Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

MULTI-LEVEL MODELING AND SIMULATION OF CELL BIOLOGICAL SYSTEMS WITH ML-RULES – A TUTORIAL

Tobias Helms

Institute of Computer Science
University of Rostock
Albert-Einstein-Straße 22
18059 Rostock, GERMANY

Carsten Maus

Division of Theoretical Systems Biology
German Cancer Research Center
Im Neuenheimer Feld 280
69120 Heidelberg, GERMANY

Fiete Haack

Institute of Computer Science
University of Rostock
Albert-Einstein-Straße 22
18059 Rostock, GERMANY

Adelinde M. Uhrmacher

Institute of Computer Science
University of Rostock
Albert-Einstein-Straße 22
18059 Rostock, GERMANY

ABSTRACT

Multi-level modeling is concerned with describing a system at different levels of organization and relating their dynamics. ML-Rules is a rule-based language developed for supporting the modeling of cell biological systems. It supports nested rule schemata, the hierarchical dynamic nesting of species, the assignment of attributes and solutions to species at each level, and a flexible definition of reaction rate kinetics. As ML-Rules allows the compact description of rather complex models, means for an efficient execution were developed, e.g., approximate and adaptive algorithms. Experimentation with ML-Rules models is further supported by domain-specific languages for instrumentation and experimentation which have been developed in the context of the modeling and simulation framework JAMES II. A signaling pathway example will illustrate modeling and simulation with ML-Rules.

1 INTRODUCTION

Multi-level modeling plays an important role in many application areas, among those demography (Silverman et al. 2013) as well as molecular and cell biology (Noble 2008). Modeling means structuring the knowledge about a given system. To support this structuring a modeling language should provide a compact, succinct description of the system of interest. Particularly in disciplines, where the usual scientific experimental approach does not prepare the scientist well for developing formal models of the system of interest, the modeling language becomes crucial, also as Lazebnik put it for "overcoming a fear of long-forgotten mathematical symbols." (Lazebnik 2002).

In modeling and simulation not only semantics but also syntax matters. The syntax has a deep effect on "what inferences can be drawn from it, what kinds of idealizations will work well with it, etc." (Winsberg 2013). Therefore, domain-specific languages are required, which are tailored specifically to an application domain and offer "appropriate notations and abstractions, and an expressive power focused on, and usually restricted to, a particular problem domain" (Van Deursen et al. 2000). Based on this definition of domain-specific languages, a multi-level formalism for cell biology will likely not be entirely suitable for multi-level

modeling in other disciplines. Even in one area the most suitable modeling approach will vary, depending on the level of organization we are interested in. For example, for describing dynamics at cellular level the reactive systems metaphor might appear most suitable (Fisher et al. 2011) whereas for biochemical reaction networks, such as signaling pathways and metabolic networks, other more reaction-centric language approaches might be more suitable. To capture both biochemical networks and inter-cellular dynamics we developed the modeling language ML-Rules which supports variable structure modeling at intra and inter-cellular level. ML-Rules joins other rule-based approaches in systems biology in pursuing "Lazebnik's vision of precise formal modeling languages for biology" (Faeder 2011).

However, a modeling language with a simple syntax and an expressive semantics is, although crucial, only a pre-requisite for facilitating modeling and simulation of biological systems. As many experiments with the model are needed for validation (Rybacki et al. 2014), an efficient execution of simulation runs and a convenient definition of experiments facilitate a thorough analysis of the model, its successive refinement and enrichment and thus, its development. The tutorial will give a brief introduction how we dealt with the different challenges in the context of ML-Rules.

2 MULTI-LEVEL MODELING IN COMPUTATIONAL SYSTEMS BIOLOGY

2.1 Flat vs. Hierarchical Model Structure

Although "multi-level" and "multi-scale" are often used synonymously to characterize systems described at multiple layers or levels of organization, in this tutorial we do not want to focus on the different spatial and temporal scales at which dynamic processes of such systems are typically operating but on their structuring into hierarchical levels which might be subject to dynamics themselves, and relating dynamics between different levels of such hierarchies. In biology, hierarchical organization typically refers to a spatial nesting: proteins being located in the lysosome, the lysosome being located in the cytosol, the cytosol being part of a cell, etc. In modeling these nested structures, two general strategies can be distinguished: those in which the structure is flattened, and those in which the structure is explicitly encoded (see Fig. 1).

Prominent examples of flat formalisms applied in the field of systems biology are classic species-reactions networks, Petri nets (Heiner et al. 2008), or rule-based approaches like BioNetGen (Faeder et al. 2009). Flat approaches do not support hierarchical structuring of models. That means, different organizational levels can only *implicitly* be described, typically by defining multiple similarly named state variables for the same kind of model entity residing within different localities, e.g., a protein species P located within the cytoplasmic compartment (P_{cyt}) and the nucleus (P_{nuc}) of a cell. The same holds true for modeling according state transitions or interaction rules like biochemical reactions, which may need to be defined numerous times once for each location they could take place in. For example, two different reactions for the degradation of protein P, one for its degradation within the cytoplasmic compartment and another one for the same reaction taking place within the nucleus. Moreover, since all hierarchical relationships remain implicit, flat formalisms hamper modeling of inter-level causation (i.e., causation upward and downward the hierarchy).

To overcome such problems, various modeling languages with explicit support for hierarchical model structures have been developed. To those belong hierarchical extensions of originally flat formalisms, e.g., cBNGL, the rule-based BioNetGen language extended by explicit means for modeling nested compartments (Harris et al. 2009), as well as inherently hierarchical modeling approaches, like Statecharts (Harel 1987), the "Discrete Event System Specification" DEVS (Zeigler et al. 2000), and Bigraphs (Milner 2006), the latter of which supports also dynamic nesting structures. A crucial feature for multi-level modeling is also to equip the containing structures, e.g., cells or intra-cellular compartments, with an own state and behavior, which, however, is not supported by many hierarchical modeling approach. In cBNGL, for instance, compartment volumes are constant and assigning additional states is not possible. With Statecharts, by contrast, there is no distinction between behavioral and structural parts of a model and therefore each component at each hierarchy level may has its own state and dynamics.

