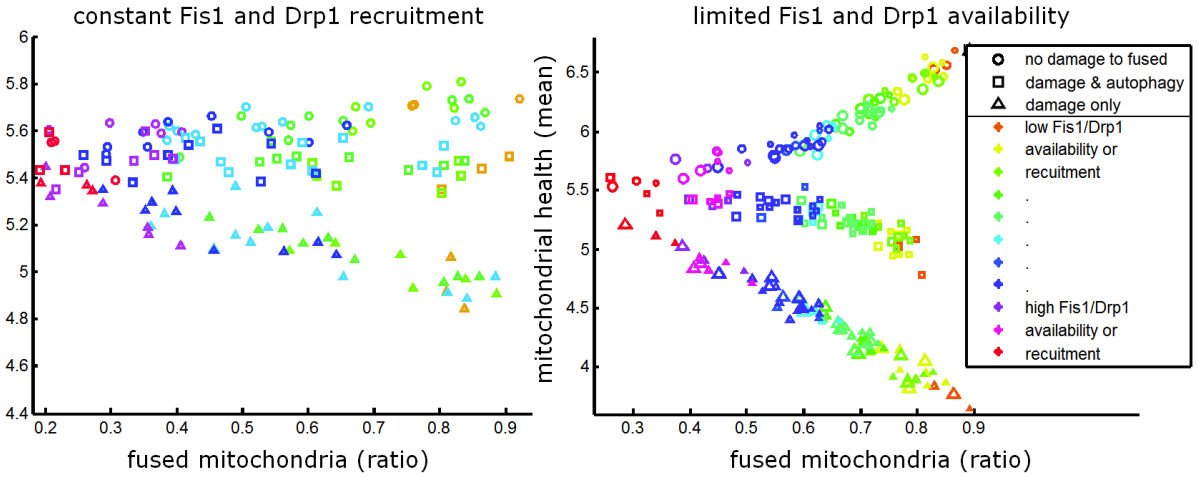


Figure 5.5. Scatter plots of key results for the ubiquitous and limited Fis1/Drp1 scenario.

In initial simulations of the simple model (Figure 5.5 left), Fis1 and Drp1 recruitment were (predictably) negatively correlated with the ratio of fused against free mitochondria (the closest analogy to network structure in the simulation results) and positively correlated with mitochondrial health. This was when fused mitochondria did not loose health on their own like free mitochondria (round markers in Figure 5.5), so fewer fission events meant more mitochondria being safe from damage, which is not realistic. The positive correlation disappeared when damage to fused mitochondria was allowed (triangular markers), and became clearly negative when damaged parts of fused mitochondria could also undergo autophagy (which allowed a new, healthier one to be generated; squares).

In the simple model, average health varied only slightly overall. In the model with explicit Fis1 and Drp1 amounts, these amounts are also parameters influencing fission frequency and thus the ratio of fused mitochondria. Changes in the Fis1 amount had roughly the same effect as changes to the recruitment rate (w.r.t. multiplication by a factor). By varying the amount and recruitment parameters, a much wider range of average mitochondrial health values was covered, and a more pronounced correlation of fused mitochondria ratio (and thus fission frequency) and average health could be observed (Fig. 5.5 right). The parameter for the number of Fis1 required before Drp1 recruitment, interestingly, made no significant difference to the results.

Recent studies indicate that adapter proteins, namely Mff, MID49 and MID51 are important for Drp1 regulated mitochondrial fission. Thus, future research in this direction will include not only expanding the model by explicit fission protein entities whose spatial distribution may not be homogeneous (to be simulated also with our hybrid approach of continuous and discrete space), but also incorporating new wet-lab findings regarding mitochondrial fission.

We also attempted to apply ML-Space to reactive oxygen stress related questions. Presence of reactive oxygen species (ROS), e. g., H_2O_2 and O_2^- radicals, are produced in mitochondria during their normal function, but an abundance decreases their membrane potential (i. e. health for the purposes of our model so far). Attempts to extend our ML-Space model with an existing ROS-related spatial model by (Park, Lee, and Choi 2011), however, were not met with success as the original simulations could not be performed equivalently in ML-Space, i. e. the results could not be reproduced, partly due to insufficient information on how the ROS model's simulation (an "agent-based" implementation tailored for that model) proceeds, partly due to unclear parameter values.

5.4 Individual-based with Bindings and Large Structures: Actin Filaments

5.4.1 Background

Biological motivation[1]

Cells at the interface to an implant surface are able to sense mechanical and biochemical changes in their environment, for instance induced by the interaction with chemical and topographical characteristics of the biomaterial surface via their focal contacts (Selhuber-Unkel et al. 2010). According to the distinct physico-chemical properties of the biomaterial surfaces, cells have the capacity to adapt to it via cell-specific morphological (García 2005; Schwartz et al. 1999; Schwarz and Safran 2013) and functional aspects, e. g., changes in cell morphology, intracellular architecture of adhesion components (Anselme et al. 2000; Lüthen et al. 2005; Nebe et al. 2007) and/or gene and protein expression pathways. For bone cells that were growing on titanium surfaces with regular micro-geometry (namely pillars or grooves), an adaptation of extracellular and intracellular phenotypic traits, including significantly emerging actin filament patterns, has been shown (Matschegewski et al. 2010; Matschegewski et al. 2012).

It could be recognized in diverse experiments that expression and appearance of intracellular structures as well as overall cell shape are influenced by diverse environmental parameters, especially the physical and geometrical properties of the extracellular matrix, e. g., rigidity, dimensionality, composition and ligand spacing (Geiger, Spatz, and Bershadsky 2009; Discher, Janmey, and Wang 2005; Spatz and Geiger 2007). In our experiments human osteoblasts rearrange their actin cytoskeleton in typical patterns mimicking the underlying micro-topography (5 µm dimensions, material surface in Figure 5.6 right). Those changes have so far emerged independently on several chemical cues and variations (Nebe et al. 2014), e. g., usage of glass instead of the bulk material titanium, micro-structured titanium surfaces modified with (i) fibronectin layer due to fetal calf serum, (ii) collagen I coating of the pillars, (iii) sputtering with gold (Stählke et al. 2010) as well as (iv) deposition of a plasma polymer nanolayer exhibiting positively charges to cells (Nebe et al. 2007; Finke et al. 2007). However, the mechanisms behind this restructuring of the actin-cytoskeleton are not clear.

Actin is an abundant and highly conserved eukaryotic cellular protein and the major cytoskeletal component in all eukaryotic cells. It plays a central role in important cellular processes, comprising the transduction of extracellular forces and tensions to the nucleus as well as cell spreading and migration processes (Stricker, Falzone, and Gardel 2010). In particular, many studies report that the structural arrangement of the actin cytoskeleton is decisive for subsequent cellular events (Bershadsky, Kozlov, and Geiger 2006), like the length control of cells or protein expression pathways (Pollard and Cooper 2009). Actin

[1] This section was written, and the figures therein were created, mainly by C. Matschegewski and J. B. Nebe, i. e. the co-author of Bittig et al. (2014a) with a biological background.

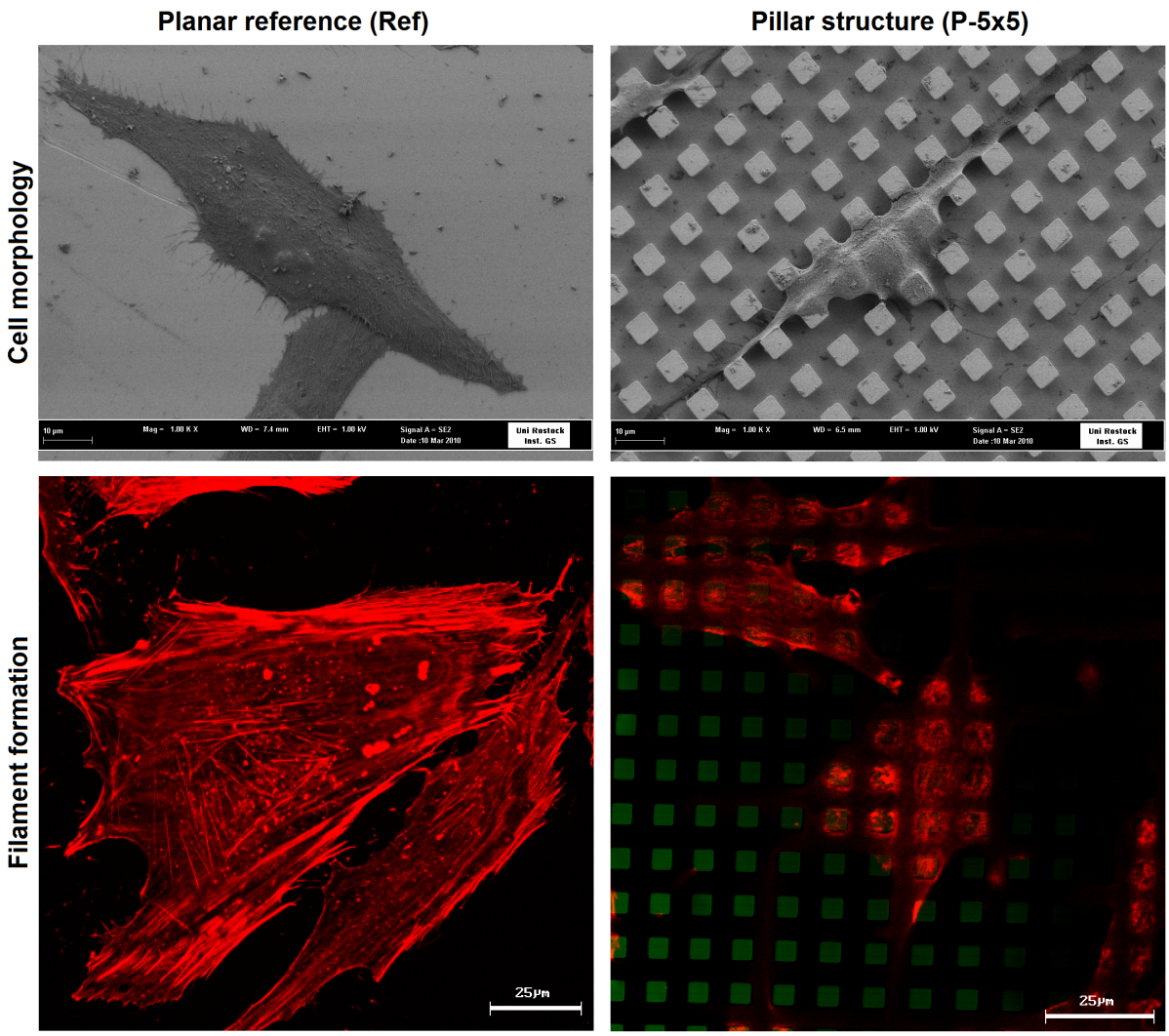


Figure 5.6. Cell morphology (top) and formation of actin filaments (bottom) of MG-63 osteoblasts on planar (left) and geometrically micro-pillared (right) titanium surfaces after 24 h. On the planar surface cells are closely attached to the surface with their entire cell body and exhibit a flattened phenotype. On the pillared surface P-5×5×5, cells are elongated and their adhesion is mainly restricted to the surface plateaus (FE-SEM Supra 25, Carl Zeiss; bar=10 µm), image in cooperation with Regina Lange, Institute of Electronic Appliances and Circuits, University of Rostock). Long and well defined filaments actin filaments form in cells on planar surfaces, whereas on the pillared structure P-5×5×5 the actin filaments are accumulated on top on the pillars in short fibers (LSM 410, Carl Zeiss, green: reflexion mode from the surface, red: phalloidin-TRITC for actin; Matschegewski et al. 2012).

exists in two forms: as globular/monomeric G-actin and filamentous/polymeric F-actin, which is self-assembled in linear filaments containing +/- ends (also known as barbed and pointed ends, respectively) as growing and shrinking sites. The dynamic equilibrium of the continuous reorganization of the actin network is based on its controlled polymerization and depolymerization events, which are guided by a complex interplay of actin with a huge number of regulatory molecules (131 listed in Ditlev, Mayer, and Loew 2013). Among those, ADF/cofilin, Rho and ROCK (Etienne-Manneville and Hall 2002) play a central role. These sustain the balance of actin polymerization and depolymerization and therefore are strongly controlled in their expression and activity pattern (Stricker, Falzone, and Gardel 2010; Pollard and Cooper 2009; Van Troys et al. 2008). Here, the actin regulatory protein, actin depolymerization factor (ADF)/cofilin acts as a key player by severing filaments (Tsai and Lee 2012). ADF/cofilin becomes inactive when it is phosphorylated at its serine 3 residue. ADF/cofilin activity is controlled by the Rho family of small GTPases. Thereby the recruitment of Rho small GTPase and Rho-associated protein kinase (ROCK) leads to subsequent ADF/cofilin phosphorylation (Pfaendtner, de la Cruz, and Voth 2010; see also Figure 5.7). Also, PIP2 has been shown to influence the activity of ADF/cofilin via different mechanisms, including a competitive binding of actin and PIP2 on cofilin (Van Troys et al. 2008; van Rheenen et al. 2007).

Integrins, as transmembrane receptors consisting of an alpha and a beta-chain, are known to provide physical linkages between the extracellular matrix and the actin cytoskeleton via adaptor proteins, e. g., talin, vinculin, paxillin, and α-actin (Zaidel-Bar et al. 2004; Calderwood and Ginsberg 2003; Zamir and Geiger 2001; Dumbauld et al. 2013), thus building a bridge between extracellular space and the cell's interior (Yamada, Pankov, and Cukierman 2003; Brakebusch and Fässler 2003). In addition, they initiate specific biochemical reactions that further regulate the formation of actin filaments locally, e. g., by influencing PIP2 and the cycling of Rho being GTP and GDP bound (Van Troys et al. 2008).

Our hypothesis is that transmembrane receptors like integrin might be affected mechanically by the micro-ranged geometry of the titanium surfaces, and successive biochemical spatial-temporal mechanisms regulate actin polymerization locally. To explore whether this could explain the observed actin filament patterns, i. e. on a planar surface the growth of long, roughly aligned stress fibers, i. e. filament bundles, and in cells on pillar structures only shorter filament segments on pillar tops and edges, we conduct a computational study focusing on the interplay of membrane related dynamics and actin formation.

Related computational models

Many efforts have been made to study actin dynamics at a macroscopic level, incorporating actin's three nucleotide forms (ATP-, ADP·Pi-, ADP-bound), filament branching, capping and severing at different levels of detail (Carlsson 2006; Roland et al. 2008; Beltzner and Pollard 2008; Ditlev et al. 2009). For a comprehensive overview of ex-

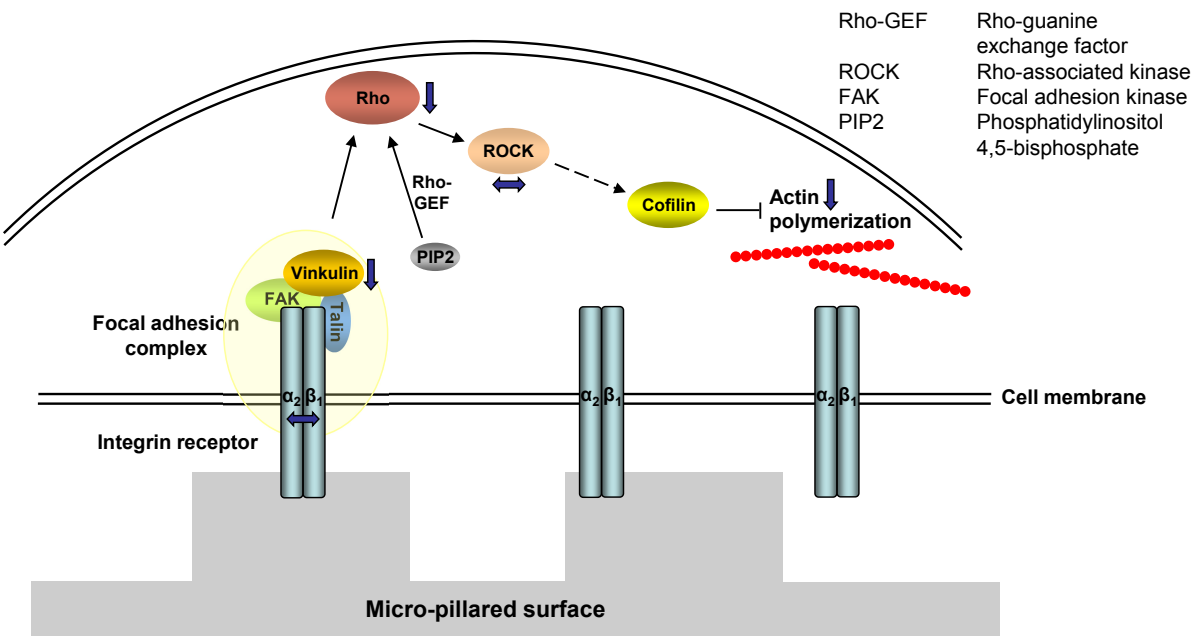


Figure 5.7. Simplified schema of signal transduction by which the topographical influence of the surface activates cofilin via the integrin-Rho-ROCK-pathway, finally inhibiting the actin polymerization. Blue arrows express the total protein expression on the micro-pillared surface in relation to the planar reference (based on the data obtained from the wet-lab experiments of Bittig et al. (2014a); down: significantly reduced expression, horizontal: no significant change).

periments and models, exploring dynamics of filament formation and the regulation of formation- and branching-enabling proteins, see (Ditlev, Mayer, and Loew 2013). Most of the models are non-spatial.

The problem of representing spatial properties can be approached in different ways. For example, BioNetGen has been applied to model actin filaments growth (Sneddon, Faeder, and Emonet 2011), based on previous kinetic models of filament elongation, depolymerization (Roland et al. 2008) and branching (Beltzner and Pollard 2008). The respective models allow studying length and branching structure of a single filament representing its structure as a graph, i. e. in terms of which molecule is bound to which, without positioning the actin filament in space. Microscopic simulations of actin filaments, i. e. where every molecule is represented by an individual model entity with its own position, include a Brownian dynamics simulation of ATP-actin polymerization (Guo, Shillcock, and Lipowsky 2009; Guo, Shillcock, and Lipowsky 2010) with focus on the process of *treadmilling*, i. e. filament growth on the barbed end and simultaneous shrinking at the pointed end. The different polymerization schemes of lammellipodum and lammellum are analyzed in (Huber, Käs, and Stuhrmann 2008). There, the competing roles of ADF/cofilin and tropomyosin lead to two compartments in the cell, one

dominated by ADF/cofilin and closer to the leading edge, and another dominated by tropomyosin. As a result, with the distance to the leading edge, a steep increase of filament length could be observed.

In Ditlev et al. (2009), a comprehensive spatial model incorporating many previous efforts (cf. Ditlev, Mayer, and Loew 2013) is presented. Based on partial differential equations, it is used to investigate the effect of the presence of N-WASp (nuclear Wiskott-Aldrich syndrome protein; known to activate Arp2/3, which in turn mediates filament branching) at the leading edge of migrating cells. By using differential equations, filaments are not individually represented, but their presence and mean length can be determined from the concentrations of filamentous actin and of barbed ends.

In our particle-based approach, we decided to leave many details of actin polymerization and depolymerization processes aside. To shed some light on the mechanisms that drive the observed actin filaments patterns on the micro-structured surface, processes at the membrane have to be integrated. Integrin receptors and the forming of focal adhesion complexes have been subject to a series of models, e. g., of focal adhesion-related signaling with focus on RNA inference (Hoffmann and Schwarz 2013), or focusing on mechanical aspects (Gao, Qian, and Chen 2011). In the above examples the simulation approaches range from non-spatial deterministic population-based modeling (ODEs in Carlsson 2006; Ditlev et al. 2009; Beltzner and Pollard 2008; Hoffmann and Schwarz 2013), discrete stochastic (Roland et al. 2008; Sneddon, Faeder, and Emonet 2011), via mesoscopic (PDEs in Ditlev et al. 2009) to microscopic techniques (individuals with Brownian motion in Huber, Käs, and Stuhrmann 2008, Browian dynamics in Guo, Shillcock, and Lipowsky 2009; Guo, Shillcock, and Lipowsky 2010).

Focus of this work

Whereas most actin models are aimed at analyzing physiological processes that drive actin polymerization and depolymerization, our goal is to understand the impact of a micro-topographic material structure. The focus of modeling turns towards spatial temporal processes close to the membrane. For this, we need to describe geometric structures in continuous space and their interaction with the cell.

As polymerization of actin filaments shall be described as concrete structures developing in continuous space, approaches that are based on discrete space, or assume no volumes associated with the key players (e. g., as adopted in Sneddon and Emonet 2012) do not appear suitable. The same hinders exploiting approaches based on partial differential equations (e. g., as in Ditlev et al. 2009). A definition as cellular automata (Deutsch and Dormann 2004) would constrain the spatial dynamics to the chosen grid granularity and shape. To represent structures emerging from moving molecules in the cell, we thus pursue a microscopic, individual-based approach with movement of molecules approximated by Brownian motion, where molecules cease moving when binding to form filaments. In other words, we apply the continuous-space part of ML-Space, which allows placement of molecules in relation to their binding partners.

Similarly as in Huber, Käs, and Stuhrmann (2008), we keep our model of filament formation as abstract as possible in order to reduce both the number of unknown parameters and the number of simulated particles. Regarding the former, while kinetics of regulatory proteins have been explored previously, conversion of these macroscopic rates to microscopic rates for particle-based simulation would require knowledge of size and diffusion constants of the involved species. Regarding the latter, particle-based simulation is computationally expensive, so simulating a realistic amount of entities (i. e. approaching the number of those present in the cell) becomes infeasible. Unlike models that cover the actin dynamics at the leading edge during cell migration, in our study, the processes of interest take place at the center of the osteoblasts growing on the micro-topographies.