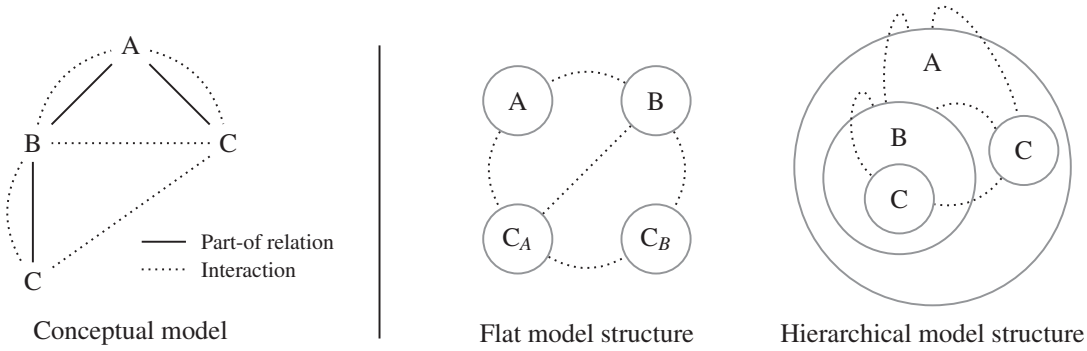

Figure 1: In a flat model, all elements reside equally side-by-side so that the model's structure is merely determined by interaction relations. Part-of relations of the conceptual model can be encoded only implicitly through according annotations. By contrast, hierarchical relations can be made explicit by nested model structures; here: both A and B enclose an entity of type C, while B is part of A. Figure modified from Maus (2013).

2.2 Suitable Modeling Paradigms

Regardless of classifying into flat and hierarchical approaches, modeling languages can be additionally characterized by the general modeling paradigms they employ: an object-centered reactive systems metaphor, e.g., Statecharts and DEVS, or a reaction-centric model perception (activity scanning approach), e.g., Petri nets, Bigraphical Reactive Systems, BNGL, and React(C). The general modeling paradigm has a strong impact on the suitability of a language for expressing certain kind of dynamics. While reactive systems approaches are well-suited for modeling dynamics at the cellular level (Fisher et al. 2011), a reaction-centric language is typically a good choice for modeling biochemical reaction networks (Danos et al. 2009, Heiner and Gilbert 2011). Colors or attributes that are assigned to the processes, species, or tokens and based on which the reactions or transitions can be constrained add to the expressiveness of these approaches (John et al. 2011).

As it is difficult to find a good compromise between expressiveness and usability due to the diversity of the different organizational levels that are subject of investigation in systems biology, ranging from molecular dynamics, over biochemistry, cells and cell populations, up to whole organism's physiology (Maus 2013, p. 122), multi-level modeling approaches should not try to cover all of them but focus on a few levels of organization. ML-Rules is tailored to biochemical reactions, cellular dynamics, and interactions within cell populations, since these levels are probably the most relevant in the field of systems biology, as is indicated by the number of publications and according models in the BioModels Database (Li et al. 2010), for example. Therefore, ML-Rules shares some similarities with React(C), which constitutes a flat colored rule-based approach (John et al. 2011).

3 BIOLOGICAL EXAMPLE

To illustrate the concepts of ML-Rules, we use a simplified model of the canonical Wnt signaling pathway (Mazemondet et al. 2012). This pathway plays a central role in stem cell homeostasis and differentiation. Accordingly Wnt signaling is involved in embryonic development, but also in a number of human cancers and developmental disorders. However, the exact mechanisms for the activation and regulation of Wnt signaling are not completely understood yet. Therefore a computational, stochastic model of Wnt signaling has been previously implemented in ML-Rules, based on experimental measurements (Mazemondet et al. 2011). It comprises four hierarchical levels (extracellular, cell, membrane/nucleus, protein) and represents

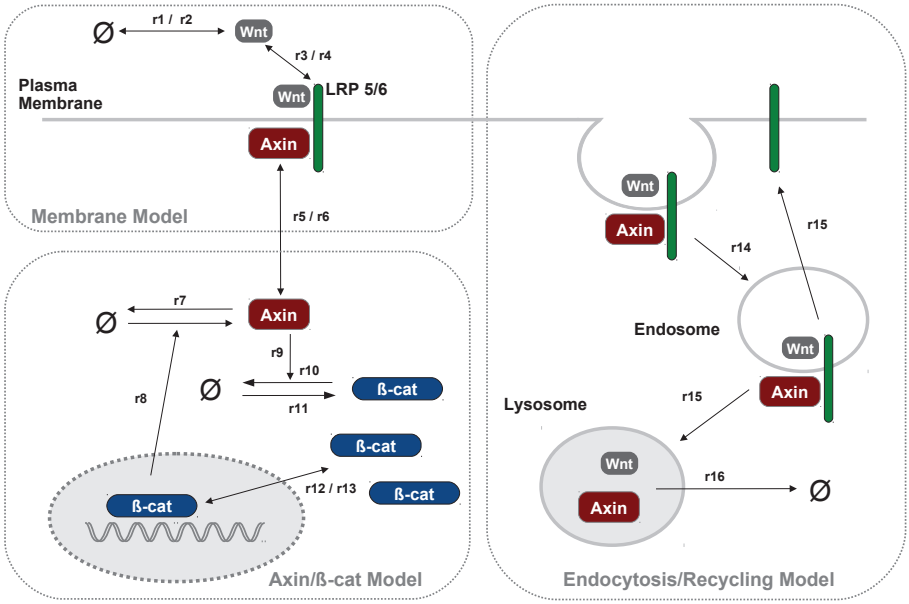

Figure 2: Example Model of canonical Wnt signaling and potential extension including endocytotic cycle.

the essential dynamics of four proteins (Wnt, β-catenin, LRP6 and Axin), being the key players that control the activation of Wnt signaling. A schematic view of the simplified Wnt signaling model is depicted in Figure 2. Basically, the signal transduction gets activated by extracellular Wnt proteins, that bind to receptors (LRP6) in the membrane. The activated receptor may now recruit and bind Axin from the cytosol. In the normal, inactive state Axin acts as general inhibitor by continuously degrading β-catenin. With Axin bound to the activated LRP6 receptor, β-catenin accumulates in the cytosol and is subsequently transported into the nucleus, where it activates specific genes. The relocation of β-catenin into the nucleus is the main indicator for the activation of canonical Wnt signaling. The model thus demonstrates how an extracellular signal is typically transduced into the cell, passing various cellular structures, such as cell membrane and cell plasma until it eventually reaches the nucleus, activating specific pathway-related genes. Membrane, plasma, and nucleus are all part of the cell. Therefore, the basic model represents a simple, but typical example for the hierarchical modeling of cellular signal transduction.

The model may now serve for further studies, like fitting and validation experiments, sensitivity and time-scale analysis; or for further model extensions. For illustration purpose, one such potential extension of our model is depicted on the right-hand side of Figure 2. In fact, ligand/receptor/protein complexes, such as the Wnt/LRP6/Axin complex in our model, are typically internalized some time after activation, primarily for sorting and recycling purpose. This means, the entire complex gets encapsulated in a small cytosolic compartment termed endosome, which is responsible for selective sorting and transport mechanisms. Thereby the LRP6 receptor is recycled, i.e., unbound from its complex and relocated to the membrane, whereas the remaining proteins are delivered to lysosomes for degradation. To model these dynamics, the modeling approach has to provide means for dynamic nesting structures.

4 ML-RULES MODEL SPECIFICATION

4.1 Main Concepts of the Language

ML-Rules is a rule-based language for modeling biological systems and their dynamics at different levels of a nested hierarchy (Maus, Rybacki, and Uhrmacher 2011, Maus 2013). Its semantics is discrete population-

based and translates basically to continuous-time Markov chains (CTMCs), meaning that the number of different model states can be infinite and each transition from one state to another follows an exponentially distributed random waiting time (Wilkinson 2011). Consequently, and since this is often required for describing cell biological systems at multiple levels, stochasticity is an essential feature of ML-Rules.

The basic model entities in ML-Rules are called *species*, which may represent any object of interest, e.g., a cell or a protein. Each species consists of a name and a fixed tuple of attributes, i.e., in the basic language concept attributes do not have names and are identified by their position only. Attributes are written within parenthesis behind the species name, e.g., `A("on",5)` describes a species with name "A" and two attributes "on" and "5". In addition, ML-Rules supports the concept of *nested species* to build hierarchical model structures, i.e., species can be enclosed by other species and can enclose other species themselves. That means, species are not only characterized by their names and attributes, but also by their context (the species they are enclosed by) and content (the set of species they contain, called *solution*). Nested species are specified with the help of square brackets, so that the abstract example from Fig. 1 would look like `A[B[C]+C]`. Note that species at any level within such a hierarchy may still have assigned attributes, e.g., `A[B(1)[C("off")]+C("on")]`.

Since ML-Rules belongs to the reaction-centric family of rule-based formalisms, the dynamics of a model are described by so called *rule schemata*, each of which may encode for possibly infinitely many concrete rule instantiations, helping to effectively reduce redundancy and thereby facilitating compact model descriptions (Danos et al. 2009). A rule schema consists of three parts: a set of reactant species, a set of product species, and a firing rate. The general notation for specifying a rule schema is as follows:

```
reactants -> products @ rate
```

A rule schema can be instantiated at any (sub-)solution of the current model state to which the set of reactants would match, i.e., the rule schema `C -> D+E`, which produces both species D and E from one C (rates are omitted), would lead to two instances of the rule at two different levels of the hierarchy when applying it to solution `A[B[C]+C]`.

The firing rate $r \in \mathbb{R}_0^+$ of a rule determines the frequency with which a rule is being executed. To let the rate depend on the amount of matched reactants, so called *species identifiers* can be defined through which the according species population size can be dynamically accessed. For example, the above rule could look like `C:c -> D+E @ #c`, in case its rate shall be proportional to the amount of species C within a given solution. In contrast to many other formalisms, e.g., the π-calculus and the κ-calculus, rate kinetics in ML-Rules are not restricted to the law of mass-action, which is an important feature for multi-level modeling in general (Mazemondet et al. 2009, Maus 2013). Complex mathematical expressions and conditional constraints are allowed to manipulate the reaction rate of a rule schema, e.g., to specify thresholds that control a rule to only fire if a certain amount of reactants is available.

To model upward and downward causation between different hierarchical levels, ML-Rules supports the specification of rule schemata that involve nested reactant and product species. In the same way, changing the model structure dynamically becomes possible by specifying nested reactants and products of a rule, which is another important feature for specifying biological multi-level models, since many biological processes, e.g., endocytosis, cell division, and death, change the hierarchical composition of the system. The support for variable model structures during runtime and the fact that species are allowed to have attributes (states) at any organizational level, distinguishes ML-Rules clearly from DEVS and cBNGL. Please note, extensions of DEVS offer these features, though, e.g., ML-DEVS (Uhrmacher et al. 2007). Similarly as has been described above, nested reactants and products are specified by using square brackets. For example, the following rule describes the release of a species C from a species B that encloses C:

```
B[C] -> B + C @ ...
```

Additionally, to bind the *remainder-solution* of B, a special variable of type `<name>?` can be introduced:

```
B[C + sol?] -> B[sol?] + C @ ...
```

In this case, the special variable `sol?` binds all species contained by the matched species B, except the one (C) explicitly specified. Without this variable, all enclosed species would get lost after firing, because the semantics assumes product species being substitutes and not modifications of the reactants.