In the following sections, we first present wet-lab experiments done in addition to previously published ones (Matschegewski et al. 2012). These experiments motivated the modeling efforts and influenced the choices of abstraction in the model. In the next section, we describe our choice of modeling and simulation approach and introduce the key reactions of the model created here. The following section contains results of wet-lab and dry-lab experiments. We conclude with model extensions and wet-lab experiments planned in the future.

5.4.2 Model and Simulation Setup

Model abstractions

Since we will primarily focus on the interface of the (structured) surface and the membrane of the cell growing on it, and since the cells ultimately are lying relatively flat on the surface (see also the cell morphology shown in Figure 5.6), we use a two-dimensional model to describe the system. Our model thus covers a section of the cell near the surface and involves both entities that are actually mostly part of the membrane (e. g., receptor complexes) as well as those from the cytoplasm (e. g., free actin). To keep computational effort to a reasonable limit, our models only comprise a section of the center of the cell. We chose an area large enough to fit 3×3 pillars of 5 µm width with an equally wide gap between them.

Even this area can fit a much higher number of entities (many million proteins at a diameter of 4–6 nm, for example) than can be simulated in reasonable time (usually a five-digit number of steps per second depending on machine and scenario). We thus needed to simulate fewer actin molecules than realistically present in the considered area, usually a few thousand per run. Since what is observed in microscopic images of filament formation are actually *bundles* of actin filaments, we chose the size of actin particles larger than it should be relative to the surface structures. The simulation of actin binding in silico can be thought of as representing the formation of several filaments at once. We chose the remaining particle size parameters in proportion to the protein sizes (measured in number of amino acids) for lack of authoritative information, and their diffusion constants inversely proportional to the size's square roots.

Model components

Key model components, actin and integrin molecules, are represented in ML-Space code like this:

```
1 Actin(shape:disk,size:actSize,diffusion:actDiff)
      <pointed:0°,barbed:180°,branch1:110°,branch2:250°>;
2 Integrin(shape:disk,size:intSize,diffusion:intDiff,focal:{"yes","no"})<bs:0>
```

The pointed end binding site is for connecting to an existing filament and the barbed end is where filament growth can subsequently continue. Two binding sites for branching are given to allow for branches extending to either side of the filament chain (at an angle of 70° relative to the growth direction).

In our model, the focal adhesion complexis represented solely by the integrin entity that forms part of it (whose attribute "focal" has the value "yes"); further binding partners of the complex are not explicitly represented in the model. Its sole binding site will later be used to bind an actin to form the start of a filament.

A full model description including definitions of used constants, the branch-initiating species' definition and branching reactions is provided in appendix A.3.

Modeling surface structure

The model also explicitly includes the surface structures. As ML-Space supports hierarchical nesting, e. g., to represent a cell nucleus and proteins inside and outside of it, the extra-cellular surface structure can be represented the same way as a cellular compartment. With a soft boundary the structure represents a "region" whose boundary can be overlapped (as described in subsection 4.2.3).

```
3 SurfStruct(shape:square,boundary:soft,size:5*5);
```

Still, rules have to be specified indicating which particles can enter and leave the region and with which probability they do so when their center moves across the boundary. Attribute values of the moving entities may be changed in the process.

```
  Integrin() + SurfStruct() -> SurfStruct()[Integrin(focal="yes")] @ p=1
  SurfStruct()[Integrin()] -> Integrin(focal="no") + SurfStruct() @ p=1
```

The example would express that integrins can enter the surface structure anytime (i. e. with probability 1 given a collision) and that each integrin that moves onto a surface structure is considered to immediately bind to an (excluded) surface-sensing agent and become part of a focal adhesion complex (attribute change on the right). This may not be a realistic assumption. It should also be considered that a complex of proteins would in reality diffuse more slowly and thus the diffusion attribute should be changed along the way. Moreover, slowing diffusion in distinct regions typically implies accumulation of the slowed down entities in that region (Nicolau et al. 2006), while at the same time the

slowdown of activated integrin at the surface structure might facilitate the recruitment of cytosolic proteins (as indicated in Haack et al. 2013 for slow binding kinetics) and might help clustering the focal adhesion complexes at the surface structure. The forming of the focal adhesion/integrin receptor complex can thus alternatively be modeled to happen on surface structures with a certain rate *rIntComplexFormation*.

```
4 SurfStruct()[Integrin(focal=="no")] ->
      SurfStruct()[Integrin(focal="yes",diffusion=intDiff*intSlowdownFactor)]
      @ rIntComplexFormation
5 Integrin(focal="yes")<bs:FREE> -> Integrin(focal="no",diffusion=intDiff) @
      rIntComplexDis
6 Integrin() + SurfStruct() -> SurfStruct()[Integrin()] @ pIntOntoStruct
7 SurfStruct()[Integrin()] -> Integrin(focal="no") + SurfStruct() @ 1
```

The above also includes a reaction of a focal adhesion complex dissolving, resulting in a freely moving integrin. This shall only be allowed if the respective focal adhesion has no actin (filament) bound, hence the requirement of an unoccupied binding site on the rule's left hand side.

Model reactions: filament growth

Since we aim to reproduce the observed growth pattern with as simple a model as possible, we omit details unrelated to wet-lab observations like actin phosphorylation state, also because more states or attributes would introduce more yet unidentified model parameters and increase the risk of overfitting (Fernández Slezak et al. 2010). We start with a very simple model where free actin exists in only one state (ready to bind to a filament), filaments grow only at the barbed end and filament formation is dependent on an activated integrin receptor / focal adhesion complex.

```
8 Integrin(focal=="yes")<bs:free> + Actin()<pointed:free> ->
      Integrin(diffusion=0)<bs:new>.Actin(diffusion=0)<pointed:new>
9 Actin()<pointed:occ,barbed:free> + Actin()<pointed:free> ->
      Actin()<pointed:occ,barbed:new>.Actin(diffusion=0)<pointed:new>
```

Again, `occ` (or `occupied`) indicates that something, no matter what, is bound at this binding site, where `free` specifies the opposite and the keyword *`new`* (or a different one that occurs exactly twice, e. g., "`bind`") indicates establishing of a new bond if it occurs on the right hand side of a rule. All our "filaments", even those consisting of only two molecules, are considered to be immobile, i. e. we do not model any filament movement.

Model reactions: filament severing

We include in our model a species filling the role of severing actin filaments on collision at the impact site (i. e. the filamentous actin it collided with). This species, called cofilin, can be active or not (although actual cofilin proteins influence actin filaments

in significantly more complex ways –Ditlev et al. 2009; Roland et al. 2008; Tania et al. 2011; note that the actual cofilin is inactive when phosphorylated).

```
10 Cofilin(active=="yes") + Actin()<pointed:occ> -> Cofilin() +
       Actin(diffusion=actinDiff)<pointed:free,barbed:free> @ pActRelease
11 Actin()<pointed:free,barbed:occ> ->
       Actin(diffusion=actinDiff)<barbed:free> @ rFilDissolution
```

The first rule specifies that if active cofilin collides with an actin that has another entity bound at the pointed end, all bindings of the actin are released (potential branching sites omitted above) with given probability. As binding is a symmetric relationship, what is bound at the pointed and barbed end will be affected by this rule, too: the value of its respective binding site will be set to FREE as well. The second rule says that the remaining filament part starting with a free pointed end actin will be dissolve from the pointed end (if the specified rate $rFilDissolution$ is > 0; in our simulations we used an infinite rate, i. e. the whole chain will be converted to free actins in the same time step, as the same rule will be applied successively to all actins in the remaining filament trunk).

Wet-lab results indicated that cofilin activity is (negatively) regulated by actors related to the integrin receptor complex. We integrated two different potential regulatory relations into our model.

First, we let cofilin be deactivated on every contact with the focal adhesion complex, and get reactivated on its own (i. e. by agents assumed to be constant and not explicitly in the model) with a given rate $rCofilinReactivation$.

```
12 Integrin(focal=="yes") + Cofilin(active=="yes") -> Integrin() +
       Cofilin(active="no") @ pCofDeactAtInt
13 Cofilin(active=="no") -> Cofilin(active="yes") @ rCofilinReactivation
```

Secondly, we represented the intermediate steps by an intermediate component called `CofReg`, which stands in for several potential regulatory proteins possibly in a multi-step cascade whose details, including kinetic parameters, i. e. reaction rates, are not known. Our `CofReg` deactivates cofilin and is itself activated at the focal adhesion complex. We let `CofReg` appear near the receptor complex and disappear with a certain rate. Alternatively, one could simulate a fixed amount of `CofReg` entities that are activated near the receptor and get deactivated on their own, like cofilin. Both variants should lead to the same patterns with respect to amounts and distribution of active `CofReg` during the simulation. The second alternative requires more simulation effort for the inactive `CofReg` not present otherwise (and could also lead to more molecular crowding effects due to molecules impeding each other's movement more often, which is undesired here since our 2-dimensional approach already leads to occasional spurious blocking of particles).

By setting either $pCofDeactAtInt$ or $rCofRegAppearance$ to 0 one can then select the mechanism to be simulated.

```
Integrin(focal=="yes") -> Integrin() + CofReg() @ rCofRegAppearance
CofReg() -> @ rCofRegDisappearance
CofReg() + Cofilin(active=="yes") -> CofReg() + Cofilin(active="no") @ 1
Cofilin(active=="no") -> Cofilin(active="yes") @ rCofilinReactivation
```

Simulation and experiments

We simulated models with differences in some mechanisms, e. g., regarding cofilin activity (holding active cofilin constant, having it deactivated as first-order reaction, cofilin deactivation by the integrin/focal adhesion complex or by an intermediate entity that is itself activated by the receptor complex). For lack of information on the amounts of proteins of each type and the reaction probabilities, we tested different values for crucial parameters.

We simulated our model with all possible combinations of values for key parameters given in Table 5.1. In addition, we also ran simulations with cofilin activity regulation directly at the integrin receptor complex ($pCofDeactAtInt = 1$), i. e. without any `CofReg` ($rCofRegAppearance = 0$). For collision-triggered reaction, a decrease in the probability of the reaction happening on each collision and an increase in the amount of a reactant by the same factor have opposite effects on the number of occurring reactions, which roughly cancel each other. Probabilities of collision-triggered reactions were thus set to 1 or near 1, unless otherwise noted.

Table 5.1. Key parameters in model simulations. Values were chosen after multiple experiments with values outside the given ranges had lead to either behavior similar to those reached with a combination of the above values, or to behavior not matching the wet-lab observations (e. g., very short filaments, or very long filaments in both the planar and the micro-structured setting). Cofilin activity regulation via intermediate step `CofReg` has been assumed above, i. e. $pCofDeactAtInt = 0$. For actin, initial amounts are given. Additional actin was generated during the simulation (at a constant rate, such that there were 5000–6000 actin entities at the end of each simulation). Note that our actin chains are supposed to represent filament bundles and thus the proportion of actin to other actors, especially cofilin and integrin, need not correspond to realistic conditions.

Parameter	Type	Values used
Actin	amount	250, 500[(1)]
Cofilin	amount	400, 800
Integrin	amount	100, 200
rCofRegAppearance	reaction rate (1st order)	0.5, 2
rCofRegDisappearance	reaction rate (1st order)	0.5, 2
angle deviation	distribution	$\mathcal{N}(0, 7.5°^2)$, $\mathcal{N}(0, 15°^2)$, $\mathcal{N}(0, 30°^2)$
surface structure	(qualitative)	planar surface, (groves), pillars

The simulations were repeated several times, with a very small variance. Thus, in the following we will focus on one set of results. The simulation end time was the same for all runs (48 a.u.) and chosen from experience with previous tests such that no significant change in key model outcomes, especially average filament lengths, should be expected anymore.

ML-Space model interpreter and simulator are implemented in JAVA and integrated into the modeling and simulation framework JAMES II (Ewald et al. 2010), which also served as experimental framework for most of the parameter scan experiments. Some optimization experiments were executed by using the simulation experiment specification language SESSL (Ewald and Uhrmacher 2014). *MATLAB version 7.10.0 (R2010a)* (2010) was also used for the evaluation of larger result sets.

5.4.3 Results

In our *in silico* experiments, we compare different strategies of activating integrin, analyze the impact of changing the amount of integrin, actin, or cofilin in the system, and take a closer look at the impact of the severing mechanisms and the local distribution of the severing agent.

Amounts of key players

The amount of *actin* is the most obvious determining factor of filament lengths. Too few of them, and filaments eventually run out of free monomers to bind, staying short. Too many actins, however, and many simulation steps consist of actin moves and possibly collisions without reaction, especially at the beginning of the simulation when few nucleation points or filament ends are available. Since actins in filaments are more tightly packed than free actins, once enough actins are bound to filaments, the few left free actins (or those freed again by depolymerization) move through larger patches of empty space before occasionally encountering a filament to bind to again. Thus, we decided to start with a comparatively low amount of actin and to create actin molecules during the simulation to roughly maintain the density of free actin. This can be interpreted as recruitment of free actin from the cytosol. Once we did this, the initial amount of actin was no longer a key parameter for the length of filaments encountered later.

Since in our model filament formation starts at an *integrin* receptor complex and we allow only one filament at each of them, the amount of integrins in the system limits the number of filaments that can form. When integrins could easily turn into focal adhesions (and thus filament nucleation points), their amount was indeed found to negatively influence the resulting average filament length (as illustrated in Figure 5.8), at least when keeping other parameters constant. Thus when actin is added at a constant rate as above and filaments are severed, a simulation with more integrin must be run longer to exhibit filaments of same average length as a simulation with fewer integrin. When filament severing happens often, the integrin amount becomes less relevant: filaments stay comparatively short independent of whether there are few or many of them.

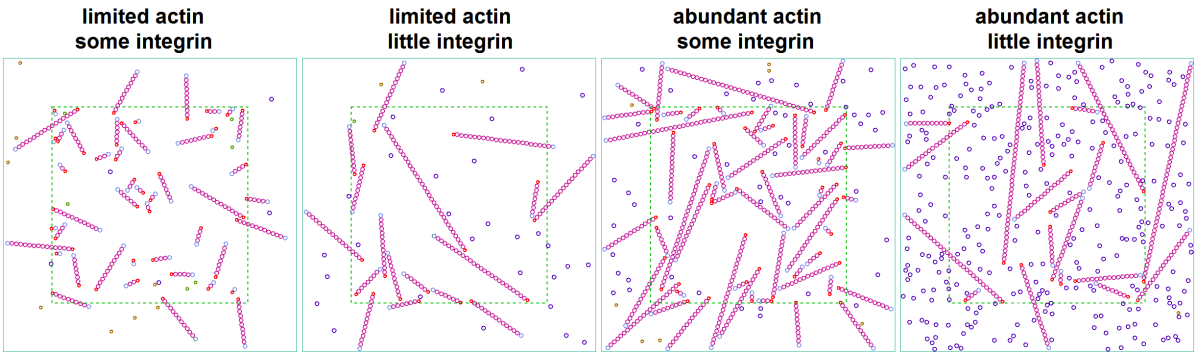


Figure 5.8. Simulation illustration with small single surface structure with a fixed amount of actin and no limitation of filament orientation or severing (i. e. no cofilin). Magenta circles represent actin entities with other entities bound at both ends (i. e. in filament chains), light blue circles are actins at filement ends (i. e. occupied pointed end, free barbed end), Purple/dark blue circles: free actin, smaller red circles: integrins at start of filament, dark green and brown circles, free integrin on and outside of a surface structure. Surface structure boundaries marked by green dashed lines. Too little actin can lead to relatively short filaments (left), too little integrin and abundant actin to a few long filaments and many free actins that have no room to bind to a filament (right). Filament "growth" can also be impeded by the two-dimensional approach where filament crossing or bending is not allowed (center right; note also several free actins trapped in regions bound by different filament segments).

More *cofilin*, i. e. filament severing agent, in the simulation expectedly lead to shorter filaments (again when keeping other parameters constant). However, it was the amount of *active cofilin* that is relevant for filament lengths, and the amount of active cofilin depends on these other parameters, i. e. the rates of reactions regulating cofilin activity (see "Severing by cofilin" below and Figure 5.9), so the total cofilin amount alone is not a key parameter.

Mechanisms of integrin activation

The mechanism by which free integrin turned into focal adhesions – with a certain stochastic rate (but only when on surface structures) or instantly upon entering such a surface structure (cf. "Modeling surface structure") – made no difference to the simulation results. In the former case, increasing this rate and increasing the amount of integrin had the same effect – what mattered was the amount of active integrin / focal adhesions available for filament formation.

The initially hypothesized slowdown of integrin on pillars in our simulations leads to accumulation on these structures. When at the same time integrins cannot easily enter a surface structure, a roughly homogeneous distribution of integrin can be achieved,

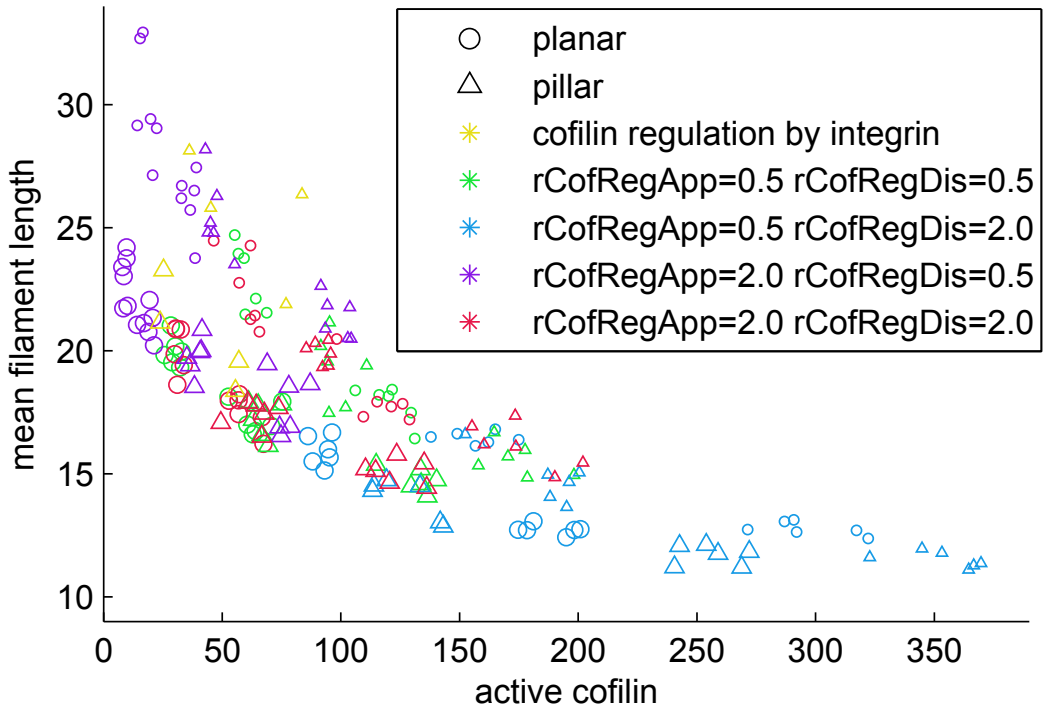


Figure 5.9. Relation of filament size to amounts of active cofilin and other parameters. Each marker indicates the result of a simulation run with different parameters. Colors denote parameters related to the cofilin activation mechanism, marker types distinguish simulations of pillar structure systems (triangles) and planar surfaces (circles), and larger markers indicate simulations with 200 initial integrin entities vs. 100 for the smaller markers (mostly to be found above the former). More active cofilin thus coincides with shorter filaments.

too (Bittig et al. 2014a, fig. 4). However, in our simulation most integrins eventually organize in focal adhesions. This is partly because we limited the number of integrins to reduce calculation time. Nevertheless, the wet-lab observation of more activated integrin on pillars than between them can thus also arise in models without a slowdown.

Severing by cofilin

We compared cofilin regulation directly at (i. e. on collision with) the focal adhesion complex with a short cascade with an intermediate signaling entity, `CofReg`. The amount of active cofilin in the whole system depends on the parameters of the cofilin-regulating reactions (and the total amount of cofilin). With more `CofReg` ($rCofRegAppearance > rCofRegDisappearance$, purple markers in Figure 5.9), only a small fraction is active and filaments grow longer on average, with little `CofReg` ($rCofRegAppearance <$

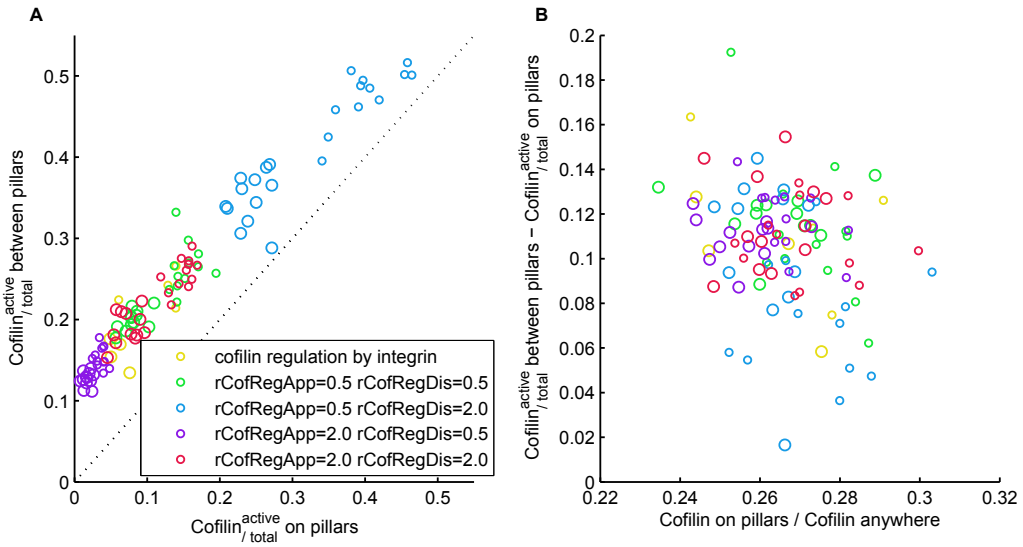


Figure 5.10. Overrepresentation of active cofilin between pillars due to (de-)activation mechanism tied to the focal adhesion complex. (Rate parameter names related to CofReg, *rCofRegAppearance* and *rCofRegDissapearance* slightly shortened.) **A:** The fraction of cofilin that is active is higher in the region between surface structures than on surface structures (all points are above the dotted line, where both would be equal). **B:** The fraction of cofilin that is located on pillars (x-axis) is often slightly lower than the pillars' share of the system size (29.7%), but unrelated to the difference in activity ratios of cofilin between the respective regions (note that each marker's position on the y-axis corresponds to the respective markers' distance from the dotted line in the left panel).

rCofRegDisappearance, blue markers), filaments are severed earlier and/or more often.