4.2 The Wnt Signaling Model in ML-Rules

To further illustrate the concepts of ML-Rules, essential parts of the Wnt signaling model encoded in ML-Rules (Fig. 3) will be described in the following.

First of all, constants like reaction rate coefficient and initial species amounts need to be defined (line 2). Next, species types are defined, i.e., valid names and according attribute numbers (lines 5-6). Most species in the example have no attributes, except of `Endo`, `Lyso`, and `Lrp6`, each of which has exactly one attribute, designated by `<name>(1)`. In case of the compartmental species `Endo` and `Lyso`, the attribute shall represent the compartment's volume and the attribute of the LRP6 species is a binary attribute used to describe whether the receptor has bound (`Lrp6("b")`) a Wnt molecule or is free, i.e., is unbound (`Lrp6("u")`). Note that at this point ML-Rules does not distinguish between "compartment" and "atomic" species, i.e., each of them may contain a set of other species.

Next, the initial solution needs to be defined, starting with the keyword `>>INIT` (lines 9-18). In our example, the initial amounts of species are specified by model parameters, e.g., `initCell` specifies the initial amount of cells in the model. Similarly, the initial volume size of the lysosome is determined by a constant parameter (`initVolL`). Each cell in this initial solution contains one nucleus (`Nucl`), a membrane (`Memb`), a lysosome (`Lyso`), as well as a couple of Axin and β-catenin protein molecules (`initAxin Axin` and `initBcatC Bcat`). In addition, the nucleus contains a number of β-catenins (`initBcatN Bcat`), the membrane a number of LRP6 receptors (`initLrp6 Lrp6`), and in the extracellular compartment at the highest level of the model hierarchy, the initial solution includes a number of Wnt proteins (`initWnt Wnt`).

Finally, the dynamics of the model are defined in terms of a set of rule schemata (lines 21-52). All rates of the example are following the law of mass-action, i.e., the probability that a rule fires is proportional to the amount of reactant species. Since sometimes the context is of importance for a certain reaction, species at superior levels might be part of the rule schema, e.g., Axin degradation is only allowed within the cell compartment, which can be described by specifying the context explicitly with the help of a nested reactant species (line 33). Note that in this case the amount of the explicitly specified context species consequently also needs to be considered within the rate expression.

From the multi-level modeling aspect, the endocytosis and receptor recycling rules in lines 46-48 are of particular interest, as here the structure of the model changes significantly due to introducing new compartments and merging existing ones. Note that the volume of the lysosome increases by the endosome volume when both compartments fuse (`volL+volE`). In this model, compartment volumes could also be ignored as they do not contribute to the overall dynamics, however, they could be used to model volume-dependent rates of bimolecular reactions taking place within the compartments, for instance.

5 ML-RULES MODEL EXECUTION

The simulation algorithm of ML-Rules bases on the "Direct Method" of the exact "Stochastic Simulation Algorithm" (SSA) proposed by Gillespie (1977), i.e., ML-Rules models are executed in a discrete event-based manner. Figure 4 illustrates the basic algorithm. At first, for each rule schema, all rules based on the current state are computed (line 2-5). For this, the "matched reactants" are computed for each rule schema (line 3), i.e., for each reactant of a rule schema, the set of matching reactants is determined. The sets of reactants is afterwards used to compute concrete rules for the chosen rule schema (line 4), i.e., each valid combination of the reactants needs to be found. Next, the rule to fire is chosen depending on its proportion of the propensity sum of all rules (line 6). Afterwards, all reactants of the chosen rule are removed from the current model solution and the products of the chosen rule are inserted (line 7-12). Finally, the simulation time advance is sampled exponentially based on the propensity sum of all rules.

```
1  //Constant parameters
2  initBcatC:10e3; initBcatN:200; k1:0.01;...
3
4  //Species definitions
5  Cell(0); Memb(0); Nucl(0); Endo(1); Lyso(1);
6  Wnt(0); Lrp6(1); Axin(0); Lrp6Axin(0); Bcat(0);
7
8  //Initial solution
9  >>INIT[
10    initCell Cell[
11       Memb[ initLrp6 Lrp6("u")] +
12       Nucl[ initBcatN Bcat] +
13       Lyso(initVolL) +
14       initAxin Axin +
15       initBcatC Bcat
16    ] +
17    initWnt Wnt
18 ];
19
20 //Production and degradation of Wnt
21 Cell[C?]:c -> Cell[C?] + Wnt @ k1*#c;
22 Wnt:w -> @ k2*#w;
23
24 //Receptor activation/inactivation through Wnt ligand binding/unbinding
25 Wnt:w+Cell[Memb[Lrp6("u"):r+M?]+C?]:c->Cell[Memb[Lrp6("b")+M?]+C?]@k3*#w*#c*#r;
26 Cell[Memb[Lrp6("b"):r + M?] + C?]:c -> Wnt + Cell[Memb[Lrp6("u") + M?] + C?] @ k4*#c*#r;
27
28 //Binding and unbinding of Axin to/from activated LRP6 receptor
29 Memb[Lrp6("b"):r + M?] + Axin:a -> Memb[Lrp6Axin + M?] @ k5*#r*#a;
30 Memb[Lrp6Axin:r + M?] -> Memb[Lrp6("b") + M?] + Axin @ k6*#r;
31
32 //Degradation and synthesis of Axin
33 Cell[Axin:a + C?]:c -> Cell[C?] @ k7*#c*#a;
34 Nucl[Bcat:b + N?] -> Nucl[Bcat + N?] + Axin @ k8*#b;
35
36 //Degradation and synthesis of beta-catenin
37 Bcat:b + Axin:a -> Axin @ k9*#b*#a;
38 Cell[Bcat:b + C?]:c -> Cell[C?] @ k10*#c*#b;
39 Cell[C?]:c -> Cell[Bcat + C?] @ k11*#c;
40
41 //Nuclear-cytoplasmic shuttling of beta-catenin
42 Bcat:b + Nuc[N?] -> Nuc[Bcat + N?] @ k12*#b;
43 Nucl[Bcat:b + N?] -> Bcat + Nucl[N?] @ k13*#b;
44
45 //Endocytosis and receptor recycling
46 Memb[Lrp6Axin:r + M?] -> Memb[M?] + Endo(initVolE)[Lrp6Axin] @ k14*#r;
47 Memb[M?] + Endo(volE)[Lrp6Axin:r + E?]:e + Lyso(volL)[L?]
48                -> Memb[Lrp6("u") + M?] + Lyso(volL+volE)[Axin + Wnt + E?] @ k15*#e*#r;
49
50 //Lysosomal protein degradation
51 Lyso(volL)[Axin:a + L?] -> Lyso(volL)[L?] @ k16*#a;
52 Lyso(volL)[Wnt:w + L?] -> Lyso(volL)[L?] @ k16*#w;
```

Figure 3: The multi-level Wnt signaling model encoded in ML-Rules. Due to limited space, most constant definitions are omitted. The numbering of rate coefficients k_x is consistent with the reactions r_x in Fig. 2.

```
1  Require: state, rule schemata
2    for all rule schema ∈ rule schemata do
3      matched reactants ← MatchReactants(rule schema, state)
4      rules ← Append(rules, CreateRules(rule schema, matched reactants, state))
5    end for
6    rule ← SSA(rules)
7    for all reactant ∈ rule.reactants do
8      RemoveReactant(reactant)
9    end for
10   for all product ∈ rule.products do
11     InsertProduct(product)
12   end for
13   t ← t + TimeAdvance(rules)
```

Figure 4: Pseudo-Code of the abstract simulation algorithm of ML-Rules.

The implementation of the ML-Rules simulation algorithm is integrated into the modeling and simulation framework JAMES II (Himmelspach and Uhrmacher 2007). Furthermore, we developed an editor (including syntax-highlighting and syntax-checking), parser and compiler for ML-Rules models. Referring to the simulation algorithm, we developed a plugin-based simulator to address specific computational challenges. For example, we developed three different plugins for the handling of species: a list-based implementation, a map-based implementation, and a "grid file"-based implementation. The overhead of the list-based plugin is small, but on the other hand no efficient selection of species is possible. The map-based plugin indexes the species by their type, i.e., an efficient selection by species types is possible. The third implementation uses the multi-key index data structure grid-file (Nievergelt, Hinterberger, and Sevcik 1984) to save the species. This plugin can also retrieve species efficiently based on their attribute values. Nevertheless, the organizational overhead of this data structure is significantly higher compared to the others, so that it is typically not clear which data structure to choose for achieving the best performance. Furthermore, the state of a model can dramatically change during the simulation, such that the computational requirements change as well and therefore also the most suitable configuration of the simulation algorithm changes. Thus, we applied an adaptive simulator which automatically selects and adapts the simulation algorithm to be used during the simulation (Helms et al. 2013). By using this adaptive simulator, the user does not have to configure the simulation algorithm any more and still receives good performance results, which is particularly helpful if the user is not familiar with data structures and computational analysis. Since the adaptive simulator can adapt the simulation algorithm and its configuration during runtime, it can perform better than any non-adaptive algorithm and configuration. The challenge of this approach is, of course, to learn when to use which algorithm effectively. The adaptive simulator relies on reinforcement learning (Sutton and Barto 1998), and, thus, on balancing exploration and exploitation, i.e., to explore the efficiency of algorithms and configurations or to exploit gained knowledge.

Besides these implementation and dynamic configuration efforts to speed-up the simulation, it is also possible to change the type of the simulation algorithm. A useful review of different mechanisms and techniques is given by Gillespie, Hellander, and Petzold (2013). One possibility is to switch to an individual-based approach (Sneddon, Faeder, and Emonet 2011), so that equal species are no longer merged to one population but treated as individual entities. Hybrid approaches, which treat specific species individually and others in a population-based manner could also be used (Hogg, Harris, Stover, Nair, and Faeder 2014). Another possibility is to introduce an acceptable loss of accuracy to save computational time. One common approximate variant of the SSA is τ-leaping (Gillespie 2001). This algorithm basically executes "leaps" over the time scale and estimates all reactions probably be fired within this time span. Many versions of τ-leaping have been developed so far; some to increase computational performance (Sandmann 2009, Tian and Burrage 2004, Harris and Clancy 2006), some to extend its applicability, e.g., for spatial models

(Marquez-Lago and Burrage 2007, Jeschke, Ewald, and Uhrmacher 2011). Recently, we also adapted this algorithm so that it can be used within ML-Rules and we achieved a significant speed-up with some of our models (Helms et al. 2013), including the presented Wnt signaling model.

A second common class of approximate algorithms are multi-scale algorithms (Cao et al. 2005, E et al. 2005). Here, "fast" reaction channels are considered continuously and quasi-stationary distributions are used (computed either analytically or empirically) to approximate the states of these channels. The remaining "slow" reaction channels are computed as usual. Even a combination of τ-leaping and the multi-scale algorithms has also already been developed (Cao and Petzold 2008). A multi-scale simulation algorithm for ML-Rules models is currently under development.