Since we assume that a significant part of the cofilin regulation mechanism happens in regions where the focal adhesion complexes reside, we compared the ratio of active cofilin to total cofilin in the respective regions, i. e. on and between surface structures (pillars). A small difference can be seen between the respective fraction for surface structures and for regions between them, also when cofilin (de)activation does not depend on an intermediate signaling entity (see Figure 5.10 A; note that the comparison only makes sense for experiments with pillared surface structures, not for planar surfaces). This can go so far that there is almost no active cofilin on pillars, but at the same time the vast majority of cofilin between pillars is also inactive. This is not an artifact of a skewed overall distribution of cofilin (e. g., resulting from accumulation outside pillar regions; Figure 5.10 B).

Note that all shown results stem from simulations with the same rate at which deactivated cofilin becomes active again (*rCofilinReactivation*). With higher or lower values for this rate, the active cofilin ratios can be made higher (towards the blue markers in Figure 5.10 A) or lower (towards the purple markers), respectively.

Table 5.2. Parameters and results of simulation runs shown in Figure 5.11. All four runs had these parameter values in common: initial actin amount 500 (changes here did not significantly change the results as most actin was "produced" during the simulation), integrin amount 200 (a lower amount, 100, lead to 20% longer filaments in situations where cofilin activity was low, i.e. corresponding to the top panels), cofilin amount 800 (lower values lead slightly longer filaments in *all* four settings).

Figure 5.11 panel		**Top left**	**Top right**	**Bottom left**	**Bottom right**
rCofRegAppearance		2.0	2.0	0.5	0.5
rCofRegDisappearance		0.5	0.5	2.0	2.0
angle deviation		$\mathcal{N}(0, 15^{\circ 2})$	$\mathcal{N}(0, 15^{\circ 2})$	$\mathcal{N}(0, 30^{\circ 2})$	$\mathcal{N}(0, 30^{\circ 2})$
$\text{Cofilin}^{active}_{/total}$	on pillars	0.028	0.021	0.22	0.26
	between pillars		0.141		0.38
average filament length (# part.)		21.1	18.5	13.0	11.4
max. filament length (# particles)		69	58	29	33

Also, the pillared region experiments lead to slightly shorter actin filaments than comparable experiments with a planar surface (Figure 5.11 right vs. left; cf. Table 5.2 and Table 5.3). As the ratio of active cofilin vs. total cofilin on pillars is roughly equal to the one in comparable planar surface simulations, we attribute the shorter filaments in pillar structure simulations to the higher chance of filaments being cut off between pillars rather than on them.

Filament orientation and branching

Average filament length is slightly negatively correlated with our parameter angular deviation, which lets the filaments grow in similar direction, again when holding other parameters constant. More specifically, here the angle of each filament relative to the horizontal is set to a normally distributed value with mean 0 and a certain standard deviation (representing orientation dispersion). When filaments are more closely aligned, they can become slightly longer on average, mostly because when we allow filaments to grow in whichever direction they please (which would be the direction from which the first bound actin approached the original focal integrin), they get in each other's way more often, limiting further growth (provided the other chosen parameters allow long filaments in principle). This could be considered an artifact of the chosen two-dimensional approach, assuming that one filament could simply grow above or below the other for a small segment in a three-dimensional approach.

Occurrences of one filament segment preventing further growth of another one are slightly more frequent in simulations where proteins are allowed to branch, i.e. where a certain amount of Arp2/3 entities are available from the simulation start. Here, overall filament sizes (when counting all entities in one filament complex, i.e. all branches of it)

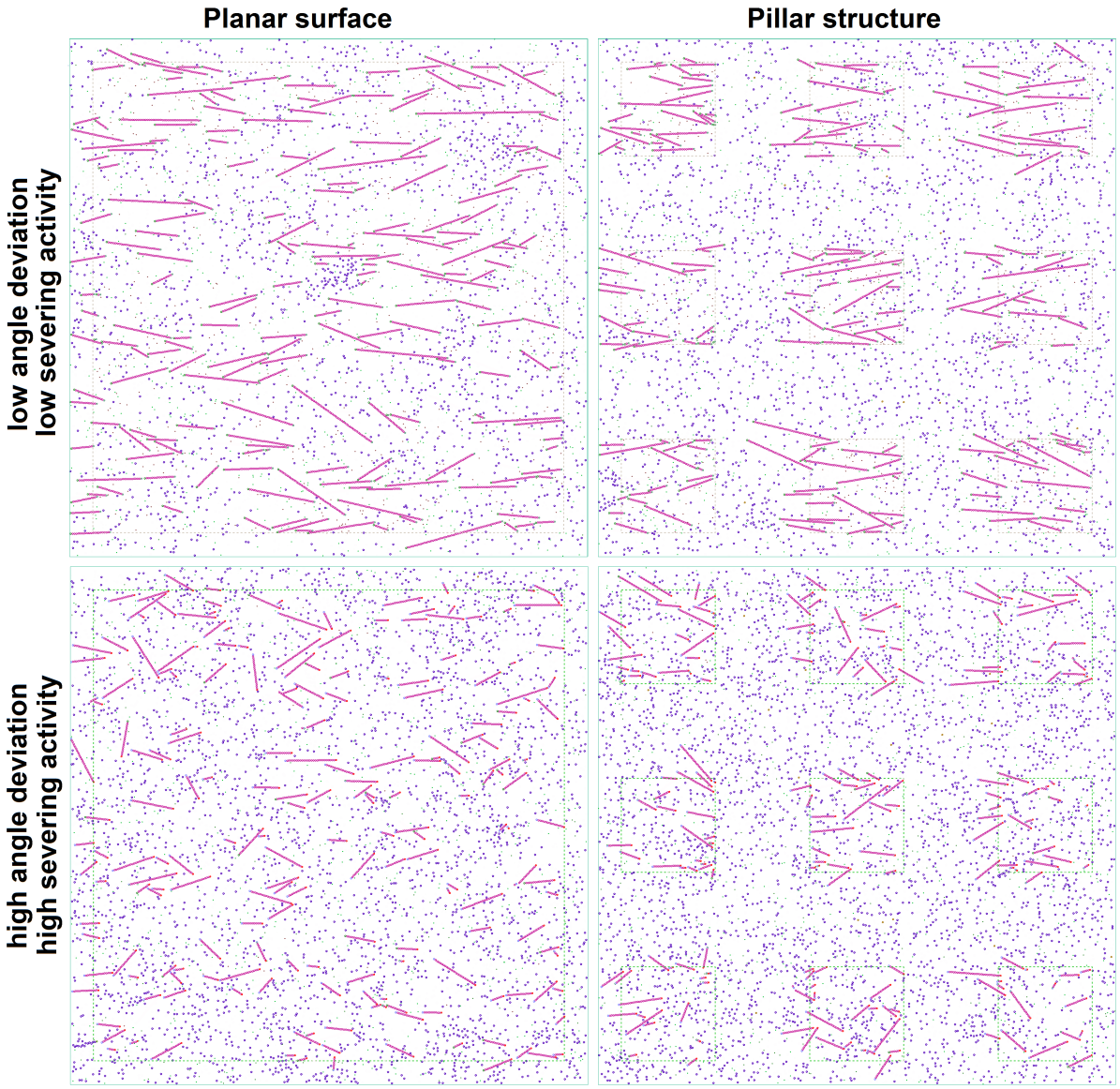


Figure 5.11. Simulation results with different severing agent amounts and orientation dispersion values. See also Tables 5.2 and 5.3 and note that the difference in conditions between upper left and lower right reflects wet-lab observations.

Table 5.3. Quantification of wet-lab and simulation experiments. Ref and P-5×5 refer to wet-lab experiments for planar and pillared surfaces (taken from Matschegewski et al. 2012, Table 1; $p < 0.001$ for the changes in all three quantified properties; averages and standard deviation of 30 cells per specimen). The final four columns refer to the simulation experiments (averages and standard deviation of seven simulation runs for each parameter combination; filament length calculated based on an actin diameter of 0.05 µm). The two columns in bold there best reflect the changes of conditions for the actual cell. Changes in average filament length significant at $p < 0.001$ planar vs. pillar with same severing agent amount and angle deviation, all changes significant when comparing the first and last experiment column. Note that length values *in vitro* and *in silico* are not comparable – the FilaQuant software processing florescence microscopy images has cutoff parameters (regarding length and thickness of lines to consider a filament) while for the simulation every integrin/focal adhesion with at least one bound actin is considered a filament.

	***in vitro* (Matschegewski et al. 2012)**		***in silico* (Figure 5.11)**			
			low angle dev. low severing		**high angle dev. high severing**	
	Ref	**P-5×5**	**planar**	pillar	planar	**pillar**
Average filament length (µm)	9.7±1.5	3.1±1.5	**1.09±.03**	0.90±.03	0.64±.03	**0.55±.03**
Maximum filament length (µm)	51.5±11.9	6.7±2.0	**3.03±.33**	2.86±.17	1.82±.35	**1.77±.18**
Orientation dispersion (%)	66±14	84±10	**36±1**	36±1	65±1	**65±1**

become larger relative to simulations with otherwise equal parameters but only straight filaments, although the longest segment chains in filaments seem shorter on average (cf. Figure 5.12). The angular deviation model parameter then has little effect on the orientation dispersion observed in the simulation, as branches always grow with an angle of 70° relative to the filament from which they branch off, leading to differently oriented branches.

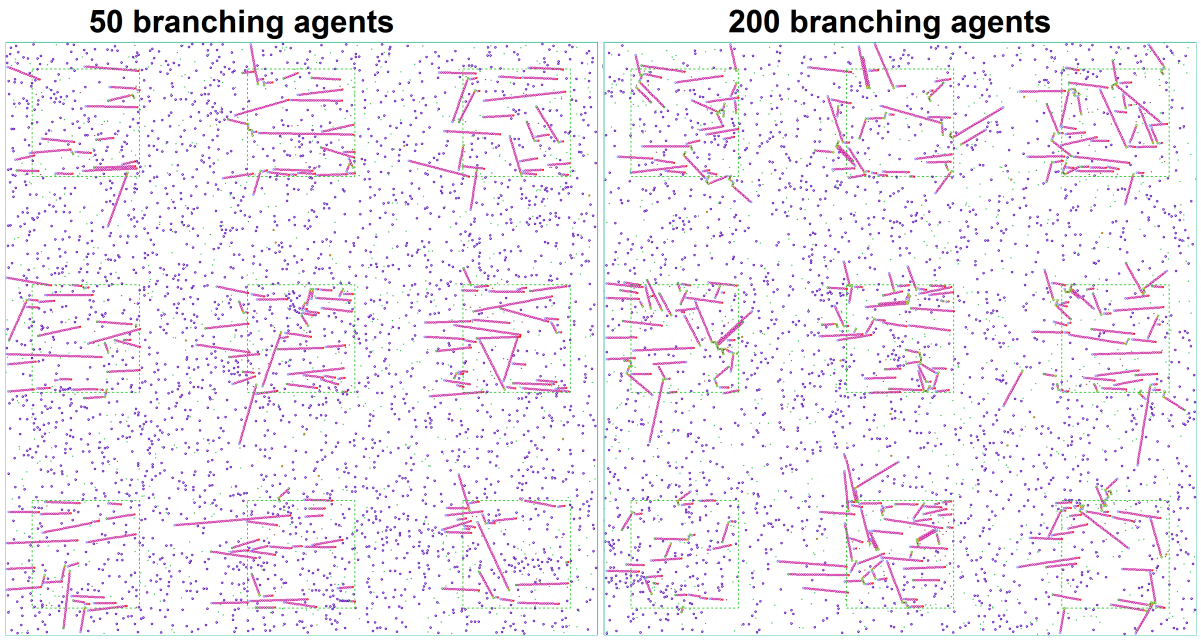


Figure 5.12. Simulation results with branching enabled. Other model parameters correspond to the low orientation dispersion, low severing situation (Figure 5.11 top). The amount of integrin (and thus maximum number of filaments) was 200.

5.4.4 Discussion and Conclusion

So far we focussed on the effect of changes to single or few parameters while keeping the others constant. We also found that the "structure parameter", i. e. the size of the area where the cell has surface contact and thus where filament growth can start and where the actin depolymerizing factor (cofilin) is regulated, can already explain part of the lower filament length in systems with the pillar structure compared to planar surfaces (Figure 5.11 left vs right).

Wet-lab experiments demonstrated that several of the aforementioned parameters (e. g., orientation dispersion and severing agent regulation as exemplified by Rho and vinculin expression) are different between the two situations, it is more appropriate to compare the planar surface simulations with low dispersion and low severing with the pillar structure simulation with high orientation dispersion and high severing agent (cofilin) activity (Figure 5.11 top left vs. bottom right). Then, the difference in filament lengths and patterns becomes more pronounced (see also Table 5.3).

Since we could establish that average filament lengths are sensitive to several parameters, some of which should be different between cells on planar surfaces and those on micro-structured ones, tweaking these parameters in one of the two settings can make the difference between their simulations larger or smaller.

Overall, our results indicate that sensing mechanisms and bio-chemical regulation of actin filament severing via cofilin might play a central role in explaining the phenotypical

differences between osteoblasts grown on planar vs. geometrically micro-structured surfaces, the former due to the apparent concentration of filaments in areas where the cells had surface contact, the latter because of expression differences in regulatory proteins upstream of cofilin.

Based on the wet-lab results, we developed a spatial computational model of actin filament formation with several abstractions that lumped multi-step processes with yet unquantified components into single steps. First, based on a hampered entry of integrin into the pillar structure regions and a subsequent slowdown of integrins, the observation of homogeneously distributed integrin in wet-lab experiments could be reproduced. Subsequent wet-lab experiments then showed that activated integrins are slightly clustered on pillars. In earlier studies, it had already been shown that vinculin can be found strongly clustered on pillars. From both finding, we hypothesized that the focal adhesions, of which vinculin is a part, indeed form predominantly on pillars.

The spatial patterns of further selected members of the focal adhesion complex will be analyzed in future studies, also to shed light on their role in regulating the actin cytoskeleton. Our results indicate that filament growth pattern can result from the receptor complex' role in regulating the actin depolymerizing factor (ADF)/cofilin.

Due to deactivation of cofilin in the vicinity of focal adhesion complexes (containing vinculin and activated integrin), which were clustered on the pillars, a higher concentration of active cofilin could be found between the pillars in our simulations, which increased the probability of actin filaments being cut off outside the pillar structures.

The resulting differences in our simulations, if we assume otherwise identical model parameters (i. e. reaction rates and actor amounts) for structured and non-structured surfaces, appear less pronounced than in *in vitro* observations. However, our findings in wet-lab experiments suggest that these identical model parameters are not realistic. e. g., we find a higher alignment (i. e. orientation dispersion) of actin filaments and higher Rho expressions on planar surfaces. The simulation results for pillar structure surfaces and those for planar surfaces were even closer to the wet-lab observation if these measured differences were added explicitly to the simulated models, i. e. orientation dispersion and the regulation parameter for cofilin were adjusted accordingly. The effect of shorter actin filaments on pillared structures compared to planar ones is, however, still more prominent *in vitro* than observed *in silico*. processes that could induce the release and activation of additional cofilin at the pillared surface.

On the computational side, ML-Space allowed to probe different hypothesis and a successive extension of the model easily. Potentially useful expansions for studying actin filament dynamics include movement of bound entities i. e. filaments, instead of fixing their position in the event of binding, ways of representing the cellular stress to be the source of orientation alignment, and to speed up the simulation to allow more simulation runs with more realistic protein sizes and consequently many more proteins in the system.

6 Discussion and Conclusion

6.1 Limitations and Future Work

By bringing together entities with attributes, multi-compartmental semantics and spatial aspects, ML-Space requires a much more complex simulator than one covering only one or two of these issues, necessitating distinction of many special cases that require targeted solutions rather than following a single overarching idea. For some of these cases, a line had to be drawn for how complex a situation should be supported, e. g., by restricting spatial effects of rules to the organizational level at which they were triggered, except for explicit transfers. Here, pushing these lines by adding more special case handling will allow ML-Space to cover an even wider range of possible processes. However, this may also make the simulator behavior more complex in ways that might confuse potential modelers who would need only a fraction of its capabilities.

6.1.1 Modeling Language

Multi-Level Rules Flexibility

Reasoning that a collision between entities should have an effect only on the immediate vicinity and only on these entities' organizational level, a decision was made early on to limit multi-level reactions to transfers of a single entity into or out of another, possibly with attribute changes. This principle was eventually softened: to cover upward causation, a collision can also change attributes of the surrounding entity, which in turn may trigger immediate first- or zeroth-order reactions inside this entity, but not necessarily in the vicinity of the original collision. This approach could be relaxed further: implementation-wise, it is not difficult to let a particle colliding with a larger particle's boundary trigger the appearance of an entity of a *different species* on the other side of the boundary (e. g., an outside signal making a receptor release something inside the cell). One might even want to allow endosome formation like this: `P + Cell ->Cell[Endosome[P]]` (which is already possible in ML-Rules, where no spatial constraints need to be considered).

Further BNGL or ML-Rules Features

BioNetGen and its language allow entities to be bound at multiple sites, or to be connected via several different routes in a complex (e. g., circular molecules). This is difficult to reproduce in ML-Space due to the imposed spatial constraints.

Other features of BNGL, however, could conceivably be added to ML-Space and facilitate extensions of our mitochondria model. For example, entities can be created bound to already existing ones, e. g., to express recruitment of abundant, and hence not explicitly modeled entities. Also, entities in the same complex can react even if not directly bound to each other, which could be used for entities on mitochondrial surfaces.

Regarding bindings, ML-Space' spatial resolution is currently focused on two-dimensional applications and fixed angles. Flexible angles – e. g., specifying a mean angle and a variance – and three-dimensional specification could be added, which would mainly be a question of choosing a suitable representation of spatial angles for the language and implementing the maths of aligning the center of spheres accordingly. However, flexible binding angles are already handled by SRSim (Gruenert et al. 2010) and the underlying Molecular Dynamics simulator in a more physically accurate way (but without entity attributes other than binding sites and no multi-level dynamics).

Entity counts, e. g., to keep the number of mitochondria in a cell constant, can be maintained in ML-Space as an attribute of the context, i. e. surrounding entity, but need to be explicitly incorporated into every rule affecting the respective number. BioNetGen's named observables and ML-Rules's functions on solutions offer alternatives that some modelers may find more elegant (but which are probably less efficient when applied to ML-Space).

Spatially Dependent Rules

Space itself has no inherent properties in ML-Space. In the actin filament model, we used regular, immobile model entities to represent regions to which some reaction rules were limited. We also added special constants for orientation of bound entities to account for the fact that actin filament orient themselves and grow towards the *tip* of a cell. This concept of a preferred direction could be generalized, such that one could also express, for example, that mitochondria predominantly move towards or away from the nucleus if the (implicit) microtubule they move along in our model were oriented this way.

Also, some unicellular organisms are able to sense concentrations in their surrounding and move according to the gradient, e. g., *Dictyostelium discoideum* moving towards higher cAMP concentrations. In attributed formalisms where space can only be encoded explicitly in the model, this can be directly incorporated into the rules (Beccuti et al. 2015). In ML-Space it would have to be established first whether a gradient exists – an equivalent of *funtions on solutions* could help – and new syntactic constructs to make movement dependent on it would be needed.

Asymmetrical Shapes and Particle Complexes

For future applications, one might actually wish to represent particles of more complex shapes, e. g., rods with a given orientation to explicitly model micotubule along which the mitochondria move. This can already be approximated by using a chain of bound spheres instead (cf. actin filaments).

Interactions with molecule complexes in ML-Space, however, are always focused on a single particle of that molecule. Thus it is not yet possible to express moving along such a chain, which would involve unbinding from one molecule of the chain and binding to the next one, i. e. the release of a particle `P` from a molecule `C` in a chain would be expressed `C<prev:C,next:C,side:P>` → `C<side:release>` and neither `P` nor the `C` at the `next` position would be available for further changes, as first-order rules have no direct spatial effect beyond the bond now being considered released.

6.1.2 Physical Realism

Rates and Kinetics

Second-order reactions between spatial, individual entities in ML-Space require a probability for reactions between two particles when a collision occurs, i. e. when diffusive movement of particles momentarily leads to the centers of two of them being closer than the sum of their radii (which is "resolved" by moving a particle away from the other before the end of the respective simulation step). As shown in section 2.3.6, reaction probabilities can be derived from macroscopic rates for second-order reactions. These derivations are only physically accurate if several conditions are fulfilled:

- The particles are roughly spherical.
- The diffusion constants of the involved particles in the given environment are known with sufficient accuracy.
- The mass action kinetic rate constant is known with sufficient accuracy.
- Mass action kinetics accurately describe the actual behavior of the reaction in the given environment.

These could not be taken for granted in any model ML-Space was applied to up to now. Also, particle numbers were not known or uncertain in our major applications.