6 EXPERIMENTATION WITH ML-RULES

6.1 Instrumentation

The specific selection and early filtering of observed model data can significantly reduce the amount of data to be saved and can consequently improve the performance of the whole experiment (Helms et al. 2012). In many M&S software, only simple solutions for this process are available. For example, in SDML ("Strictly Declarative Modelling Language") all data is collected and saved automatically, without the possibility of an early selection and filtering (Moss et al. 1998). Other tools like Snoopy (Rohr et al. 2010) and COPASI (Hoops et al. 2006) allow selections (via checkboxes) to specify the entities of interest. Another approach is to add the observation specification directly to the model file like it is done, e.g., in MATLAB/Simulink and BioNetGen (Blinov et al. 2004). The "Open Simulation Instrumentation Framework" (OSIF) uses the aspect-oriented programming paradigm to separate modeling and instrumentation (Ribault et al. 2010). In JAMES II, instrumenters need to be implemented to specify the data of interest and observer patterns are used to realize the observation.

The above approaches have several drawbacks, e.g., complex instrumentations might be impossible or need to be coded by a developer. Although coding instrumentations like in OSIF or JAMES II enables the user to write arbitrary complex instrumentations, this approach is worthless if the experimenter is not familiar with programing. Furthermore, coding instrumentations is comparably time-consuming and error-prone. A versatile instrumentation language that enables the user to specify instrumentation queries in an easy and intuitive manner denotes an alternative approach. We developed such an instrumentation language to specify which data to collect, when to the save the data, and how to aggregate it (Helms et al. 2012). This language is inspired by the structured query language (SQL) and most of its parts are independent of concrete modeling formalisms. Once implemented for a specific formalism, the language enables the user to write complex instrumentation queries without the need to write source code. The system can also directly give error messages if the user specifies invalid queries. Additionally, queries could be used as an instrumentation specification for an experiment reused within different simulation systems.

To make the instrumentation language working with ML-Rules, some language-specific constructs were introduced, e.g., regarding attributes and hierarchies. An example query is shown in Figure 6.1. This query describes an instrumentation for a model (line 1) and counts species of type `Bcat` (line 2-3). The hierarchy of the species is considered (line 4), i.e., `Bcat` species with different context hierarchies are considered separately. Finally, species counts are collected only every second time unit (line 5). Applied to the initial solution of the Wnt signaling example in Fig. 3, a concrete observation of this query would comprise two entries: `"Cell[Bcat]"=10e3` and `"Cell[Nucl[Bcat]]"=200`.

6.2 Experiment Specification with SESSL

Creating concise and unambiguous experiment specifications is an important concern in M&S (Pawlikowski et al. 2002). The embedded domain specific language SESSL (Simulation Experiment Specification via a Scala Layer) helps experimenters to formulate such experiment specifications (Ewald and Uhrmacher 2014). As an embedded domain specific language (DSL), SESSL bases on a host language, i.e., Scala (Odersky

```
1  INSTRUMENT MODEL
2    OBSERVE COUNT(species.quantity)
3    WHERE species.name = "Bcat"
4    GROUP BY species.name hierarchy aware
5    EVERY 2.0 T;
```

Figure 5: A simple instrumentation query useful for the Wnt signaling model. Here, the "Bcat" species distinguished by their context hierarchy would be counted every second time unit.

et al. 2011), which is completely interoperable with Java. Consequently, many tools can be used to program SESSL experiment specifications, i.e., editors with syntax highlighting and code completion.

SESSL serves as an additional software layer between users and simulation systems, i.e., SESSL itself is independent of concrete simulation systems. To support a specific simulation system, a so-called "binding" must be implemented. After creating the "binding", core features of SESSL can directly be used to specify experiments using the chosen simulation system. Nevertheless, since SESSL is an embedded DSL, specific features of a simulation system can be considered easily by implementing according methods, i.e., the user is not restricted to SESSL's core features. Experiment specifications are composed from experiment facets. The experimenter has to specify only the facets which are important for the experiment. Furthermore, (some) dependencies between facets are determined and analyzed automatically, so that inconsistent experiment specifications are directly rejected by SESSL and error messages are shown.

Figure 6 illustrates the specification of a simple ML-Rules experiment. Initially, the SESSL classes and the JAMES II binding are imported (line 1-2). In line 5, the facets of the experiment are specified. Since the specified experiment is simple, only the `Observation` facet is added, which provides methods to observe model entities. For more complex experiments, further facets exists, e.g., `ParallelExecution` providing methods to parallelize simulation execution and `Report` providing methods to automatically create result reports. With all available facets, complex simulation experiments can be specified, e.g., simulation-based optimization experiments to estimate model parameters, and runtime performance analysis experiments. Within the lines 6-8, the model location, the simulation end time and the number of replications are set. More complex stop policies are possible (e.g., considering the wall-clock time) as well as more complex replication policies (e.g., considering confidence intervals). Different conditions can arbitrarily be nested via "and" and "or" constructs. Next, a parameter scan is defined (line 10): three values for the variable `initWnt` (the initial number of Wnt within the model) shall be used, i.e., altogether 9 replications are computed; three with each value of `initWnt`. Similar to the instrumentation language example (Figure 6.1) of the previous section, line 12 and line 13 define what to observe, i.e., the `Bcat` species within the Nucleus, and when to observe, i.e., every second time unit, beginning from 0 until 270. An integration of the instrumentation language in SESSL is currently not available; we plan to add this feature. Finally, from line 15 to 17, the methods to save the observed data are specified, i.e., the observed information is simply written into a CSV file.

6.3 Model Composition and Statistical Model Checking

Composition is a common approach for reusing models in order to create more complex ones (Szabo and Teo 2009). Since models are typically developed to answer specific questions, unfortunately, model composition can be difficult to apply automatically. For example, the contexts of the models that shall be composed can be conflicting (Szabo and Teo 2009). Compared to many technical areas, model composition in systems biology with reaction network models is even more difficult, because of the lack of widely accepted modular building blocks (Barabási and Oltvai 2004). Instead, models usually describe bio-chemical processes, which include several compartmental structures, e.g., a signaling pathway comprising reactions outside and inside of cells. Referring to ML-Rules, specific challenges arise when composing ML-Rules models (Peng et al. 2013), e.g., the consistency of compartments must be ensured. Furthermore, to support

```
1  import sessl._
2  import sessl.james._
3
4  execute {
5    new Experiment with Observation {
6      model = "file-mlrj:/path/to/model/Wnt.mlrj"
7      stopTime = 270.0
8      replications = 3
9
10     scan("initWnt" <~ (1000,2000,3000))
11
12     observe("Cell/Nucl/Bcat()")
13     observeAt(range(0, 2, 270))
14
15     withRunResult { results =>
16       CSVFileWriter("output.csv") << results.trajectory("Cell/Nucl/Bcat()")
17     }
18   }
19 }
```

Figure 6: An experiment specification of a simple experiment written in SESSL.

model composition, e.g., aggregation (Randhawa, Shaffer, and Tyson 2009), ML-Rules models would need according interfaces and interaction points (Röhl and Uhrmacher 2008). However, even with sufficient interfaces and interaction points, user interaction is probably needed (Krause et al. 2010).

To support users in composing respectively extending ML-Rules models by reusing other models, SESSL has been extended to enable the specification of probabilistic statements based on a variant of *linear temporal logic* (LTL) and *continuous stochastic logic* (CSL) (Peng et al. 2014). Experimenters can now specify hypotheses about ML-Rules models and can add according tests within SESSL experiment specifications. These hypotheses can not only be used to check the validity of an individual model, but also to check the validity of a composed model. For this, the algorithm automatically adapts and checks hypotheses of individual models used for a composed model. Combined with further features like hypotheses priorities, the algorithm can support experimenters to create semantically reasonable compositions. Peng et al. (2014) demonstrated the benefits with respect to compositions of Lotka-Volterra models written in ML-Rules. We are currently composing respectively extending the Wnt signaling model with a model about membrane related dynamics. Existing hypothesis about the basic Wnt pathway model, e.g., statements about the amount of β-catenin within the nucleus of a cell, should also hold for the composed model.

6.4 Standardized Modeling Formats and ML-Rules

For computational models in systems biology, the *Systems Biology Markup Language* (SBML) has become an important standardized modeling format (Strömbäck 2006). Providing methods to transform models of a modeling language to standardized formats like SBML and back has many advantages, e.g., models can simply be shared between researchers and the models can be used within different tools. On the other hand, standardized formats are always a compromise between the capabilities of individual modeling languages. Thus, there will be probably a mismatch of expressiveness between a modeling language and a standardized format. However, standardized formats are a valuable approach to identify and confirm the essentials of a set of modeling languages needed by a whole community within a specific domain.

To examine the relations between ML-Rules and SBML, we developed a mechanism to transform both into each other, i.e., ML-Rules models can be transformed automatically to SBML models and vice versa (Nähring et al. 2013). In our opinion this research is of importance as ML-Rules supports features possibly added to SBML in future, like dynamically nested structures (Hucka et al. 2010). In both transformation directions, we determined specific challenges and restrictions which are not supported by the target formalism. From SBML to ML-Rules, e.g., time-triggered events cannot be transformed properly,

because ML-Rules currently does not support time-triggered events. From ML-Rules to SBML, e.g., dynamic nesting of species and attributed species are challenging, as each nesting and attribute assignment of an ML-Rules species need to be mapped to a distinct species type in SBML, which typically results in a very large number of SBML species. Furthermore, an infinite number of possible nestings and assignments in ML-Rules need to be reduced into finite subsets in order to be represented in SBML. We suggest to use either additional meta-data to deal with these problems or to execute a finite number of simulation runs only considering recorded species, like it can be done in BioNetGen, for instance (Faeder, Blinov, and Hlavacek 2009). Despite all the challenges and restrictions, we were already able to automatically transform more than two-thirds of the models in the BioModels database (Li et al. 2010) (version of July 9th, 2012) successfully into the ML-Rules modeling language (Nähring et al. 2013).

7 CONCLUSIONS

In this tutorial paper, we presented the rule-based multi-level language ML-Rules (Maus et al. 2011, Maus 2013) and means to improve its execution performance (Section 5) and means to support users specifying and executing experiments with ML-Rules models (Section 6). ML-Rules was developed to enable modelers to write succinct and compact multi-level models of complex cell biological systems. Therefore, ML-Rules employs a rule-based modeling approach and supports the hierarchical dynamic nesting of species, the assignment of attributes and solutions to species at each level, and a flexible definition of reaction rate kinetics (Section 4). Basic features of ML-Rules were illustrated with the help of a signal transduction pathway model (Section 3). To perform complex experiments with ML-Rules models, different tools and approaches were presented that help users in their task. Firstly, an instrumentation language for ML-Rules allows experimenters to specify complex instrumentation queries (Section 6.1). Secondly, the specification of experiments is facilitated by SESSL (Section 6.2), an embedded domain specific language for the specification of simulation experiments written in SCALA (Ewald and Uhrmacher 2014). Here, also hypothesis about the model can be formulated and tested (Section 6.3). Finally, means are available to transform SBML models to ML-Rules and vice versa (Section 6.4). However, the work on ML-Rules has not finished yet. We plan various extensions of ML-Rules to improve its applicability and usability. For example, one planned extension is the support of time-triggered events, a basic feature of SBML which is currently missing in ML-Rules. Furthermore, we plan to develop a multi-scale simulation algorithm for ML-Rules. Currently, a sandbox stand-alone application is available to create ML-Rules models with an editor and to execute simulations with these models. The advanced simulation algorithms and the instrumentation language will soon be added to the sandbox tool (wwwmosi.informatik.uni-rostock.de/mosi/en/software). For power-users we refer to JAMES II (www.jamesii.org) and SESSL (www.sessl.org).