When fitting both particle numbers (or concentrations) and reaction probabilities (or rates) to observed system behavior (e. g., a known number of reactions occurring in a given time interval), only the product of the two unknown parameters can be estimated. In such a case, it is computationally convenient to use probability 1 because fewer individual particles have to be simulated (as it has been done for the actin filament model). The question of deriving probabilities then becomes moot.

However, future versions of ML-Space could provide assistance in converting macroscopic rate constants to probabilities, e. g., by dividing a given rate by the sums of interacting particle's radii and diffusion coefficients *on the spot* when encountering a rate (`...@ r=...`) in a rule that requires a probability (if the participant's diffusion attributes are known at modeling time, this conversion can already be specified explicitly). Conversion at simulation time has downsides: the modeler needs to be aware of this intertwining when diffusion attributes can also change and if a larger step size is chosen the probabilities would need to be adjusted upward to account for missed collisions. However, the appropriate adjustment factor would also depend on the crowding of the

environment and the dimensions of the system and is therefore most easily derived by fitting again.

Adaptations to the simulation algorithm or even the spatial semantics would be needed to apply other methods mentioned in section 2.3.6: the (Smoldyn) approach of reaction radii, i. e. particle distances below which a reaction certainly happens, is inapplicable to simulation with hard spheres (instead of point-based particles) as a reaction with an attribute-dependent rate (e. g., `E(x=nPhos)+S ->E+P @k*x`) would lead to different values for the sum of the reactant's radii for different attribute values, but no reactant should be expected to change size significantly on a minor modification. The approach of reaction radii combined with probabilities for particles close enough would require keeping track of the time particles have spent in each other's proximity (the Δt for the time between Brownian "jumps" may differ between the two) or switching to the same fixed time steps for all particles (instead of fixed average length steps), but this would lead to unnecessarily many very small steps of larger (slower) particles.

Applying these methods to ML-Space could be done by introducing the concept of reaction radii distinct from particle sizes or by not resolving overlap between particles in each step (i. e. by allowing $||x_1 - x_2|| < r_1 + r_2$ for non-zero time intervals; which would complicate positioning for binding at given angles, however).

6.1.3 Simulator

Small Spatial Entities in Large Subvolumes

So far we distinguish two types of entities in an ML-Space model: dimensionless entities for subvolume-based simulation and spatial entities with extensions and individual positions, which may contain other entities (of either kind). ML-Space' hybrid simulator requires subvolumes to be smaller than the smallest spatial entity in the system such that there can be no spatial entities sitting "between" subvolumes yet potentially containing dimensionless entities that can be transfered out of them.

The hybrid approach by Klann, Ganguly, and Koeppl (2012) combines population-based simulation and individual particles without hierarchical nesting (but with excluded volumes, e. g., for cellular structures and membranes), where entity representation as individuals or parts of a population can be switched during the simulation (in ML-Space, the entity type is determined by their species definition).

Extending ML-Space by small individuals in subvolumes was briefly considered, mainly while working on the actin filament model as it might have allowed representing free actin, cofilin and other actors as populations with only bound actins, i. e. filamentous structures made of individual particles. While such an extension might work without any modification of the modeling language, semantics and simulator implementation would be significantly affected:

- One would need to distinguish three types of entities: dimensionless ones, large individuals that are potential compartments and small individuals that can never contain other entities.

- Interactions between subvolume populations and individual particles would need to be handled (e. g., with second-order reactions considered like first-order reactions of an individual in a fixed concentration of other particles).
- The subvolume's volume fraction occupied by small individuals would need to be accounted for (unless one went for point-based individual particles with reaction radii but no hard boundaries). If the small individuals could form complexes, these might actually create a barrier inside a subvolume to which the subvolume simulator would be oblivious (e. g., an actin filament "cutting through" a 2D subvolume).
- Binding between large and small individuals would also be problematic as their movement would be handled differently. (However, the current hybrid simulator is not very suitable either when individuals can bind, as the movement in steps of subvolume size conflicts with the positioning of bound particles according to given angles.)

Changing Shapes

In relation to the language extensions for more complex shapes considered above, the simulator would need to be adapted as well, e. g., when a particle bound to one molecule in a complex switches "allegiance" to another in the same complex and thus needs to move.

Additionally, we have so far avoided re-sizing of spatial entities, which would require a policy to deal with compartments shrinking such that the content does not fit anymore or trying to expand but being blocked by others. Splitting or unification of compartments was not covered for similar reasons, especially when it would involve rather drastic shape changes (replacing a sphere with two smaller, neighboring ones or vice versa) where finding new positions for all previously contained entities would be difficult and, more importantly, will lead to behavior the modeler may not expect. ("Spawning" new compartments or destroying them on collision is still possible, but no content would be created or absorbed, respectively.)

Exchanging Components: Continuous Approximations/PDEs

ML-Space was implemented in a component-based framework and is itself separated into components (cf. Figure 4.4), the major of which are the individual-based, continuous-space and the population based subvolume simulator. The latter, for example, could be exchanged by one using spatial τ-leaping (Jeschke, Ewald, and Uhrmacher 2011) or the (mostly) deterministic, PDE-approximating scheme of Kossow et al. (2015) with appropriate adaptations.

Efficiency Investigations

The current ML-Space simulator should be considered a reference implementation build with maintainability and extensibility in mind. Some possible shortcuts could be taken

to make simulation faster in various practical scenarios.

For example, each subvolume is represented as an own object with a list of pointers to its neighbors (and an associated factor for diffusion involving shared surface and distance). This was build to allow adaptive splitting and uniting of subvolumes as done in Jeschke and Uhrmacher (2008), which was, however, not used in ML-Space so far. Simply using an n-dimensional array to represent the subvolume grid, with neighborhood relations only implicit, may make processing diffusion events faster and allow for more efficient collision detection in hybrid simulation as spatial-entity–subvolume overlap has to be calculated anyway.

Another example involves each entity's attributes: these are stored as name/value-pairs in a dictionary (HashMap) data structure closely following how they are written in the modeling language. Using arrays instead and identifying attributes internally by, for example, their position in the species definition (similar to how the original ML-Rules identifies attributes in the language as well) would make attribute value lookup more efficient and thus speed up rule matching.

6.1.4 Applications

ML-Space' development was significantly influenced by the actin filament investigations, with several features integrated when they were needed for that modeling problem. Possible extensions here include a more detailed regulatory mechanism regarding the phosphorylation of (free and bound) actin and the filament severing (e. g., incorporating results of Beltzner and Pollard 2008; Pollard and Cooper 2009; Ditlev, Mayer, and Loew 2013). However, appropriate spatially resolved experimental data to test such a model's validity will be hard to come by. Extending the model in a more phenomenological way, e. g., to cover the directed growth of filaments towards one end of the cell (on flat surfaces) and its effect on the overall cell, will likely require further targeted simulator adaptations.

The mitochondrial fission model has been extended with a more complex interplay of fission-enabling proteins. We used attributes of mitochondria entities to express the number of bound proteins above. However, more entity states (i. e. possible attribute values) as well as more species and interactions between them, e. g., activation or formation of small complexes, were quite cumbersome to incorporate into this framework.[1] Also, the expression of mitochondrial fusion is rather simplistic so far, and could be extended to cover different shapes of mitochondrial networks observed, i. e. clusters in some cells and long, elongated "networks" in cells treated differently.

Generally, many spatial models currently expressed in other frameworks or specially tailored approaches could also be expressed and simulated in ML-Space, providing a human-readable, easily extensible model description in rule-based form along the way. However, if a model does not benefit from using an attributed language, explicit compartmental structures, or hybrid individual-based and spatial Gillespie simulation, e. g.,

[1] C. Mahler and A. M. Uhrmacher, personal communication

because it contains only a few different species, only one (or two simple, fixed) compartment(s), or is tailored to subvolume-based simulation in the first place, respectively, then simply translating it to ML-Space and simulating it with the current reference implementation will likely be slower than using a simulator that focuses on one of these aspects only. Then again, models where all three are important, i.e. multi-state species (giving rise to combinatorial explosion), hierarchical nesting and multi-resolution simulation, cover a wide array of biological processes and require data from various experimental sources to fit and/or validate. Establishing such a model will itself be a long and complex process. ML-Space now provides a unique tool aiding this process, but must always be open for extensions to cover specific spatial aspects not yet generalized.

6.2 Conclusion

ML-Space is a modeling language and simulation approach targeting spatial aspects in biology, mostly at the cellular level. The attributed, rule-based modeling language incorporates aspects of existing languages, particularly dynamic organizational hierarchies of ML-Rules and explicit binding sites of BioNetGen and Kappa, and combines them with spatial semantics. Spatial simulation can be individual-based, with rigid bodies of different extensions diffusing randomly or drifting in given directions and interacting by collisions. Alternatively, the spatial Gillespie approach of partitioning space into subvolumes can be used, where each subvolume is assumed to contain a well-mixed solution of entities, with reactions among each other and diffusions into neighboring subvolumes.

ML-Space introduces a hybrid simulation approach combining the two, with entities that may contain others or at least take up space not available to other entities represented as individuals and small entities at the lowest organizational level in subvolumes. Since the population-based simulation of subvolumes is less demanding both computationally and in terms of required information, the hybrid approach has several applications:

- With many small entities in a system and crowding not relevant, the hybrid simulator can be orders of magnitude faster than tracking all entities individually as particles in continuous space.
- When details about amounts, sizes or diffusion constants of entities are uncertain or unknown, representing them as dimensionless particles requires fewer assumptions about (or less fitting of) parameters put into the model.

ML-Space has successfully been applied to reproduce existing simulation results (the Hes1 oscillator) and to gain new insights into currently investigated spatial patterns in cell biology (actin filaments and mitochondrial networks). All applications offer potential for further research, and for expanding the expressiveness of the ML-Space language and the capabilities of the simulator.

ML-Space is not the first approach to bring individual- and subvolume-based simulation together. However, it is, to our knowledge, the first to combine individual, mobile

compartments (even dynamic ones w.r.t. creation, destruction and hierarchical composition) and subvolume-based simulation.

A Appendix

A.1 The ML-Space Language's Full Grammar

Listing A.1. ML-Space' grammar in Extended Backus-Naur Form. The final four constructs (`observationTargets` ff.) are not for parts of the model but enable the separate specification of observables like the matching patterns in rules.

```
FLOAT      ::= [0-9]+ | [0-9]+ EXPONENT
             | [0-9]+ '.' [0-9]* EXPONENT? | '.' [0-9]+ EXPONENT?
             | 'Infinity'
EXPONENT   ::= ( 'e' | 'E' ) ( '+' | '-' )? [0-9]+
ID         ::= ( [a-z] | [A-Z] ) ( [a-z] | [A-Z] | [0-9] | '_' )*
STRING     ::= '"' ID '"' | "'" ID "'"

fullmodel
           ::= constant_defs species_defs ( init ( ';' rules )? |
               rules ';' init ) EOF
constant_defs
           ::= ( constant_def ';'? )*
constant_def
           ::= ID '=' valset_or_const
valset_or_const
           ::= interval | range | set | vector | numexpr | STRING
             | ID
attributes_match
           ::= attribute_match ( ',' attribute_match )*
attribute_match
           ::= '(' ID '=' ID var_interval? ')'
             | ID ( '=' ID var_interval? | var_interval )
var_interval
           ::= '==' varexpr | '==' STRING
             | '>' '='? varexpr | '<' '='? varexpr
             | ('IN'|'in') ('('|'[') varexpr (','|'...') varexpr
               (')'|']')
numexpr    ::= expr // involving only numbers and constant identifiers
varexpr    ::= expr // possibly involving local (rule)variable names
expr       ::= multNode ( '+' multNode | '-' multNode )*
```

```
28              | ('if'|'If'|'IF') boolNode ('then'|'Then'|'THEN')
                  expr ( ('else'|'Else'|'ELSE') expr )?
29  multNode ::= atomNode ( '*' atomNode | '/' atomNode )*
30  atomNode ::= ( '-' | '+' )?
31               ( numval | ID | '(' boolNode ')'
32                 | '(' expr ')' | '[' expr ']'
33                 | ('min'|'Min'|'MIN') '(' expr ',' expr ')'
34                 | ('max'|'Max'|'MAX') '(' expr ',' expr ')'
35               ) ( '3' | '3' | '\textcolor{stringmauve}{\degrees}'
                   | '^' atomNode )?
36  boolNode ::= expr compareOp expr
37  compareOp
38            ::= '<' '='? | '>' '='? | '<>' | '=='
39  interval ::= '[' numexpr ( '..' | '...' ) numexpr ']'
40             | '>' '='? numexpr
41             | '<' '='? numexpr
42  range    ::= numexpr ':' numexpr ( ':' numexpr )?
43  set      ::= numset | idset
44  idset    ::= '{' STRING ( ',' STRING )* '}'
45  numset   ::= '{' numexpr ( ',' numexpr )* '}'
46  vector   ::= '(' numexpr ( ',' numexpr )+ ')'
47  intval_or_var
48           ::= numexpr
49  numval   ::= FLOAT | ID // constant identifier, sometimes also variable name
        (context-dependent)
50  species_defs
51           ::= ( species_def ';'? )+
52  species_def
53           ::= ID '(' attributes_def? ')' bindingsitesdef?
54  attributes_def
55           ::= attribute_def ( ',' attribute_def )*
56  attribute_def
57           ::= ID ':' valset_or_const
58  bindingsitesdef
59           ::= '<' bindingsitedef ( ',' bindingsitedef )* '>'
60  bindingsitedef
61           ::= ID ':' ( ID | numexpr )
62  species  ::= ID
63  rules    ::= ( rule ';'? )+
64  rule     ::= ( ID ':' )? rule_left_hand_side ARROW
       rule_right_hand_side? '@' ('p='|'r=')? varexpr
65  rule_left_hand_side
66           ::= entity_match ( ( '[' | ( '+' | '.' ) ) (
                entity_match ( ( '+' | '.' ) entity_match )* )? (
                ']' | ) )?
```

```
67 rule_right_hand_side
68           ::= entity_result ( ( '[' | ( '+' | '.' ) ) (
                 entity_result ( ( '+' | '.' ) entity_result )* )? (
                 ']' | ) )?
69 entity_match
70           ::= species ( '(' attributes_match? ')' )? bindingsites?
71 bindingsites
72           ::= '<' bindingsite ( ',' bindingsite )* '>'
73 bindingsite
74           ::= ID ':' ( entity_match | 'free' | 'occ' | 'occupied'
                 )
75 entities_result
76           ::= ( entity_result ( ( '+' | '.' ) entity_result )* )?
77 entity_result
78           ::= species ( '(' ( ID valmod ( ',' ID valmod )* )? ')'
                 )? bindingactions?
79 valmod    ::= '++' | '--' | op '=' varexpr | '=' varexpr
80         | '=' STRING | '=' ID '(' varexpr ')'
81 op        ::= '+' | '-' | '*' | '/'
82 bindingactions
83           ::= '<' bindingaction ( ',' bindingaction )* '>'
84 bindingaction
85           ::= ID ':' ( 'bind' | 'release' | 'replace' )
86 init      ::= init_element ( ( '+' | '.' ) init_element )*
87 init_element
88           ::= for_each
89             | intval_or_var ( entity_result | '[' entities_result
                  ']' ) ( '[' init ']' )? ';'?
90 for_each  ::= ( ( 'FOR' | 'for' ) for_var '{' init '}' )+
91 for_var   ::= ID '=' ( range | set )
92
93 observationTargets
94           ::= obsTargetEntry ( ';' obsTargetEntry )*
95 obsTargetEntry
96           ::= obs_matches ( ( 'IN' | 'in' ) obs_matches )* ( '='
                 idlist )?
97 obs_matches
98           ::= ( entity_match ( ',' entity_match )* )?
99 idlist    ::= ( ID | '#' ) ( ',' ID | '#' )*
```

A.2 The Mitochondial Network Model

Listing A.2. Mitochondrial network model used in section 5.3 and Bittig et al. (2014b)

```
// "Optimal Dynamics for Quality Control in Spatially Distributed Mitochondrial Networks" accoding to Patel, Shiriha, Huang; PLoS Comp Biol 9(7), doi:10.1371/journal.pcbi.1003108, adapted and extended with Fis1 and Drp1 for CMSB submission 39 (2014)

//// Constants ////
maxHealth = 10;
minNewHealth = maxHealth/2;

ExchangeDirSwitch = 0; // 1 -> up, 0 -> either, -1 -> down, -1.3 -> up and down
pUp = [(ExchangeDirSwitch+0.25)^2];
pDown = [(-ExchangeDirSwitch+1)/2];
pEither = [1-ExchangeDirSwitch^2];

skipSameUp = 0; // flag (0 or 1) whether to skip "up"ward "healing" in case of same-health collision
skipSameDown = 0;

rDamage = 5*10^-4;
// rFusion = 0.1; // fusion probabilities in case of collisions used instead
// rFission = 0.1; // superseded by nFis & nDrp expressions
rAutophagy = 3.33*10^-3;
rMicrotubeAttach = 0.2; // not clear from original paper
rMicrotubeDetach = 0.2; // not clear from original paper

// all lengths in micro-m, all times in min (thus speed in micro-m/min)
mitoRadius = 0.5;
mitoDiff = 0.0;
mitoSpeed = 0.5;
healthExchange = 2;
cellSideLength = 25;
mitoNum = 150;
mitoNumVariability = 1;

rReplication = 0.02 * mitoNum / cellSideLength^2; // adjusted for 0th-order reaction instead of 1st-order

maxFis = 8;
maxDrp = 2;
```

```
35 rFisRecruitment = 1.;
36 rDrpRecruitment = 1.;
37 DrpRecThreshold = 4;
38
39
40 fusionThreshold = 0.3*maxHealth;
41 autophagyThreshold = 0.3*maxHealth;
42 maxHealthForExchange = maxHealth-healthExchange;
43
44 switchFused = 2; // 0: no damage or autophagy of fused mitos, 1: damage,
       but no autophagy, 2: damage & autophagy (incl. more likely fission)
45 switchFusedDamage = [(switchFused+1)/2]; // min(switchFused,1);
46 switchFusedAutophagy = [switchFused/2]; // max(switchFused,0);
47
48 fisdrpFactor = 1;
49 fisNum = mitoNum * maxFis * fisdrpFactor;
50 drpNum = mitoNum * maxDrp * fisdrpFactor;
51
52 tau = 2*PI;
53
54
55 //// Species definitions ////
56 Cell(shape:square,size:cellSideLength^2, nm:0,nFis:fisNum,nDrp:drpNum);
57 Mito(shape:circle,size:PI*mitoRadius^2,
       diffusion:mitoDiff,direction:[0...tau],velocity:[0...mitoSpeed],
       health:minNewHealth:maxHealth,nFis:0,nDrp:0)<bs:0>;
58
59
60 //// Initial state ////
61 1 Cell(position:(0,0),nm:mitoNum)[
62    mitoNum-1 Mito(velocity:0) + 1 Mito(velocity:mitoSpeed,direction:PI)
63 ];
64
65 //// Rules ////
66
67 // microtubule attachment and detatchment rules
68 Mito(velocity>0) -> Mito(velocity=0) @ rMicrotubeDetach
69 Mito(velocity==0)<bs:free> -> Mito(velocity=mitoSpeed,direction:[0...tau])
       @ rMicrotubeAttach
70
71 // fusion rules (incl. health unit exchange)
72 Mito(h=health>=fusionThreshold)<bs:free> + Mito(health in
       [h+skipSameUp...maxHealthForExchange])<bs:free>
73    -> Mito(velocity=0,health-=healthExchange)<bs:bind>
          .Mito(velocity=0,health+=healthExchange)<bs:bind> @ pUp
```

```
74 Mito(h=health<=maxHealth)<bs:free> + Mito(health in
       [fusionThreshold...min(h-skipSameDown,maxHealthForExchange)])<bs:free>
75    -> Mito(velocity=0,health-=healthExchange)<bs:bind>
          .Mito(velocity=0,health+=healthExchange)<bs:bind> @ pDown
76 Mito(health>=fusionThreshold)<bs:free> +
       Mito(health<=maxHealthForExchange)<bs:free>
77    -> Mito(velocity=0,health-=healthExchange)<bs:bind>
          .Mito(velocity=0,health+=healthExchange)<bs:bind> @ pEither
78
79 // Fis and Drp recruitment
80 Cell(cf=nFis>0)[Mito(f=nFis<maxFis)<bs:Mito(nFis<maxFis-f)>] ->
       Cell(nFis-=1)[Mito(nFis+=1)] @ rFisRecruitment * cf
81 Cell(cd=nDrp>0)[Mito(f=nFis,d=nDrp)
       <bs:Mito(nFis>=DrpRecThreshold-f,nDrp<maxDrp-d)> ] ->
       Cell(nDrp-=1)[Mito(nDrp+=1)] @ rDrpRecruitment * cd
82
83 // fission; Mito().Mito() ->...would be interpreted as 2nd-order rule
84 Cell[Mito(f=nFis,d=nDrp)<bs:Mito(nFis>=maxFis-f,nDrp>=maxDrp-d)>] ->
       Cell(nFis+=f,nDrp+=d)[Mito(nFis=0,nDrp=0)<bs:release>] @ Infinity
85 Cell[Mito(f=nFis>0)<bs:free>] -> Cell(nFis+=f)[Mito(nFis=0)] @ Infinity
86 Cell[Mito(d=nDrp>0)<bs:free>] -> Cell(nDrp+=d)[Mito(nDrp=0)] @ Infinity
87
88 // damage
89 Mito(h=health>0)<bs:free> -> Mito(health-=1) @ rDamage*h
90 Mito(h=health>0)<bs:occ> -> Mito(health-=1) @ rDamage*h*switchFusedDamage
91
92 // autophagy and biogenesis
93 Cell[Mito(health<autophagyThreshold)<bs:free>] -> Cell(nm-=1)[] @
       rAutophagy
94 Mito(health<autophagyThreshold)<bs:occ> -> Mito(health=-1)<bs:release> @
       rAutophagy*switchFusedAutophagy
95 Cell[Mito(f=nFis,d=nDrp,health==-1)] -> Cell(nm-=1,nDrp+=d,nFis+=f)[] @
       Infinity
96 Cell(x=nm)[] -> Cell(nm+=1)[Mito(velocity=0)] @ rReplication * 1/(1+e^((x
       - mitoNum)/mitoNumVariability))
```

A.3 The Actin Filament Model

Listing A.3. Actin filament growth model used in section 5.4 and Bittig et al. (2014a). Updated to current syntax of ML-Space compared to (Bittig et al. 2014a, add. file 2).

```
// "Membrane related dynamics and the formation of actin in cells growing
    on micro-topographies: a spatial computational model" BMC Systems
    Biology 2014, 8:106 -- http://www.biomedcentral.com/1752-0509/8/106
    doi:10.1186/s12918-014-0106-2

// system surface structure properties
surfStructSwitch = 1; // 0: flat, 1: pillars, 2: groves
sysFlat = [1-surfStructSwitch / 2];
sysPillar = 1-(surfStructSwitch-1)^2;
sysGrove = [surfStructSwitch / 2];

// reaction probabilities/rates
filFormProb = 1.;
pActBind = 1.;
pActRelease = .3; // probability of cofilin hitting filament leading to
    dissolution
rFilDissolution = Infinity;
rActinProd = 1; // two separate rate constants used later
rActinProdSys = 1 * rActinProd;
rActinProdSurf = 1 * rActinProd;
pCofDeact = 0.9; // probability of Cofilin deactivation on collision with
    regulating entity
rCofReact = 0.1; // rate of cofilin reactivation ("no" external trigger)
rIntActivation = 9.;
rIntDeactivation = 3;

skipCofRegSwitch = 0;
cofRegProdMechanismSwitch = 1; // boolean parameter:
// CofReg always in System, activated on contact with Integrin,
    deactivated stochastically anywhere,
// or CofReg appearing near Integrin, disappearing stochastically anywhere
rCofRegProdAtInt = 0.5 * cofRegProdMechanismSwitch * skipCofRegSwitch;
rCofRegDegr = 2. / rCofRegProdAtInt *
    cofRegProdMechanismSwitch*skipCofRegSwitch;
rCofRegDeact = 0.7 * (1-cofRegProdMechanismSwitch) * skipCofRegSwitch;
pCofRegAct = 1 * (1-cofRegProdMechanismSwitch) * skipCofRegSwitch;
pCofDeactAtInt = pCofDeact * skipCofRegSwitch;

// species size/movement constants
actinDiam = 0.01;
```

```
actinBaseSize = PI*actinDiam^2;
actinDiff = 0.3;
integrinSize = actinBaseSize / 377 * 988; // 788-1188
integrinDiff = 0.05;
cofSize = actinBaseSize / 377 * 166;
cofDiff = 0.6;
arpDiff = actinDiff;
arpSize = actinBaseSize / 377 * 406; // 394-418
cofRegDiff = (actinDiff + cofDiff) / 2;
cofRegSize = (actinBaseSize + cofSize) / 2;
actinScaling = 1;
actinSize = actinBaseSize * actinScaling; // for separate overriding

// initial amounts
totalActin = 1000; // total; "no"t all initially on surface structure
totalIntegrin = 200;
totalCofilin = 400;
totalArp = 50; // branch-enabling entity
totalCofReg = 200; // actual number dependent on switches, see next line
numCofReg = totalCofReg * (1-cofRegProdMechanismSwitch) * skipCofRegSwitch;

// system size properties
shortSize = 2;
longSize = 10;
gapSize = 0.5;
sysSize = (longSize+2*gapSize)^2;

pillarNum = 9;
idxP = ((pillarNum^0.5)-1)/2;
ridgeNum = 3;
idxR = (ridgeNum-1)/2;

// information for simulator:
initialAbsoluteAngle = 0; // horizontal default filament orientation
angleDevDeg = 30; // specified in degrees for easier overriding
initialAbsoluteAngleDeviation = angleDevDeg/180*PI; // degree->radians
    conversion

postponedRegionInit = 1; // simulator switch for distributing entities
// initially defined outside of pillar regions between _and_ on them


// species definitions:
SurfStruct(shape:rectangle,aspectratio:(1,1),boundary:soft,size:[0...sysSize]);
System(shape:square,size:sysSize);
```

```
78
79  Actin(shape:circle,size:actinSize,diffusion:[0...actinDiff])
        <pointed:0,barbed:180/180*PI,branch1:110/180*PI,branch2:250/180*PI>;
        // always bind-ready
80  Integrin(shape:circle,size:integrinSize,diffusion:[0...integrinDiff],
        active:{"yes","no"})<bs:0>; // active only
81  Arp23s(shape:circle,size:arpSize,diffusion:[0...arpDiff])
        <fil:0,straight:180/180*PI>; // for side branching
82  Cofilin(shape:circle,size:cofSize,diffusion:cofDiff, active:{"yes","no"});
83  CofReg(shape:circle,size:cofRegSize,diffusion:cofRegDiff,
        active:{"yes","no"});
84
85
86  // initial state definition
87  0+sysPillar System(position:(0,0))[
88      totalActin Actin(diffusion:actinDiff) +
89      totalCofilin Cofilin(active:"yes") +
90      totalIntegrin Integrin(diffusion:integrinDiff,active:"no") +
91      totalArp Arp23s(diffusion:arpDiff) +
92      FOR x=-idxP:idxP {
93        FOR y=-idxP:idxP {
94          1 SurfStruct(size:shortSize^2,relpos:(x*2*shortSize,y*2*shortSize))
95        }
96    }]
97
98   +sysFlat System(position:(0,0))[
99      totalActin Actin(diffusion:actinDiff) +
100     totalCofilin Cofilin(active:"yes") +
101     totalIntegrin Integrin(diffusion:integrinDiff,active:"no") +
102     totalArp Arp23s(diffusion:arpDiff) +
103     1 SurfStruct(size:longSize^2,relpos:(0,0))
104 ]
105
106  +sysGrove System(position:(0,0))[
107     totalActin Actin(diffusion:actinDiff) +
108     totalCofilin Cofilin(active:"yes") +
109     totalIntegrin Integrin(diffusion:integrinDiff,active:"no") +
110     totalArp Arp23s(diffusion:arpDiff) +
111     FOR y=-idxR:idxR {
112       1 SurfStruct(size:shortSize*longSize,
              aspectratio:(longSize,shortSize),relpos:(0,y*2*shortSize))
113 }];
114
115
116 // transfer rules: unimpeded migration onto/off surface structures
```

```
117 Actin() + SurfStruct() -> SurfStruct()[Actin()] @ 1
118 SurfStruct()[Actin()] -> Actin() + SurfStruct() @ 1
119 Cofilin() + SurfStruct() -> SurfStruct()[Cofilin()] @ 1
120 SurfStruct()[Cofilin()] -> Cofilin() + SurfStruct() @ 1
121 Integrin() + SurfStruct() -> SurfStruct()[Integrin()] @ 1
122 SurfStruct()[Integrin()] -> Integrin() + SurfStruct() @ 1
123 Arp23s() + SurfStruct() -> SurfStruct()[Arp23s()] @ 1
124 SurfStruct()[Arp23s()] -> Arp23s() + SurfStruct() @ 1
125
126 SurfStruct()[Integrin(active=="no")] ->
        SurfStruct()[Integrin(active="yes",diffusion=0)] @ rIntActivation
127 Integrin(active=="yes")<bs:free> ->
        Integrin(active="no",diffusion=integrinDiff) @ rIntDeactivation
128
129 // filament formation:
130 Actin()<pointed:free> + Integrin(active=="yes")<bs:free>
131    -> Actin(diffusion=0)<pointed:bind>.Integrin(diffusion=0)<bs:bind> @
           pActBind
132 Actin()<pointed:free> + Actin()<pointed:occ,barbed:free>
133    -> Actin(diffusion=0)<pointed:bind>.Actin()<barbed:bind> @ filFormProb
134
135 // Cofilin and CofReg activity regulation
136 Integrin(active=="yes") -> Integrin() + CofReg(active="yes") @
        rCofRegProdAtInt
137 CofReg() -> @ rCofRegDegr
138 SurfStruct()[Integrin(active=="yes") + CofReg(active=="no")] ->
        SurfStruct()[Integrin() + CofReg(active="yes")] @ pCofRegAct
139 CofReg(active=="yes") -> CofReg(active="no") @ rCofRegDeact
140
141 Integrin(active=="yes") + Cofilin(active=="yes") -> Integrin() +
        Cofilin(active="no") @ pCofDeactAtInt
142
143 CofReg(active=="yes") + Cofilin(active=="yes") -> CofReg() +
        Cofilin(active="no") @ pCofDeact
144 Cofilin(active=="no") -> Cofilin(active="yes") @ rCofReact
145
146 // filament destruction
147 Cofilin(active=="yes") + Actin()<pointed:occ>
148    -> Cofilin() +
           Actin(diffusion=actinDiff)<pointed:release,barbed:release> @
           pActRelease
149 Actin()<pointed:free,barbed:occ> ->
        Actin(diffusion=actinDiff)<barbed:release> @rFilDissolution
150 // cutoff filament becoming mobile again
```

```
151 Actin(diffusion==0)<pointed:free, barbed:free> ->
        Actin(diffusion=actinDiff) @ Infinity
152 // Integrin(diffusion==0)<bs:free> -> Integrin(diffusion=integrinDiff) @
        Infinity
153 // not if integrin complex is immobile itself, see above
154
155 // actin creation
156 System()[] -> System()[Actin(diffusion=actinDiff)] @ rActinProdSys
157 SurfStruct()[] -> SurfStruct()[Actin(diffusion=actinDiff)] @ rActinProdSurf
158
159 // side branching
160 Arp23s()<fil:free> + Actin()<pointed:occ,branch1:free,branch2:free>
161     -> Arp23s(diffusion=0)<fil:bind>.Actin()<branch1:bind> @ 1
162 Arp23s()<fil:free> + Actin()<pointed:occ,branch1:free,branch2:free>
163     -> Arp23s(diffusion=0)<fil:bind>.Actin()<branch2:bind> @ 1
164 Actin()<pointed:free> + Arp23s()<fil:occ,straight:free>
165     -> Actin(diffusion:0)<pointed:bind>.Arp23s()<straight:bind> @ 1
166 Arp23s(diffusion==0)<fil:free,straight:free> -> Arp23s(diffusion=arpDiff)
        @ Infinity
```

Bibliography

Adcock, Stewart A. and J. Andrew McCammon (2006). "Molecular Dynamics:âĂĽ Survey of Methods for Simulating the Activity of Proteins". In: *Chemical reviews* 106.5, pp. 1589–1615. ISSN: 0009-2665. DOI: 10.1021/cr040426m (p. 17).

Alber, Mark S., Maria A. Kiskowski, James A. Glazier, and Yi Jiang (2002). "On Cellular Automaton Approaches to Modeling Biological Cells". In: *In IMA Mathematical Systems Theory in Biology, Communication, and Finance*. Vol. 134, pp. 1–39. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.3334 (p. 19).

Andrews, Steve (2016). *Smoldyn User's Manual for Smoldyn version 2.47*. URL: http://www.smoldyn.org/SmoldynUsersManual.pdf (p. 27).

Andrews, Steven S. (2012). "Spatial and stochastic cellular modeling with the Smoldyn simulator". In: *Methods in molecular biology (Clifton, N.J.)* 804, pp. 519–542. ISSN: 1940-6029. DOI: 10.1007/978-1-61779-361-5_26 (p. 80).

Andrews, Steven S. and Dennis Bray (2004). "Stochastic simulation of chemical reactions with spatial resolution and single molecule detail." In: *Physical biology* 1.3-4, pp. 137–151. ISSN: 1478-3967. DOI: 10.1088/1478-3967/1/3/001 (pp. 17, 21, 80).

Andrews, Steven S., Satya N. V. Arjunan, Gianfranco Balbo, Arne T. Bittig, Jerome Feret, Kazunari Kaizu, and Fei Liu (2015). "Simulating macromolecular crowding with particle and lattice-based methods". In: *Multiscale Spatial Computational Systems Biology (Dagstuhl Seminar 14481)*. Ed. by David Gilbert, Monika Heiner, Koichi Takahashi, and Adelinde M. Uhrmacher. Vol. 4. Dagstuhl Reports 11. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. Chap. 4.1, pp. 170–187. ISBN: 2192-5283. URL: http://drops.dagstuhl.de/opus/volltexte/2015/4972/pdf/dagrep_v004_i011_p138_s14481.pdf#page=33. DOI: 10.4230/DagRep.4.11.138 (p. 21).

Anselme, K., P. Linez, M. Bigerelle, D. Le Maguer, A. Le Maguer, P. Hardouin, H. F. Hildebrand, A. Iost, and J. M. Leroy (2000). "The relative influence of the topography and chemistry of TiAl6V4 surfaces on osteoblastic cell behaviour." In: *Biomaterials* 21.15, pp. 1567–1577. ISSN: 0142-9612. URL: http://view.ncbi.nlm.nih.gov/pubmed/10885729 (p. 94).

ANTLR, ANother Tool for Language Recognition, v3. http://www.antlr3.org/ (p. 75).

Arjunan, Satya N. V. and Masaru Tomita (2010). "A new multicompartmental reaction-diffusion modeling method links transient membrane attachment of E. coli MinE to E-ring formation". In: *Systems and Synthetic Biology* 4.1, pp. 35–53. DOI: 10.1007/s11693-009-9047-2 (pp. 19, 27).

Arvo, James (1990). "A simple method for box-sphere intersection testing". In: *Graphics Gems*. Ed. by Andrew S. Glassner. San Diego, CA, USA: Academic Press Professional, Inc. Chap. A Simple Method for Box-sphere Intersection Testing, pp. 335–339. ISBN: 0-12-286169-5. URL: https://books.google.de/books?id=Mqn8BAAAQBAJ&pg=PA335 (p. 79).

Auger, Anne, Philippe Chatelain, and Petros Koumoutsakos (2006). "R-leaping: Accelerating the stochastic simulation algorithm by reaction leaps". In: *The Journal of Chemical Physics* 125.8, pp. 084103+. ISSN: 0021-9606. DOI: 10.1063/1.2218339 (p. 15).

Bar-Yam, Yaneer (2011). *Concepts: Separation of Scales*. website. accessed 2015-07-15. URL: http://necsi.edu/guide/concepts/scale_separation.html (p. vii).

Baras, F. and M. Malek Mansour (1996). "Reaction-diffusion master equation: A comparison with microscopic simulations". In: *Physical Review E* 54.6, pp. 6139–6148. DOI: 10.1103/physreve.54.6139 (p. 19).

Bartocci, E., F. Corradini, M. R. Di Berardini, E. Merelli, and L. Tesei (2010a). "Shape Calculus. A Spatial Mobile Calculus for 3D Shapes". In: *Scientific Annals of Computer Science* 20, pp. 1–31. URL: http://www.info.uaic.ro/bin/download/Annals/XX/XX_0.pdf (p. 24).

Bartocci, Ezio, Diletta R. Cacciagrano, Maria R. Di Berardini, Emanuela Merelli, and Luca Tesei (2010b). "Shape Calculus: Timed Operational Semantics and Well-formedness". In: *Scientific Annals of Computer Science* 20, pp. 33–52. URL: http://arxiv.org/abs/1011.2488. arXiv: 1011.2488.

Beccuti, Marco, Mary A. Blätke, Martin Falk, Simon Hardy, Monika Heiner, Carsten Maus, Sebastian Nähring, and Christian Rohr (2015). "Dictyostelium discoideum: Aggregation and Synchronisation of Amoebas in Time and Space". In: *Multiscale Spatial Computational Systems Biology (Dagstuhl Seminar 14481)*. Ed. by David Gilbert, Monika Heiner, Koichi Takahashi, and Adelinde M. Uhrmacher. Vol. 4. Dagstuhl Reports 11. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. Chap. 4.7, pp. 195–214 (p. 114).

Beltzner, Christopher C. and Thomas D. Pollard (2008). "Pathway of Actin Filament Branch Formation by Arp2/3 Complex". In: *Journal of Biological Chemistry* 283.11, pp. 7135–7144. ISSN: 1083-351X. DOI: 10.1074/jbc.m705894200 (pp. 96–98, 118).

Bernstein, David (2005). "Simulating mesoscopic reaction-diffusion systems using the Gillespie algorithm". In: *Physical Review E* 71.4, pp. 041103+. ISSN: 1539-3755. DOI: 10.1103/physreve.71.041103 (p. 35).

Bershadsky, Alexander, Michael Kozlov, and Benjamin Geiger (2006). "Adhesion-mediated mechanosensitivity: a time to experiment, and a time to theorize." In: *Current Opinion in Cell Biology* 18.5, pp. 472–481. DOI: 10.1016/j.ceb.2006.08.012 (p. 94).

Bittig, Arne T., Matthias Jeschke, and Adelinde M. Uhrmacher (2010). "Towards Modelling and Simulation of Crowded Environments in Cell Biology". In: *ICNAAM 2010: International Conference of Numerical Analysis and Applied Mathematics*. Ed. by

Theodore E. Simos, George Psihoyios, and Ch Tsitouras. Vol. 1281. 1. Rhodes, Greece: AIP, pp. 1326–1329. DOI: 10.1063/1.3497964 (p. 7).

Bittig, Arne T. and Adelinde M. Uhrmacher (2010). "Spatial Modeling in Cell Biology at Multiple Levels". In: *Proceedings of the 2010 Winter Simulation Conference*. Ed. by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan. Baltimore, MD, USA: IEEE Computer Science, pp. 608–619. DOI: 10.1109/WSC.2010.5679125 (pp. ix, 7, 15).

– (2016). "ML-Space: Hybrid Spatial Gillespie and Particle Simulation of Multi-level Rule-based Models in Cell Biology". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* PP. ISSN: 1545-5963. DOI: 10.1109/TCBB.2016.2598162 (pp. ix, 6, 7, 83).

Bittig, Arne T., Fiete Haack, Carsten Maus, and Adelinde M. Uhrmacher (2011). "Adapting rule-based model descriptions for simulating in continuous and hybrid space". In: *Proceedings of the 9th International Conference on Computational Methods in Systems Biology*. CMSB '11. Paris, France: ACM, pp. 161–170. ISBN: 978-1-4503-0817-5. DOI: 10.1145/2037509.2037533 (pp. 7, 83).

Bittig, Arne T., Claudia Matschegewski, Barbara Nebe, and Adelinde M. Uhrmacher (2012). "Spatial Simulation of Actin Filament Dynamics on Structured Surfaces". In: *Winter Simulation Conference* (p. 83).

Bittig, Arne T., Claudia Matschegewski, J. Barbara Nebe, Susanne Stählke, and Adelinde M. Uhrmacher (2014a). "Membrane related dynamics and the formation of actin in cells growing on micro-topographies: a spatial computational model". In: *BMC Systems Biology* 8, pp. 106+. ISSN: 1752-0509. DOI: 10.1186/s12918-014-0106-2 (pp. x, 7, 83, 94, 97, 106, 127).

Bittig, Arne T., Florian Reinhardt, Simone Baltrusch, and Adelinde M. Uhrmacher (2014b). "Predictive Modelling of Mitochondrial Spatial Structure and Health". In: *Computational Methods in Systems Biology*. Ed. by Pedro Mendes, Joseph O. Dada, and Kieran Smallbone. Vol. 8859. Lecture Notes in Computer Science. Springer International Publishing. Chap. 20, pp. 252–255. ISBN: 978-3-319-12981-5. DOI: 10.1007/978-3-319-12982-2_20 (pp. ix, x, 7, 83, 124).

Blinov, Michael L., James R. Faeder, Byron Goldstein, and William S. Hlavacek (2004). "BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains." In: *Bioinformatics (Oxford, England)* 20.17, pp. 3289–3291. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bth378 (pp. 25, 50).

Bortz, A. B., M. H. Kalos, and J. L. Lebowitz (1975). "A new algorithm for Monte Carlo simulation of Ising spin systems". In: *Journal of Computational Physics* 17.1, pp. 10–18. ISSN: 00219991. DOI: 10.1016/0021-9991(75)90060-1 (pp. 4, 11).

Box, G. E. P. and Mervin E. Muller (1958). "A Note on the Generation of Random Normal Deviates". In: *The Annals of Mathematical Statistics* 29.2, pp. 610–611. ISSN: 0003-4851. DOI: 10.1214/aoms/1177706645 (p. 78).

Brakebusch, Cord and Reinhard Fässler (2003). "The integrin-actin connection, an eternal love affair." In: *The EMBO journal* 22.10, pp. 2324–2333. ISSN: 0261-4189. DOI: 10.1093/emboj/cdg245 (p. 96).

Brown, Randy (1988). "Calendar Queues: A Fast O(1) Priority Queue Implementation for the Simulation Event Set Problem". In: *Communications of the ACM* 31.10, pp. 1220–1227. ISSN: 0001-0782. DOI: 10.1145/63039.63045 (pp. 75, 77, 78).

Burrage, Kevin, PamelaM Burrage, André Leier, Tatiana Marquez-Lago, and DanV Nicolau (2011). "Stochastic Simulation for Spatial Modelling of Dynamic Processes in a Living Cell". In: *Design and Analysis of Biomolecular Circuits*. Ed. by Heinz Koeppl, Gianluca Setti, Mario di Bernardo, and Douglas Densmore. Springer New York. Chap. 2, pp. 43–62. ISBN: 978-1-4419-6765-7. DOI: 10.1007/978-1-4419-6766-4_2 (p. 28).

Calderwood, David A. and Mark H. Ginsberg (2003). "Talin forges the links between integrins and actin." In: *Nature cell biology* 5.8, pp. 694–697. ISSN: 1465-7392. DOI: 10.1038/ncb0803-694 (p. 96).

Cao, Yang, Daniel T. Gillespie, and Linda R. Petzold (2005). "The slow-scale stochastic simulation algorithm". In: *The Journal of Chemical Physics* 122.1, pp. 014116+. ISSN: 0021-9606. DOI: 10.1063/1.1824902 (p. 15).

Cao, Yang, Hong Li, and Linda Petzold (2004). "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems". In: *The Journal of Chemical Physics* 121.9, pp. 4059–4067. ISSN: 0021-9606. DOI: 10.1063/1.1778376 (pp. 12, 15).

Cardelli, Luca (2005). "Brane Calculi". In: *Computational Methods in Systems Biology*. Ed. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Vincent Danos, and Vincent Schachter. Vol. 3082. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. 24, pp. 257–278. ISBN: 978-3-540-25375-4. URL: http://www.springerlink.com/content/yrnafekrwqr9n1mk. DOI: 10.1007/978-3-540-25974-9_24 (p. 24).

Cardelli, Luca and Andrew D. Gordon (2000). "Mobile ambients". In: *Theoretical Computer Science* 240.1, pp. 177–213. ISSN: 03043975. DOI: 10.1016/s0304-3975(99)00231-5 (p. 23).

Carlsson, A. E. (2006). "Stimulation of Actin Polymerization by Filament Severing". In: *Biophysical Journal* 90.2, pp. 413–422. ISSN: 00063495. DOI: 10.1529/biophysj.105.069765 (pp. 96, 98).

Chylek, Lily A., Leonard A. Harris, James R. Faeder, and William S. Hlavacek (2015). "Modeling for (physical) biologists: an introduction to the rule-based approach". In: *Physical Biology* 12.4, pp. 045007+. ISSN: 1478-3975. DOI: 10.1088/1478-3975/12/4/045007 (p. 4).

Ciocchetta, F. and Jane Hillston (2008). "Bio-PEPA: An Extension of the Process Algebra PEPA for Biochemical Networks". In: *Electronic Notes in Theoretical Computer Science* 194.3, pp. 103–117. ISSN: 15710661. DOI: 10.1016/j.entcs.2007.12.008 (p. 24).

Ciocchetta, Federica and Maria L. Guerriero (2009). "Modelling Biological Compartments in Bio-PEPA". In: *Electron. Notes Theor. Comput. Sci.* 227, pp. 77–95. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2008.12.105 (p. 24).

Ciocchetta, Federica and Jane Hillston (2009). "Bio-PEPA: A framework for the modelling and analysis of biological systems". In: *Theoretical Computer Science* 410.33-34, pp. 3065–3084. ISSN: 03043975. DOI: 10.1016/j.tcs.2009.02.037 (p. 28).

Collins, Frank C. and George E. Kimball (1949). "Diffusion-controlled reaction rates". In: *Journal of Colloid Science* 4.4, pp. 425–437. ISSN: 0095-8522. DOI: 10.1016/0095-8522(49)90023-9 (p. 21).

Danos, Vincent, Jérôme Feret, Walter Fontana, Russell Harmer, and Jean Krivine (2007). "Rule-based Modelling of Cellular Signalling". In: *Proceedings of the 18 th International Conference on Concurrency Theory (CONCUR'07), Lecture Notes in Computer Science*. Vol. 4703, pp. 17–41. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.107.228 (pp. 25, 30).

Danos, Vincent, Jérôme Feret, Walter Fontana, Russ Harmer, and Jean Krivine (2009). "Rule-Based Modelling and Model Perturbation". In: *Transactions on Computational Systems Biology XI*. Ed. by Corrado Priami, Ralph-Johan Back, and Ion Petre. Vol. 5750. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. 6, pp. 116–137. ISBN: 978-3-642-04185-3. DOI: 10.1007/978-3-642-04186-0_6 (p. 25).

De Back, Walter, Andreas Deutsch, Dirk Drasdo, Akira Funahashi, and Adelinde M. Uhrmacher (2015). "Towards a standard exchange format for spatial, multilevel multicellular models". In: *Multiscale Spatial Computational Systems Biology (Dagstuhl Seminar 14481)*. Ed. by David Gilbert, Monika Heiner, Koichi Takahashi, and Adelinde M. Uhrmacher. Vol. 4. Dagstuhl Reports 11. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. Chap. 4.5, pp. 214–225. URL: http://drops.dagstuhl.de/opus/volltexte/2015/4972/pdf/dagrep_v004_i011_p138_s14481.pdf#page=77. DOI: 10.4230/DagRep.4.11.138 (p. 23).

Degenring, Daniela, Mathias Röhl, and Adelinde M. Uhrmacher (2004). "Discrete event, multi-level simulation of metabolite channeling". In: *Bio Systems* 75.1-3, pp. 29–41. ISSN: 03032647. DOI: 10.1016/j.biosystems.2004.03.008 (p. 27).

Deutsch, Andreas and Sabine Dormann (2004). *Cellular Automaton Modeling of Biological Pattern Formation*. 2005th ed. Basel: Birkhäuser. ISBN: 0817642811. URL: http://www.worldcat.org/isbn/0817642811 (pp. 18, 26, 98).

Dewdney, Alexander K. (1984). "Sharks and fish wage an ecological war on the toroidal planet Wa-Tor". In: *Scientific American* 251.6, pp. 14–22 (p. 18).

Discher, Dennis E., Paul Janmey, and Yu-Li L. Wang (2005). "Tissue cells feel and respond to the stiffness of their substrate." In: *Science (New York, N.Y.)* 310.5751, pp. 1139–1143. ISSN: 1095-9203. DOI: 10.1126/science.1116995 (p. 94).

Ditlev, Jonathon A., Bruce J. Mayer, and Leslie M. Loew (2013). "There is more than one way to model an elephant. Experiment-driven modeling of the actin cytoskeleton." In: *Biophysical journal* 104.3, pp. 520–532. ISSN: 1542-0086. URL: http://view.ncbi.nlm.nih.gov/pubmed/23442903 (pp. 96–98, 118).

Ditlev, Jonathon A., Nathaniel M. Vacanti, Igor L. Novak, and Leslie M. Loew (2009). "An open model of actin dendritic nucleation." In: *Biophysical journal* 96.9, pp. 3529–3542. ISSN: 1542-0086. DOI: 10.1016/j.bpj.2009.01.037 (pp. 96, 98, 102).

Doob, J. L. (1945). "Markoff Chains – Denumerable Case". In: *Transactions of the American Mathematical Society* 58.3, pp. 455–473. URL: http://www.jstor.org/stable/1990339 (p. 11).

Drawert, Brian, Stefan Engblom, and Andreas Hellander (2012). "URDME: a modular framework for stochastic simulation of reaction-transport processes in complex geometries". In: *BMC Systems Biology* 6.1, pp. 76+. ISSN: 1752-0509. DOI: 10.1186/1752-0509-6-76 (p. 84).

Dumbauld, David W., Ted T. Lee, Ankur Singh, Jan Scrimgeour, Charles A. Gersbach, Evan A. Zamir, Jianping Fu, Christopher S. Chen, Jennifer E. Curtis, Susan W. Craig, and Andrés J. García (2013). "How vinculin regulates force transmission". In: *Proceedings of the National Academy of Sciences* 110.24, pp. 9788–9793. ISSN: 1091-6490. DOI: 10.1073/pnas.1216209110 (p. 96).

Eichner, Christian, Arne Bittig, Heidrun Schumann, and Christian Tominski (2014). "Analyzing simulations of biochemical systems with feature-based visual analytics". In: *Computers & Graphics* 38, pp. 18–26. ISSN: 00978493. DOI: 10.1016/j.cag.2013.09.001 (p. 8).

Elf, J. and M. Ehrenberg (2004). "Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases". In: *Systems Biology* 1.2, pp. 230–236. ISSN: 1741-2471. DOI: 10.1049/sb:20045021 (pp. 5, 19, 46, 49, 72, 76).

Elf, Johan, Andreas Doncic, and Mans Ehrenberg (2003). "Mesoscopic reaction-diffusion in intracellular signaling". In: *SPIE's First International Symposium on Fluctuations and Noise*. Ed. by Sergey M. Bezrukov, Hans Frauenfelder, and Frank Moss. Vol. 5110. Proc. SPIE. Society of Photo-Optical Instrumentation Engineers (SPIE). Santa Fe, NM: SPIE, pp. 114–124. DOI: 10.1117/12.497009 (p. 19).

Ericson, Christer (2004). *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology)*. HAR/CDR. CRC Press. ISBN: 1558607323. URL: http://www.worldcat.org/isbn/1558607323 (p. 80).

Ermak, Donald L. and J. A. McCammon (1978). "Brownian dynamics with hydrodynamic interactions". In: *The Journal of Chemical Physics* 69.4, pp. 1352–1360. DOI: 10.1063/1.436761 (p. 17).

Etienne-Manneville, Sandrine and Alan Hall (2002). "Rho GTPases in cell biology." In: *Nature* 420.6916, pp. 629–635. ISSN: 0028-0836. DOI: 10.1038/nature01148 (p. 96).

Ewald, R., J. Himmelspach, M. Jeschke, S. Leye, and A. M. Uhrmacher (2010). "Flexible experimentation in the modeling and simulation framework JAMES II–implications for computational systems biology". In: *Briefings in Bioinformatics* 11.3, pp. 290–300. ISSN: 1477-4054. DOI: 10.1093/bib/bbp067 (pp. 60, 75, 104).

Ewald, Roland and Adelinde M. Uhrmacher (2014). "SESSL: A Domain-specific Language for Simulation Experiments". In: *ACM Transactions on Modeling and Computer*

Simulation (TOMACS) 24.2, 11:1–11:25. ISSN: 1049-3301. DOI: 10.1145/2567895 (pp. 81, 104).

Faeder, James R., Michael L. Blinov, and William S. Hlavacek (2009). "Rule-Based Modeling of Biochemical Systems with BioNetGen". In: *Methods in molecular biology (Clifton, N.J.)* Methods in Molecular Biology 500. Ed. by Ivan V. Maly, pp. 113–167. ISSN: 1064-3745. DOI: 10.1007/978-1-59745-525-1_5 (pp. 25, 56).

Faeder, James R., Michael L. Blinov, Byron Goldstein, and William S. Hlavacek (2005). "Rule-based modeling of biochemical networks". In: *Complexity* 10.4, pp. 22–41. DOI: 10.1002/cplx.20074 (pp. 30, 50).

Fange, D., O. G. Berg, P. Sjoberg, and J. Elf (2010). "Fange, David Berg, Otto G. Sjöberg, Paul Elf, Johan". In: *Proceedings of the National Academy of Sciences* 107.46, pp. 19820–19825. ISSN: 1091-6490. DOI: 10.1073/pnas.1006565107 (p. 19).

Feller, Willy (1940). "On the integro-differential equations of purely discontinuous Markoff processes". In: *Transactions of the American Mathematical Society* 48.3, pp. 488–515. ISSN: 0002-9947. DOI: 10.1090/s0002-9947-1940-0002697-3 (p. 11).

Fernández Slezak, Diego, Cecilia Suárez, Guillermo A. Cecchi, Guillermo Marshall, and Gustavo Stolovitzky (2010). "When the Optimal Is Not the Best: Parameter Estimation in Complex Biological Models". In: *PloS one* 5.10, e13283+. DOI: 10.1371/journal.pone.0013283 (p. 101).

Finke, Birgit, Frank Luethen, Karsten Schroeder, Petra D. Mueller, Claudia Bergemann, Marion Frant, Andreas Ohl, and Barbara J. Nebe (2007). "The effect of positively charged plasma polymerization on initial osteoblastic focal adhesion on titanium surfaces." In: *Biomaterials* 28.30, pp. 4521–4534. ISSN: 0142-9612. DOI: 10.1016/j.biomaterials.2007.06.028 (p. 94).

Fischer, Emil and Arthur Speier (1895). "Darstellung der Ester". In: *Berichte der deutschen chemischen Gesellschaft* 28.3, pp. 3252–3258. ISSN: 1099-0682. DOI: 10.1002/cber.189502803176 (p. 29).

Fisher, Jasmin, Nir Piterman, Hubbard, Michael J. Stern, and David Harel (2005). "Computational insights into Caenorhabditis elegans vulval development". In: *Proceedings of the National Academy of Sciences of the United States of America* 102.6, pp. 1951–1956. ISSN: 0027-8424. DOI: 10.1073/pnas.0409433102 (p. 27).

Flegg, Mark B., S. Jonathan Chapman, and Radek Erban (2011). "The two-regime method for optimizing stochastic reactionâĂŞdiffusion simulations". In: *Journal of The Royal Society Interface* 9.70, pp. 859–868. ISSN: 1742-5662. DOI: 10.1098/rsif.2011.0574 (p. 59).

Funahashi, A., Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano (2008). "CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks". In: *Proceedings of the IEEE* 96.8, pp. 1254–1265. ISSN: 1558-2256. DOI: 10.1109/jproc.2008.925458 (p. 22).

Funahashi, Akira, Mineo Morohashi, Yukiko Matsuoka, Akiya Jouraku, and Hiroaki Kitano (2007). "CellDesigner: A Graphical Biological Network Editor and Workbench Interfacing Simulator". In: *Introduction to Systems Biology*. Ed. by Sangdun Choi.

Totowa, NJ: Humana Press. Chap. 21, pp. 422–434. ISBN: 978-1-58829-706-8. DOI: 10.1007/978-1-59745-531-2_21 (p. 22).

Galpin, Vashti (2010). "Continuous approximation of PEPA models and Petri nets". In: *International Journal of Computer Aided Engineering and Technology* 2, pp. 324–339. URL: http://www.dcs.ed.ac.uk/pepa/ContinuousPEPAandPetriNets.pdf (p. 24).

Gao, Huajian, Jin Qian, and Bin Chen (2011). "Probing mechanical principles of focal contacts in cell-matrix adhesion with a coupled stochastic-elastic modelling framework." In: *Journal of the Royal Society, Interface / the Royal Society* 8.62, pp. 1217–1232. ISSN: 1742-5662. DOI: 10.1098/rsif.2011.0157 (p. 98).

García, Andrés J. (2005). "Get a grip: integrins in cellâĂŞbiomaterial interactions". In: *Biomaterials* 26.36, pp. 7525–7529. ISSN: 0142-9612. DOI: 10.1016/j.biomaterials.2005.05.029 (p. 94).

Gardiner, C. W., K. J. McNeil, D. F. Walls, and I. S. Matheson (1976). "Correlations in stochastic theories of chemical reactions". In: *Journal of Statistical Physics* 14.4, pp. 307–331. ISSN: 0022-4715. DOI: 10.1007/bf01030197 (pp. vi, 19).

Gardner, M. (1970). "Mathematical games: The fantastic combinations of John Conway's new solitaire game 'Life'". In: *Scientific American* 223.4, pp. 120–123 (pp. 18, 26).

Gargantini, Irene (1982). "An effective way to represent quadtrees". In: *Communications of the ACM* 25.12, pp. 905–910. ISSN: 0001-0782. DOI: 10.1145/358728.358741 (p. 80).

Geiger, Benjamin, Joachim P. Spatz, and Alexander D. Bershadsky (2009). "Environmental sensing through focal adhesions". In: *Nature reviews. Molecular cell biology* 10.1, pp. 21–33. ISSN: 1471-0080. DOI: 10.1038/nrm2593 (p. 94).

Gibson, Michael A. and Jehoshua Bruck (2000). "Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels". In: *Journal of Physical Chemistry A* 104.9, pp. 1876–1889. DOI: 10.1021/jp993732q (pp. v, 14, 46).

Gilbert, David, Monika Heiner, Fei Liu, and Nigel Saunders (2013). "Colouring Space - A Coloured Framework for Spatial Modelling in Systems Biology". In: *Application and Theory of Petri Nets and Concurrency*. Ed. by José-Manuel Colom and Jörg Desel. Vol. 7927. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 230–249. DOI: 10.1007/978-3-642-38697-8_13 (p. 26).

Gillespie, Daniel T. (1976). "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions". In: *Journal of Computational Physics* 22.4, pp. 403–434. ISSN: 00219991. DOI: 10.1016/0021-9991(76)90041-3 (pp. v, vii, 4, 11, 13).

– (1977). "Exact stochastic simulation of coupled chemical reactions". In: *The Journal of Physical Chemistry* 81.25, pp. 2340–2361. ISSN: 1541-5740. DOI: 10.1021/j100540a008 (pp. v, 4).

– (1992). "A rigorous derivation of the chemical master equation". In: *Physica A: Statistical Mechanics and its Applications* 188.1-3, pp. 404–425. ISSN: 03784371. DOI: 10.1016/0378-4371(92)90283-v (p. iv).

– (2001). "Approximate accelerated stochastic simulation of chemically reacting systems". In: *The Journal of Chemical Physics* 115.4, pp. 1716–1733. DOI: 10.1063/1.1378322 (pp. iv, 14, 15).

Goh, Rick Siow Mong and Ian L. Thng (2003). "MList: an efficient pending event set structure for discrete event simulation". In: *International Journal of Simulation* 4 (p. 78).

Goldstein, Rhys and Gabriel Wainer (2009). "DEVS-based design of spatial simulations of biological systems". In: *2009 Winter Simulation Conference - (WSC 2009)*. Ed. by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls. Austin, TX, USA: IEEE, pp. 743–754. ISBN: 978-1-4244-5770-0. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.154.7023. DOI: 10.1109/WSC.2009.5429688 (p. 27).

Gosling, James, Bill Joy, Guy L. Steele, Gilad Bracha, and Alex Buckley (2013). *The Java Language Specification, Java SE 7 Edition*. 1st. Addison-Wesley Professional. ISBN: 0133260224, 9780133260229. URL: http://docs.oracle.com/javase/specs/jls/se7/html/index.html (p. 75).

Gruenert, Gerd, Bashar Ibrahim, Thorsten Lenser, Maiko Lohel, Thomas Hinze, and Peter Dittrich (2010). "Rule-based spatial modeling with diffusing, geometrically constrained molecules". In: *BMC bioinformatics* 11.1, pp. 307+. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-307 (pp. 27, 36, 69, 114).

Guo, Kunkun, Julian Shillcock, and Reinhard Lipowsky (2009). "Self-assembly of actin monomers into long filaments: Brownian dynamics simulations". In: *The Journal of Chemical Physics* 131.1, pp. 015102+. ISSN: 0021-9606. DOI: 10.1063/1.3159003 (pp. 97, 98).

– (2010). "Treadmilling of actin filaments via Brownian dynamics simulations". In: *The Journal of Chemical Physics* 133.15, pp. 155105+. DOI: 10.1063/1.3497001 (pp. 97, 98).

Haack, Fiete, Stefan Leye, and Adelinde M. Uhrmacher (2010). "A flexible architecture for modeling and simulation of diffusional association". In: *From Biology to Concurrency and back*. Ed. by E. Merelli and P. Quaglia, pp. 70–84. arXiv: 1002.4064v1. DOI: 10.4204/eptcs.19.5 (p. 24).

Haack, Fiete, Kevin Burrage, Ronald Redmer, and Adelinde M. Uhrmacher (2013). "Studying the Role of Lipid Rafts on Protein Receptor Bindings with Cellular Automata". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 10.3, pp. 760–770. ISSN: 1545-5963. DOI: 10.1109/TCBB.2013.40 (pp. 19, 26, 85, 101).

Haack, Fiete, Heiko Lemcke, Roland Ewald, Tareck Rharass, and Adelinde M. Uhrmacher (2015). "Spatio-temporal Model of Endogenous ROS and Raft-Dependent WNT/Beta-Catenin Signaling Driving Cell Fate Commitment in Human Neural Progenitor Cells". In: *PLOS Computational Biology* 11.3. Ed. by Jeffrey J. Saucerman, e1004106+. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1004106 (p. 85).

Hale, D. Hunter and G. Michael Youngblood (2014). "Collision Detection with Navigation Meshes". In: pp. 130–145. DOI: 10.1002/9781118796443.ch5 (p. 80).

Harang, Richard E., Guillaume Bonnet, and Linda R. Petzold (2012). "WAVOS: a MATLAB toolkit for wavelet analysis and visualization of oscillatory systems." In: *BMC research notes* 5.1, pp. 163+. ISSN: 1756-0500. DOI: 10.1186/1756-0500-5-163 (p. 85).

Harel, David (1987). "Statecharts: A Visual Formalism for Complex Systems". In: *Science of Computer Programming* 8.3, pp. 231–274. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.4799 (p. 27).

Harris, Leonard A., Justin S. Hogg, and James R. Faeder (2009). "Compartmental rule-based modeling of biochemical systems". In: *Proceedings of the 2009 Winter Simulation Conference (WSC)*. Ed. by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls. Austin, TX, USA: IEEE Computer Science, pp. 908–919. ISBN: 978-1-4244-5770-0. DOI: 10.1109/wsc.2009.5429719 (pp. 25, 28, 33).

Haugh, Jason M. (2002). "A unified model for signal transduction reactions in cellular membranes." In: *Biophysical journal* 82.2, pp. 591–604. ISSN: 0006-3495. DOI: 10.1016/s0006-3495(02)75424-6 (p. 18).

Heiner, Monika, David Gilbert, and Robin Donaldson (2008). "Petri Nets for Systems and Synthetic Biology". In: *Formal Methods for Computational Systems Biology*. Ed. by Marco Bernardo, Pierpaolo Degano, and Gianluigi Zavattaro. Vol. 5016. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 215–264. DOI: 10.1007/978-3-540-68894-5_7 (p. 26).

Heyes, D. M., J. Baxter, U. Tüzün, and R. S. Qin (2004). "DiscreteâĂŞelement method simulations: from micro to macro scales". In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 362.1822, pp. 1853–1865. DOI: 10.1098/rsta.2004.1420 (p. 17).

Himmelspach, Jan (2007). *Konzeption, Realisierung und Verwendung eines allgemeinen Modellierungs-, Simulations- und Experimentiersystems - Entwicklung und Evaluation effizienter Simulationsalgorithmen*. 1st. Vol. 4. Göttingen, Germany: Sierke Verlag, p. 290. URL: http://www.sierke-verlag.de/shop/index.php/konzeption-realisierung-und-verwendung-eines-allgemeinen-modellierungs-simulations-und-experimentiersystems.html (p. 75).

Himmelspach, Jan and Adelinde M. Uhrmacher (2007). "Plug'N Simulate". In: *Proceedings of the 40th Annual Simulation Symposium*. ANSS '07. Washington, DC, USA: IEEE Computer Society, pp. 137–143. ISBN: 0-7695-2814-7. DOI: 10.1109/anss.2007.34 (pp. 60, 75).

Hirata, Hiromi, Shigeki Yoshiura, Toshiyuki Ohtsuka, Yasumasa Bessho, Takahiro Harada, Kenichi Yoshikawa, and Ryoichiro Kageyama (2002). "Oscillatory Expression of the bHLH Factor Hes1 Regulated by a Negative Feedback Loop". In: *Science (New York, N.Y.)* 298.5594, pp. 840–843. ISSN: 1095-9203. DOI: 10.1126/science.1074560 (p. 84).

Hlavacek, William S., James R. Faeder, Michael L. Blinov, Richard G. Posner, Michael Hucka, and Walter Fontana (2006). "Rules for Modeling Signal-Transduction Systems".

In: *Science's STKE* 2006.344, re6+. ISSN: 1525-8882. DOI: 10.1126/stke.3442006re6 (pp. 4, 25).

Hoffmann, Max and Ulrich Schwarz (2013). "A kinetic model for RNA-interference of focal adhesions". In: *BMC Systems Biology* 7.1, pp. 2+. ISSN: 1752-0509. DOI: 10.1186/1752-0509-7-2 (p. 98).

Hogg, Justin S., Leonard A. Harris, Lori J. Stover, Niketh S. Nair, and James R. Faeder (2014). "Exact Hybrid Particle/Population Simulation of Rule-Based Models of Biochemical Systems". In: *PLoS Computational Biology* 10.4. Ed. by Jorg Stelling, e1003544+. ISSN: 1553-7358. arXiv: 1301.6854. DOI: 10.1371/journal.pcbi.1003544 (p. 30).

Holcombe, Mike and Alex Bell (1998). "Computational models of immunological pathways". In: *IPCAT '97: Proceedings of the second international workshop on Information processing in cell and tissues*. Sheffield, United Kingdom: Plenum Press, pp. 213–226. ISBN: 0-306-45839-X. URL: http://portal.acm.org/citation.cfm?id=302049 (p. 27).

Hoops, Stefan, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer (2006). "COPASIâĂŤa COmplex PAthway SImulator". In: *Bioinformatics (Oxford, England)* 22.24, pp. 3067–3074. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btl485 (p. 22).

Huber, Florian, Josef Käs, and Björn Stuhrmann (2008). "Growing Actin Networks Form Lamellipodium and Lamellum by Self-Assembly". In: *Biophysical Journal* 95.12, pp. 5508–5523. ISSN: 00063495. DOI: 10.1529/biophysj.108.134817 (pp. 97–99).

Hucka, M., A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, and Others (2003). "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models". In: *Bioinformatics (Oxford, England)* 19.4, pp. 524–531. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btg015 (p. vii).

Iyengar, Krishna A., Leonard A. Harris, and Paulette Clancy (2010). "Accurate implementation of leaping in space: The spatial partitioned-leaping algorithm". In: *The Journal of Chemical Physics* 132.9, pp. 094101+. ISSN: 0021-9606. DOI: 10.1063/1.3310808 (p. 89).

Jensen, Kurt (1981). "Coloured petri nets and the invariant-method". In: *Theoretical Computer Science* 14.3, pp. 317–336. ISSN: 03043975. DOI: 10.1016/0304-3975(81)90049-9 (p. 26).

Jeschke, Matthias, Roland Ewald, and Adelinde M. Uhrmacher (2011). "Exploring the performance of spatial stochastic simulation algorithms". In: *Journal of Computational Physics* 230.7, pp. 2562–2574. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2010.12.030 (pp. 76, 89, 117).

Jeschke, Matthias and Adelinde M. Uhrmacher (2008). "Multi-resolution spatial simulation for molecular crowding". In: *Proceedings of the 2008 Winter Simulation Conference*. Ed. by S. J. Mason, R. R. Hill, L. Moench, and O. Rose. Miami, FL, USA: IEEE, pp. 1384–1392. ISBN: 978-1-4244-2707-9. DOI: 10.1109/WSC.2008.4736214 (pp. 3, 75, 118).

John, Mathias, Roland Ewald, and Adelinde M. Uhrmacher (2008). "A Spatial Extension to the π Calculus". In: *Electronic Notes in Theoretical Computer Science* 194.3, pp. 133–148. ISSN: 15710661. DOI: 10.1016/j.entcs.2007.12.010 (pp. 24, 28).

John, Mathias, Cédric Lhoussaine, and Joachim Niehren (2009). "Dynamic Compartments in the Imperative π-Calculus". In: *Computational Methods in Systems Biology*. Ed. by Pierpaolo Degano and Roberto Gorrieri. Vol. 5688. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. 16, pp. 235–250. ISBN: 978-3-642-03844-0. DOI: 10.1007/978-3-642-03845-7_16 (p. 24).

John, Mathias, Cédric Lhoussaine, Joachim Niehren, and Adelinde Uhrmacher (2010). "The Attributed Pi-Calculus with Priorities". In: *Transactions on Computational Systems Biology XII*. Ed. by Corrado Priami, Rainer Breitling, David Gilbert, Monika Heiner, and Adelinde Uhrmacher. Vol. 5945. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin / Heidelberg. Chap. 2, pp. 13–76. ISBN: 978-3-642-11711-4. DOI: 10.1007/978-3-642-11712-1_2 (p. 24).

Kam, Naaman, Irun R. Cohen, and David Harel (2002). "Modeling biological reactivity: statecharts vs. Boolean logic". In: *AVI '02: Proceedings of the Working Conference on Advanced Visual Interfaces*. Trento, Italy: ACM, pp. 345–353. ISBN: 1-58113-537-8. DOI: 10.1145/1556262.1556318 (p. 27).

Karr, Jonathan R., Jayodita C. Sanghvi, Derek N. Macklin, Miriam V. Gutschow, Jared M. Jacobs, Benjamin Bolival, Nacyra Assad-Garcia, John I. Glass, and Markus W. Covert (2012). "A Whole-Cell Computational Model Predicts Phenotype from Genotype". In: *Cell* 150.2, pp. 389–401. ISSN: 1097-4172. DOI: 10.1016/j.cell.2012.05.044 (p. 2).

Kendall, David G. (1950). "An Artificial Realization of a Simple "Birth-and-Death" Process". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 12.1, pp. 116–119. URL: http://www.jstor.org/stable/2983837 (p. 11).

Kholodenko, Boris N. (2006). "Cell-signalling dynamics in time and space." In: *Nature reviews. Molecular cell biology* 7.3, pp. 165–176. ISSN: 1471-0072. DOI: 10.1038/nrm1838 (p. 18).

Kholodenko, Boris N., John F. Hancock, and Walter Kolch (2010). "Signalling ballet in space and time". In: *Nature Reviews Molecular Cell Biology* 11.6, pp. 414–426. ISSN: 1471-0072. DOI: 10.1038/nrm2901 (p. 5).

Kier, Lemont B., C. K. Cheng, Bernard Testa, and Pierre-Alain Carrupt (1996). "A cellular automata model of enzyme kinetics". In: *Journal of Molecular Graphics* 14.4, pp. 227–231. ISSN: 02637855. DOI: 10.1016/s0263-7855(96)00073-2 (pp. 18, 26).

Kitano, Hiroaki (2002). "Computational systems biology." In: *Nature* 420.6912, pp. 206–210. ISSN: 0028-0836. DOI: 10.1038/nature01254 (p. 1).

Klann, Michael, Arnab Ganguly, and Heinz Koeppl (2012). "Hybrid spatial Gillespie and particle tracking simulation". In: *Bioinformatics (Oxford, England)* 28.18, pp. i549–i555. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/bts384 (p. 116).

Klann, Michael and Heinz Koeppl (2012). "Spatial Simulations in Systems Biology: From Molecules to Cells". In: *International Journal of Molecular Sciences* 13.12, pp. 7798–

7827. ISSN: 1422-0067. URL: http://www.mdpi.com/1422-0067/13/6/7798. DOI: 10.3390/ijms13067798 (p. 21).

Klann, Michael, Loïc Paulevé, Tatjana Petrov, and Heinz Koeppl (2013). "Coarse-Grained Brownian Dynamics Simulation of Rule-Based Models". In: *Computational Methods in Systems Biology*. Ed. by Ashutosh Gupta and ThomasA Henzinger. Vol. 8130. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 64–77. DOI: 10.1007/978-3-642-40708-6_6 (p. 38).

Kolmogoroff, A. (1931). "Über die analytischen Methoden in der Wahrscheinlichkeitsrechnung". In: *Mathematische Annalen* 104.1, pp. 415–458. DOI: 10.1007/bf01457949 (p. 11).

Kossow, Christina, Stefan Rybacki, Thomas Millat, Katja Rateitschak, Robert Jaster, Adelinde M. Uhrmacher, and Olaf Wolkenhauer (2015). "An explicit numerical scheme to efficiently simulate molecular diffusion in environments with dynamically changing barriers". In: *Mathematical and Computer Modelling of Dynamical Systems* 0, pp. 1–25. ISSN: 1744-5051. DOI: 10.1080/13873954.2015.1033823 (pp. 6, 76, 117).

Kugler, Hillel (2013). "Biocharts: Unifying Biological Hypotheses with Models and Experiments". In: *eScience (eScience), 2013 IEEE 9th International Conference on*. IEEE, pp. 317–325. DOI: 10.1109/escience.2013.41 (p. 27).

Kugler, Hillel, Antti Larjo, and David Harel (2010). "Biocharts: a visual formalism for complex biological systems". In: *Journal of The Royal Society Interface* 7.48, pp. 1015–1024. ISSN: 1742-5662. DOI: 10.1098/rsif.2009.0457 (p. 27).

Lai, Xin, Svetoslav Nikolov, Olaf Wolkenhauer, and Julio Vera (2009). "A multi-level model accounting for the effects of JAK2-STAT5 signal modulation in erythropoiesis". In: *Computational Biology and Chemistry* 33.4, pp. 312–324. ISSN: 14769271. DOI: 10.1016/j.compbiolchem.2009.07.003 (p. 3).

Li, Hong and Linda Petzold (2006). "Logarithmic Direct Method for discrete stochastic simulation of chemically reacting systems". URL: http://www.engineering.ucsb.edu/~cse/Files/ldm0513.pdf (p. 15).

Luboschik, M., M. Röhlig, A. T. Bittig, N. Andrienko, H. Schumann, and C. Tominski (2015). "Feature-Driven Visual Analytics of Chaotic Parameter-Dependent Movement". In: *Computer Graphics Forum* 34.3, pp. 421–430. DOI: 10.1111/cgf.12654 (p. 8).

Luboschik, Martin, Christian Tominski, Arne Bittig, Adelinde M. Uhrmacher, and Heidrun Schumann (2012). "Towards Interactive Visual Analysis of Microscopic-Level Simulation Data". In: *Proceedings of SIGRAD 2012 – Interactive Visual Analysis of Data*. Växjö, Sweden, pp. 91–94. ISBN: 978-91-7519-723-4. URL: http://www.ep.liu.se/ecp/081/013/ecp12081013.pdf (p. 8).

Lüthen, Frank, Regina Lange, Petra Becker, Joachim Rychly, Ulrich Beck, and J. G. Barbara Nebe (2005). "The influence of surface roughness of titanium on beta1- and beta3-integrin adhesion and the organization of fibronectin in human osteoblastic cells." In: *Biomaterials* 26.15, pp. 2423–2440. DOI: 10.1016/j.biomaterials.2004.07.054 (p. 94).

Marsaglia, G. and T. A. Bray (1964). "A Convenient Method for Generating Normal Variables". In: *SIAM Review* 6.3, pp. 260–264. URL: http://www.jstor.org/stable/2027592 (p. 78).

Materi, Wayne and David S. Wishart (2007). "Computational systems biology in drug discovery and development: methods and applications." In: *Drug discovery today* 12.7-8, pp. 295–303. ISSN: 1359-6446. DOI: 10.1016/j.drudis.2007.02.013 (p. 26).

MATLAB version 7.10.0 (R2010a) (2010). Natick, Massachusetts (p. 104).

Matschegewski, Claudia, Susanne Staehlke, Ronny Loeffler, Regina Lange, Feng Chai, Dieter P. Kern, Ulrich Beck, and J. Barbara Nebe (2010). "Cell architecture–cell function dependencies on titanium arrays with regular geometry". In: *Biomaterials* 31.22, pp. 5729–5740. ISSN: 01429612. DOI: 10.1016/j.biomaterials.2010.03.073 (p. 94).

Matschegewski, Claudia, Susanne Staehlke, Harald Birkholz, Regina Lange, Ulrich Beck, Konrad Engel, and J. Barbara Nebe (2012). "Automatic Actin Filament Quantification of Osteoblasts and Their Morphometric Analysis on Microtextured Silicon-Titanium Arrays". In: *Materials* 5.7, pp. 1176–1195. DOI: 10.3390/ma5071176 (pp. 94, 95, 99, 110).

Matsumoto, Makoto and Takuji Nishimura (1998). "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator". In: *ACM Transactions on Modeling and Computer Simulation* 8.1, pp. 3–30. ISSN: 1049-3301. DOI: 10.1145/272991.272995 (p. 78).

Maus, Carsten (2008). "Component-Based Modelling of RNA Structure Folding". In: *Computational Methods in Systems Biology*. Ed. by Monika Heiner and Adelinde M. Uhrmacher. Vol. 5307. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. 8, pp. 44–62. ISBN: 978-3-540-88561-0. DOI: 10.1007/978-3-540-88562-7_8 (p. 17).

– (2013). *Toward Accessible Multilevel Modeling in Systems Biology: A Rule-based Language Concept*. Logos Verlag. ISBN: 3832535160. URL: http://rosdok.uni-rostock.de/resolve?urn=urn:nbn:de:gbv:28-diss2013-0152-5&pdf (pp. 5, 25, 27, 29).

Maus, Carsten, Stefan Rybacki, and Adelinde M. Uhrmacher (2011). "Rule-based multi-level modeling of cell biological systems". In: *BMC Systems Biology* 5.1, pp. 166+. ISSN: 1752-0509. DOI: 10.1186/1752-0509-5-166 (pp. 5, 25, 28, 30, 33, 37).

Maus, Carsten, Mathias John, Mathias Röhl, and Adelinde Uhrmacher (2008). "Hierarchical Modeling for Computational Biology". In: *Formal Methods for Computational Systems Biology*. Ed. by Marco Bernardo, Pierpaolo Degano, and Gianluigi Zavattaro. Vol. 5016. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. 4, pp. 81–124. ISBN: 978-3-540-68892-1. DOI: 10.1007/978-3-540-68894-5_4 (p. 27).

McCollum, James M., Gregory D. Peterson, Chris D. Cox, Michael L. Simpson, and Nagiza F. Samatova (2006). "The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior". In: *Computational Biology and Chemistry* 30.1, pp. 39–49. ISSN: 14769271. DOI: 10.1016/j.compbiolchem.2005.10.007 (p. 15).

McQuarrie, Donald A. (1967). "Stochastic Approach to Chemical Kinetics". In: *Journal of Applied Probability* 4.3, pp. 413–478. URL: http://www.jstor.org/stable/3212214 (p. 4).

Mendel, G. (1866). "Versuche über Pflanzen-Hybriden". In: *Verhandlungen des naturforschenden Vereines, Abhandlungen, Brünn* 4. "Experiments in Plant Hybridization" as translated by William Bateson around 1900, pp. 3–47. URL: http://www.mendelweb.org/Mendel.html (p. 1).

Milner, Robin (1999). *Communicating and Mobile Systems: The Pi Calculus*. 1st. Cambridge University Press. ISBN: 0521658691. URL: http://www.worldcat.org/isbn/0521658691 (p. 23).

Möhring, Michael (1996). "Social Science Multilevel Simulation with MIMOSE". In: *Social Science Microsimulation*. Ed. by K. G. Troizsch and Others. Springer-Verlag Berlin, pp. 282–306. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.533 (p. 2).

Moreira, Joana and Andreas Deutsch (2002). "Cellular Automaton Models Of Tumor Development: A Critical Review". In: *Advances in Complex Systems (ACS)* 5.02, pp. 247–267. URL: http://ideas.repec.org/a/wsi/acsxxx/v05y2002i02p247-267.html (pp. 18, 26).

Nähring, S., T. Helms, S. Rybacki, and Uhrmacher AM (2014). "Functions on Solutions – Language Extensions for ML-Rules". In: *Computational Methods in Systems Biology*. Ed. by Pedro Mendes, Joseph O. Dada, and Kieran Smallbone. Poster. URL: http://www.comp-sys-bio.org/CMSB14/abstracts/cmsb2014_submission_44.pdf (p. 54).

Nähring, Sebastian (2014). "Functions on Solutions - Language Extension of ML-Rules". MA thesis. University of Rostock (p. 54).

Nebe, J. B., B. Finke, A. Körtge, H. Rebl, and S. Stählke (2014). "Geometrical micropillars combined with chemical surface modifications – Independency of actin filament spatial distribution in primary osteoblasts". In: *Materials Science Forum* 783-786, pp. 1320–1325. DOI: 10.4028/www.scientific.net/MSF.783-786.1320 (p. 94).

Nebe, J. Barbara, Frank Lüthen, Regina Lange, and Ulrich Beck (2007). "Interface Interactions of Osteoblasts with Structured Titanium and the Correlation between Physicochemical Characteristics and Cell Biological Parameters". In: *Macromolecular Bioscience* 7.5, pp. 567–578. ISSN: 1616-5195. DOI: 10.1002/mabi.200600293 (p. 94).

Nicolau, Dan V., Kevin Burrage, Robert G. Parton, and John F. Hancock (2006). "Identifying optimal lipid raft characteristics required to promote nanoscale protein-protein interactions on the plasma membrane." In: *Molecular and cellular biology* 26.1, pp. 313–323. ISSN: 0270-7306. DOI: 10.1128/mcb.26.1.313-323.2006 (p. 100).

Nicolis, Gregoire and I. Prigogine (1977). *Self-Organization in Nonequilibrium Systems: From Dissipative Structures to Order through Fluctuations*. 1st ed. Wiley. ISBN: 0471024015. URL: http://www.worldcat.org/isbn/0471024015 (pp. vi, 19).

Noble, Denis (2012). "A theory of biological relativity: no privileged level of causation". In: *Interface focus* 2.1, pp. 55–64. ISSN: 2042-8901. DOI: 10.1098/rsfs.2011.0067 (p. 2).

Northrup, Scott H., Stuart A. Allison, and J. Andrew McCammon (1984). "Brownian dynamics simulation of diffusion-influenced bimolecular reactions". In: *The Journal of Chemical Physics* 80.4, pp. 1517–1524. DOI: 10.1063/1.446900 (p. 17).

Noyes, Richard M. (1956). "Models Relating Molecular Reactivity and Diffusion in Liquids". In: *Journal of the American Chemical Society* 78.21, pp. 5486–5490. ISSN: 1520-5126. DOI: 10.1021/ja01602a007 (p. 21).

Oury, Nicolas and Gordon Plotkin (2013). "Multi-level modelling via stochastic multi-level multiset rewriting". In: *Mathematical Structures in Computer Science* 23, pp. 471–503. ISSN: 1469-8072. DOI: 10.1017/s0960129512000199 (p. 28).

Pahle, Jürgen (2009). "Biochemical simulations: stochastic, approximate stochastic and hybrid approaches". In: *Briefings in Bioinformatics* 10.1, pp. 53–64. ISSN: 1477-4054. DOI: 10.1093/bib/bbn050 (pp. 4, 12).

Panneton, François, Pierre L'Ecuyer, and Makoto Matsumoto (2006). "Improved long-period generators based on linear recurrences modulo 2". In: *ACM Transactions on Mathematical Software* 32.1, pp. 1–16. ISSN: 0098-3500. DOI: 10.1145/1132973.1132974 (p. 78).

Park, Junseong, Jungsul Lee, and Chulhee Choi (2011). "Mitochondrial Network Determines Intracellular ROS Dynamics and Sensitivity to Oxidative Stress through Switching Inter-Mitochondrial Messengers". In: *PloS one* 6.8, e23211+. DOI: 10.1371/journal.pone.0023211 (p. 93).

Park, S. K. and K. W. Miller (1988). "Random Number Generators: Good Ones Are Hard to Find". In: *Commun. ACM* 31.10, pp. 1192–1201. ISSN: 0001-0782. DOI: 10.1145/63039.63042 (p. 78).

Parr, Terence (2007). *The Definitive ANTLR Reference: Building Domain-Specific Languages (Pragmatic Programmers)*. 1st ed. Pragmatic Bookshelf. ISBN: 0978739256. URL: http://www.worldcat.org/isbn/0978739256 (p. 75).

Patel, Pinkesh K., Orian Shirihai, and Kerwyn C. Huang (2013). "Optimal Dynamics for Quality Control in Spatially Distributed Mitochondrial Networks". In: *PLoS Computational Biology* 9.7. Ed. by Mark S. Alber, e1003108+. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1003108 (pp. 83, 90).

Pedersen, Michael and Gordon Plotkin (2008). "A Language for Biochemical Systems". In: *Computational Methods in Systems Biology*, pp. 63–82. DOI: 10.1007/978-3-540-88562-7_9 (p. 25).

– (2010). "A Language for Biochemical Systems: Design and Formal Specification". In: *Transactions on Computational Systems Biology XII*. Ed. by Corrado Priami, Rainer Breitling, David Gilbert, Monika Heiner, and Adelinde M. Uhrmacher. Vol. 5945. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. 3, pp. 77–145. ISBN: 978-3-642-11711-4. DOI: 10.1007/978-3-642-11712-1_3 (p. 25).

Peng, Xin-jun and Yi-fei Wang (2007). "L-leap: accelerating the stochastic simulation of chemically reacting systems". In: *Applied Mathematics and Mechanics* 28.10, pp. 1361–1371. DOI: 10.1007/s10483-007-1009-y (p. 15).

Petri, C. Adam and W. Reisig (2008). "Petri net". In: *Scholarpedia* 3.4. revision #91646, p. 6477. URL: http://www.scholarpedia.org/w/index.php?title=Petri_net&oldid=91646. DOI: doi:10.4249/scholarpedia.6477 (p. 26).

Pfaendtner, Jim, Enrique M. de la Cruz, and Gregory A. Voth (2010). "Actin filament remodeling by actin depolymerization factor/cofilin". In: *Proceedings of the National Academy of Sciences of the United States of America* 107.16, pp. 7299–7304. ISSN: 1091-6490. DOI: 10.1073/pnas.0911675107 (p. 96).

Plimpton, Steve (1995). "Fast Parallel Algorithms for Short-Range Molecular Dynamics". In: *Journal of Computational Physics* 117.1, pp. 1–19. ISSN: 0021-9991. DOI: 10.1006/jcph.1995.1039 (p. 36).

Pollard, Thomas D. and John A. Cooper (2009). "Actin, a Central Player in Cell Shape and Movement". In: *Science (New York, N.Y.)* 326.5957, pp. 1208–1212. ISSN: 1095-9203. DOI: 10.1126/science.1175862 (pp. 94, 96, 118).

Priami, C. (1995). "Stochastic pi-Calculus". In: *The Computer Journal* 38.7, pp. 578–589. DOI: 10.1093/comjnl/38.7.578 (p. 23).

Priami, Corrado and Paola Quaglia (2005). "Beta Binders for Biological Interactions". In: *Computational Methods in Systems Biology*. Springer Berlin Heidelberg, pp. 20–33. ISBN: 978-3-540-25375-4. URL: http://www.springerlink.com/content/py3m20bgr7g9je7f (p. 24).

Ramaswamy, Rajesh, Nélido González-Segredo, and Ivo F. Sbalzarini (2009). "A new class of highly efficient exact stochastic simulation algorithms for chemical reaction networks". In: *The Journal of Chemical Physics* 130.24, pp. 244104+. ISSN: 0021-9606. DOI: 10.1063/1.3154624 (p. 15).

Ramaswamy, Rajesh and Ivo F. Sbalzarini (2010). "A partial-propensity variant of the composition-rejection stochastic simulation algorithm for chemical reaction networks". In: *The Journal of Chemical Physics* 132.4, pp. 044102+. ISSN: 0021-9606. DOI: 10.1063/1.3297948 (p. 15).

Rathinam, Muruhan, Linda R. Petzold, Yang Cao, and Daniel T. Gillespie (2003). "Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method". In: *The Journal of Chemical Physics* 119.24, pp. 12784–12794. ISSN: 0021-9606. DOI: 10.1063/1.1627296 (p. 15).

Recalde, Laura, Serge Haddad, and Manuel Silva (2007). "Continuous Petri Nets: Expressive Power and Decidability Issues". In: *Automated Technology for Verification and Analysis*. Ed. by Kedar S. Namjoshi, Tomohiro Yoneda, Teruo Higashino, and Yoshio Okamura. Vol. 4762. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. 26, pp. 362–377. ISBN: 978-3-540-75595-1. DOI: 10.1007/978-3-540-75596-8_26 (p. 23).

Regev, Aviv and Ehud Shapiro (2004). "The π-calculus as an Abstraction for Biomolecular Systems". In: ed. by G. Rozenberg, T. Bäck, A. E. Eiben, J. N. Kok, H. P. Spaink,

Gabriel Ciobanu, and Grzegorz Rozenberg. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. 11, pp. 219–266. ISBN: 978-3-642-62269-4. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.4739. DOI: 10.1007/978-3-642-18734-6_11 (p. 23).

Regev, Aviv, Ekaterina M. Panina, William Silverman, Luca Cardelli, and Ehud Shapiro (2004). "BioAmbients: an abstraction for biological compartments". In: *Theoretical Computer Science* 325.1, pp. 141–167. ISSN: 03043975. DOI: 10.1016/j.tcs.2004.03.061 (p. 24).

Reinhardt, F., J. Schultz, R. Waterstradt, and S. Baltrusch (2013). "The mitochondrial fission proteins Fis1 and Drp1 are important for glucose-induced insulin secretion in glucose-responsive INS1 832/13 cells". In: *Experimental and Clinical Endocrinology & Diabetes* 121.10, P17+. ISSN: 1439-3646. DOI: 10.1055/s-0033-1359452 (p. 92).

Roland, Jeremy, Julien Berro, Alphée Michelot, Laurent Blanchoin, and Jean-Louis L. Martiel (2008). "Stochastic severing of actin filaments by actin depolymerizing factor/-cofilin controls the emergence of a steady dynamical regime." In: *Biophysical journal* 94.6, pp. 2082–2094. ISSN: 1542-0086. DOI: 10.1529/biophysj.107.121988 (pp. 96–98, 102).

Rönngren, Robert and Rassul Ayani (1997). "A Comparative Study of Parallel and Sequential Priority Queue Algorithms". In: *ACM Transactions on Modeling and Computer Simulation* 7.2, pp. 157–209. ISSN: 1049-3301. DOI: 10.1145/249204.249205 (p. 75).

Sackmann, Andrea, Monika Heiner, and Ina Koch (2006). "Application of Petri net based analysis techniques to signal transduction pathways". In: *BMC bioinformatics* 7.1, pp. 482+. ISSN: 1471-2105. DOI: 10.1186/1471-2105-7-482 (p. 26).

Sang, Liyun, Hilary A. Coller, and James M. Roberts (2008). "Control of the Reversibility of Cellular Quiescence by the Transcriptional Repressor HES1". In: *Science (New York, N.Y.)* 321.5892, pp. 1095–1100. ISSN: 1095-9203. DOI: 10.1126/science.1155998 (p. 84).

Sarjoughian, Hessam S. (2006). "Model Composability". In: *2006 Winter Simulation Conference*. Ed. by Luiz F. Perrone, Frederick P. Wieland, Jason Liu, Barry G. Lawson, David M. Nicol, and Richard M. Fujimoto. Monterey, CA, USA: IEEE, pp. 149–158. ISBN: 1-4244-0501-7. URL: http://dl.acm.org/citation.cfm?id=1218112.1218144. DOI: 10.1109/WSC.2006.323047 (p. 4).

Sarjoughian, Hessam S. and Bernhard P. Zeigler (2000). "DEVS and HLA: Complementary Paradigms for Modeling and Simulation?" In: *Trans. Soc. Comput. Simul. Int.* 17.4, pp. 187–197. ISSN: 0740-6797. URL: http://portal.acm.org/citation.cfm?id=374863 (p. 4).

Schmitt, H., S. Lenzen, and S. Baltrusch (2011). "Glucokinase mediates coupling of glycolysis to mitochondrial metabolism but not to beta cell damage at high glucose exposure levels." In: *Diabetologia* 54.7, pp. 1744–1755. ISSN: 1432-0428. URL: http://view.ncbi.nlm.nih.gov/pubmed/21484215 (p. 90).

Schwartz, Z., C. H. Lohmann, J. Oefinger, L. F. Bonewald, D. D. Dean, and B. D. Boyan (1999). "Implant surface characteristics modulate differentiation behavior of cells in the osteoblastic lineage." In: *Advances in dental research* 13, pp. 38–48. ISSN: 0895-9374. URL: http://view.ncbi.nlm.nih.gov/pubmed/11276745 (p. 94).

Schwarz, U. and S. A. Safran (2013). "Physics of adherent cells". In: *Reviews of modern physics* 85, pp. 1328–1381 (p. 94).

Selhuber-Unkel, C., T. Erdmann, M. López-Garc'ıa, H. Kessler, U. S. Schwarz, and J. P. Spatz (2010). "Cell adhesion strength is controlled by intermolecular spacing of adhesion receptors." In: *Biophysical journal* 98.4, pp. 543–551. ISSN: 1542-0086. DOI: 10.1016/j.bpj.2009.11.001 (p. 94).

Shimizu, Thomas S., Sergej V. Aksenov, and Dennis Bray (2003). "A Spatially Extended Stochastic Model of the Bacterial Chemotaxis Signalling Pathway". In: *Journal of Molecular Biology* 329.2, pp. 291–309. ISSN: 00222836. DOI: 10.1016/s0022-2836(03)00437-6 (p. 19).

Slepchenko, Boris M., James C. Schaff, Ian Macara, and Leslie M. Loew (2003). "Quantitative cell biology with the Virtual Cell." In: *Trends in cell biology* 13.11, pp. 570–576. ISSN: 0962-8924. URL: http://view.ncbi.nlm.nih.gov/pubmed/14573350 (p. 18).

Sneddon, Michael W. and Thierry Emonet (2012). "Modeling cellular signaling: taking space into the computation". In: *Nature Methods* 9.3, pp. 239–242. DOI: 10.1038/nmeth.1900 (p. 98).

Sneddon, Michael W., James R. Faeder, and Thierry Emonet (2011). "Efficient modeling, simulation and coarse-graining of biological complexity with NFsim". In: *Nature Methods* 8.2, pp. 177–183. ISSN: 1548-7105. DOI: 10.1038/nmeth.1546 (pp. 50, 97, 98).

Sorokina, Oksana, Anatoly Sorokin, J. Douglas Armstrong, and Vincent Danos (2013). "A simulator for spatially extended kappa models." In: *Bioinformatics (Oxford, England)* 29.23, pp. 3105–3106. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btt523 (pp. 28, 35).

Spatz, Joachim P. and Benjamin Geiger (2007). "Molecular engineering of cellular environments: cell adhesion to nano-digital surfaces." In: *Methods in cell biology* 83, pp. 89–111. ISSN: 0091-679X. DOI: 10.1016/s0091-679x(07)83005-6 (p. 94).

Stähnke, S., C. Matschegewski, R. Löffler, R. Lange, U. Beck, D. Kern, and J. B. Nebe (2010). "Time dependent analysis of intracellular signaling molecules in osteoblasts on microstructured titanium surfaces". In: *BIOmaterialien* 11, p. 179 (p. 94).

Stricker, Jonathan, Tobias Falzone, and Margaret L. Gardel (2010). "Mechanics of the F-actin cytoskeleton." In: *Journal of biomechanics* 43.1, pp. 9–14. ISSN: 1873-2380. DOI: 10.1016/j.jbiomech.2009.09.003 (pp. 94, 96).

Stundzia, Audrius B. and Charles J. Lumsden (1996). "Stochastic Simulation of Coupled Reaction-Diffusion Processes". In: *Journal of Computational Physics* 127.1, pp. 196–207. ISSN: 0021-9991. DOI: 10.1006/jcph.1996.0168 (pp. 5, 19).

Sturrock, Marc, Andreas Hellander, Anastasios Matzavinos, and Mark A. J. Chaplain (2013). "Spatial stochastic modelling of the Hes1 gene regulatory network: intrinsic

noise can explain heterogeneity in embryonic stem cell differentiation". In: *Journal of The Royal Society Interface* 10.80, pp. 20120988+. ISSN: 1742-5662. DOI: 10.1098/rsif.2012.0988 (pp. x, 41, 84, 85).

Takahashi, K., K. Kaizu, B. Hu, and M. Tomita (2004). "A multi-algorithm, multi-timescale method for cell simulation". In: *Bioinformatics (Oxford, England)* 20.4, pp. 538–546. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btg442 (p. 3).

Takahashi, Kouichi, Satya N. V. Arjunan, and Masaru Tomita (2005). "Space in systems biology of signaling pathways - towards intracellular molecular crowding in silico". In: *FEBS Letters* 579.8, pp. 1783–1788. ISSN: 0014-5793. DOI: 10.1016/j.febslet.2005.01.072 (p. 18).

Tania, Nessy, Erin Prosk, John Condeelis, and Leah Edelstein-Keshet (2011). "A temporal model of cofilin regulation and the early peak of actin barbed ends in invasive tumor cells." In: *Biophysical journal* 100.8, pp. 1883–1892. ISSN: 1542-0086. DOI: 10.1016/j.bpj.2011.02.036 (p. 102).

The Java Programming language. https://www.java.com/download/. Accessed: 2015-03-31 (p. 75).

Thomas, David B., Wayne Luk, Philip H. W. Leong, and John D. Villasenor (2007). "Gaussian Random Number Generators". In: *ACM Comput. Surv.* 39.4, pp. 11+. ISSN: 0360-0300. DOI: 10.1145/1287620.1287622 (p. 78).

Tilly, Charles (1997). *Micro, Macro or Megrim?* (P. 2).

Tsai, Cheng-Han and Yi-Jang Lee (2012). "Focus on ADF/Cofilin: Beyond Actin Cytoskeletal Regulation". In: *ISRN cell biology* 2012, pp. 597876+. URL: http://www.isrn.com/journals/cb/2012/597876/cta/. DOI: 10.5402/2012/597876 (p. 96).

Uhrmacher, Adelinde M., Roland Ewald, Mathias John, Carsten Maus, Matthias Jeschke, and Susanne Biermann (2007). "Combining micro and macro-modeling in DEVS for computational biology". In: *Proceedings of the 2007 Winter Simulation Conference*. Ed. by S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton. Washington D.C.: IEEE Press, pp. 871–880. ISBN: 1-4244-1306-0. URL: http://portal.acm.org/citation.cfm?id=1351698 (p. 27).

Van Rheenen, Jacco, Xiaoyan Song, Wies van Roosmalen, Michael Cammer, Xiaoming Chen, Vera DesMarais, Shu-Chin Yip, Jonathan M. Backer, Robert J. Eddy, and John S. Condeelis (2007). "EGF-induced PIP2 hydrolysis releases and activates cofilin locally in carcinoma cells". In: *The Journal of Cell Biology* 179.6, pp. 1247–1259. ISSN: 1540-8140. DOI: 10.1083/jcb.200706206 (p. 96).

Van Zon, Jeroen S. and Pieter R. ten Wolde (2005a). "Green's-function reaction dynamics: a particle-based approach for simulating biochemical networks in time and space." In: *The Journal of Chemical Physics* 123.23, pp. 234910+. ISSN: 0021-9606. DOI: 10.1063/1.2137716 (p. 79).

– (2005b). "Simulating Biochemical Networks at the Particle Level and in Time and Space: Green's Function Reaction Dynamics". In: *Physical Review Letters* 94.12, pp. 128103+. DOI: 10.1103/physrevlett.94.128103 (p. 17).

Van Troys, M., L. Huyck, S. Leyman, S. Dhaese, J. Vandekerkhove, and C. Ampe (2008). "Ins and outs of ADF/cofilin activity and regulation". In: *European Journal of Cell Biology* 87.8-9, pp. 649–667. ISSN: 01719335. DOI: 10.1016/j.ejcb.2008.04.001 (p. 96).

Vera, Julio, Eva Balsa-Canto, Peter Wellstead, Julio R. Banga, and Olaf Wolkenhauer (2007). "Power-law models of signal transduction pathways". In: *Cellular Signalling* 19.7, pp. 1531–1541. ISSN: 0898-6568. DOI: 10.1016/j.cellsig.2007.01.029 (p. 21).

Von Smoluchowski, Marian (1917). "Versuch einer mathematischen Theorie der Koagulationskinetik kolloider Lösungen". In: *Zeitschrift für physikalische Chemie* 92.129-168, p. 9 (p. 21).

Waltemath, Dagmar, Jonathan Karr, Frank Bergmann, Vijayalakshmi Chelliah, Michael Hucka, Marcus Krantz, Wolfram Liebermeister, Pedro Mendes, Chris Myers, Pinar Pir, Begum Alaybeyoglu, Naveen Aranganathan, Kambiz Baghalian, Arne Bittig, Paulo Burke, Matteo Cantarelli, Yin Chew, Rafael Costa, Joseph Cursons, Tobias Czauderna, Arthur Goldberg, Harold Gomez, Jens Hahn, Tuure Hameri, Daniel Gardiol, Denis Kazakiewicz, Ilya Kiselev, Vincent Knight-Schrijver, Christian Knupfer, Matthias Konig, Daewon Lee, Audald Lloret-Villas, Nikita Mandrik, J. Medley, Bertrand Moreau, Hojjat Naderi-Meshkin, Sucheendra Palaniappan, Daniel Priego-Espinosa, Martin Scharm, Mahesh Sharma, Kieran Smallbone, Natalie Stanford, Je-Hoon Song, Tom Theile, Milenko Tokic, Namrata Tomar, Vasundra Toure, Jannis Uhlendorf, Thawfeek Varusai, Leandro Watanabe, Florian Wendland, Markus Wolfien, James Yurkovich, Yan Zhu, Argyris Zardilis, Anna Zhukova, and Falk Schreiber (2016). "Toward community standards and software for whole-cell modeling". In: *IEEE Transactions on Biomedical Engineering*, p. 1. ISSN: 1558-2531. DOI: 10.1109/TBME.2016.2560762 (p. 2).

Warnke, Tom, Tobias Helms, and Adelinde M. Uhrmacher (2015). "Syntax and Semantics of a Multi-Level Modeling Language". In: *Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. London, United Kingdom: ACM Press, pp. 133–144. ISBN: 9781450335836. DOI: 10.1145/2769458.2769467 (pp. 29, 54).

Wilhelmy, Ludwig (1850). "Ueber das Gesetz, nach welchem die Einwirkung der Säuren auf den Rohrzucker stattfindet". In: *Annalen der Physik und Chemie* 157.11. "The Law By Which the Action of Acids on Cane Sugar Occurs". Translated excepts at http://web.lemoyne.edu/~{}giunta/wilhelmy.html, pp. 413–428. DOI: 10.1002/andp.18501571106 (pp. 1, 20).

Wishart, David S., Robert Yang, David Arndt, Peter Tang, and Joseph Cruz (2005). "Dynamic Cellular Automata: An Alternative Approach to Cellular Simulation". In: *In Silico Biology* 5.2, pp. 139–161. URL: http://iospress.metapress.com/content/F17MR3NG55KGV3YC (pp. 19, 26).

Yamada, K. M., R. Pankov, and E. Cukierman (2003). "Dimensions and dynamics in integrin function." In: *Brazilian Journal of Medical and Biological Research* 36.8, pp. 959–

966. ISSN: 0100-879X. URL: http://view.ncbi.nlm.nih.gov/pubmed/12886449 (p. 96).

Zaidel-Bar, R., M. Cohen, L. Addadi, and B. Geiger (2004). "Hierarchical assembly of cell-matrix adhesion complexes." In: *Biochemical Society transactions* 32.Pt3, pp. 416–420. ISSN: 0300-5127. DOI: 10.1042/bst0320416 (p. 96).

Zamir, Eli and Benjamin Geiger (2001). "Molecular complexity and dynamics of cell-matrix adhesions". In: *Journal of Cell Science* 114.20, pp. 3583–3590. ISSN: 1477-9137. URL: http://jcs.biologists.org/content/114/20/3583.abstract (p. 96).

Acknowledgments

I am, first of all, grateful to Lin Uhrmacher, for her confidence in my research, for her advice on all scientific matters and for giving the occasional twist to the research topic.

My wife and my family supported me, showed patience and leniency when I was distracted and should not have been, and provided distraction when I was not and should have been.

I would also like to thank my coauthors and my colleagues at the MoSi group for many inspiring discussions and their insights into their work and its relationship to mine.

Theses

1. Modeling and simulation methods that consider biological cells systems with a homogeneous distribution of entities are better established than those that do not. However, there are many problems where the spatial distribution of actors matters.
2. There is a trade-off between simulation speed and level of detail in any model with spatially distributed particles.
3. ML-Space combines two approaches with different levels of detail, individuals with continuous space coordinates exhibiting Brownian motion, reacting by collisions, and populations in subunits of discretized space (subvolumes), reacting within these like in non-spatial simulation and diffusing between neighboring subvolumes.
4. ML-Space also comprises a rule-based modeling language that describes model species and their spatial properties along with their reactions and a system's initial state in one concise format.
5. The modeling language also supports multi-level structures, i. e. a hierarchical nesting of entities, like the language ML-Rules before it, but now with adherence to spatial constraints.
6. Entity attributes in ML-Space' language cannot only be used for multi-state species, but also to express upward causation and (indirect) downward causation.
7. ML-Space has been used successfully to reproduce previously published spatial models with multi-level aspects.
8. Through a cycle of *in silico* hypothesis generation using ML-Space and *in vitro* experiments, a model of actin filament formation and related regulatory mechanisms could be established.
9. Our ML-Space model of mitochondrial fission showed that limited availability of fission-enabling proteins can have opposite effects on overall mitochondrial health depending on further model assumptions w.r.t. the health regulation mechanism.
10. When subvolume-based simulation is applicable, it is generally orders of magnitude faster than individual-based simulation. This is true also for ML-Space' hybrid subvolume/individual simulator compared to purely individual-based simulation.
11. Fewer model parameters are needed for subvolume-based simulation, which makes the hybrid simulator also particularly suitable when details about amounts, sizes or diffusion constants of entities are uncertain or unknown, as fewer assumptions (or fewer parameter estimations) are required.
12. ML-Space is the first approach that combines individual, mobile compartments (especially dynamic ones w.r.t. creation, destruction and hierarchical composition) and subvolume-based simulation.