ACKNOWLEDGMENTS

This research was supported by the German research foundation (DFG), via research grants CoSA (UH 66/7-3), ESCeMMo (UH-66/14), the DiER MoSiS project and the research training group dIEM oSiRiS.

REFERENCES

Barabási, A.-L., and Z. N. Oltvai. 2004. "Network biology: understanding the cell's functional organization". *Nature Reviews Genetics* 5 (2): 101–113.

Blinov, M. L., J. R. Faeder, B. Goldstein, and W. S. Hlavacek. 2004. "BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains". *Bioinformatics* 20 (17): 3289–3291.

Cao, Y., D. T. Gillespie, and L. R. Petzold. 2005. "The slow-scale stochastic simulation algorithm". *The Journal of Chemical Physics* 122 (1).

Cao, Y., and L. R. Petzold. 2008. "Slow-scale tau-leaping method". *Computer Methods in Applied Mechanics and Engineering* 197:3472–3479.

Danos, V., J. Feret, W. Fontana, R. Harmer, and J. Krivine. 2009. "Rule-Based Modelling and Model Perturbation". In *Trans. on Comput. Syst. Biol. XI*, Volume 5750, 116–137. Springer Berlin Heidelberg.

E, W., D. Liu, and E. Vanden-Eijnden. 2005. "Nested stochastic simulation algorithm for chemical kinetic systems with disparate rates". *The Journal of Chemical Physics* 123 (19).

Ewald, R., and A. M. Uhrmacher. 2014. "SESSL: A Domain-specific Language for Simulation Experiments". *ACM Transactions on Modeling and Computer Simulation* 24 (2): 11:1–11:25.

Faeder, J. R. 2011. "Toward a comprehensive language for biological systems". *BMC Systems Biology* 9 (68).

Faeder, J. R., M. L. Blinov, and W. S. Hlavacek. 2009. "Rule-Based Modeling of Biochemical Systems with BioNetGen". In *Systems Biology*, Volume 500, 113–167. Humana Press.

Fisher, J., D. Harel, and T. A. Henzinger. 2011. "Biology As Reactivity". *CACM* 54 (10): 72–82.

Gillespie, D. T. 1977. "Exact Stochastic Simulation of Coupled Chemical Reactions". *The Journal of Physical Chemistry* 81 (25): 2340–2361.

Gillespie, D. T. 2001. "Approximate accelerated stochastic simulation of chemically reacting system". *The Journal of Chemical Physics* 115 (4): 1716–1733.

Gillespie, D. T., A. Hellander, and L. R. Petzold. 2013. "Perspective: Stochastic algorithms for chemical kinetics". *The Journal of Chemical Physics* 138 (17).

Harel, D. 1987. "Statecharts: a visual formalism for complex systems". *Science of Computer Programming* 8 (3): 231–274.

Harris, L. A., and P. Clancy. 2006. "A "partitioned leaping" approach for multiscale modeling of chemical reaction dynamics.". *The Journal of Chemical Physics* 125 (14).

Harris, L. A., J. S. Hogg, and J. R. Faeder. 2009. "Compartmental Rule-based Modeling of Biochemical Systems". In *Proceedings of the 2009 Winter Simulation Conference*, edited by A. Dunkin, R. G. Ingalls, E. Yücesan, M. D. Rossetti, R. Hill, and B. Johansson, 908–919: Piscataway, NJ: IEEE.

Heiner, M., and D. Gilbert. 2011. "How Might Petri Nets Enhance Your Systems Biology Toolkit". In *Applications and Theory of Petri Nets*, Volume 6709, 17–37.

Heiner, M., D. Gilbert, and R. Donaldson. 2008. "Petri Nets for Systems and Synthetic Biology". In *Formal Methods for Computational Systems Biology*, Volume 5016, 215–264. Springer Berlin Heidelberg.

Helms, T., R. Ewald, S. Rybacki, and A. M. Uhrmacher. 2013. "A Generic Adaptive Simulation Algorithm for Component-based Simulation Systems". In *Proceedings of the 2013 Workshop on Principles of Advanced and Distributed Simulation*, 11–22.

Helms, T., J. Himmelspach, C. Maus, O. Röwer, J. Schützel, and A. M. Uhrmacher. 2012. "Toward a Language for the Flexible Observation of Simulations". In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, 1–12: Piscataway, NJ: IEEE.

Helms, T., M. Luboschik, H. Schumann, and A. M. Uhrmacher. 2013. "An Approximate Execution of Rule-Based Multi-level Models". In *Proceedings of the 2013 International Conference on Computational Methods in Systems Biology*, 19–32.

Himmelspach, J., and A. M. Uhrmacher. 2007. "Plug'n simulate". In *Proceedings of the 2007 Annual Simulation Symposium*, 137–143.

Hogg, J. S., L. A. Harris, L. J. Stover, N. S. Nair, and J. R. Faeder. 2014. "Exact Hybrid Particle/Population Simulation of Rule-Based Models of Biochemical Systems". *PLoS Computational Biology* 10.

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

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

Jeschke, M., R. Ewald, and A. M. Uhrmacher. 2011. "Exploring the Performance of Spatial Stochastic Simulation Algorithms". *The Journal of Computational Physics* 230 (7): 2562–2574.

John, M., C. Lhoussaine, J. Niehren, and C. Versari. 2011. "Biochemical Reaction Rules with Constraints". In *Programming Languages and Systems*, Volume 6602, 338–357. Springer Berlin Heidelberg.

Krause, F., J. Uhlendorf, T. Lubitz, M. Schulz, E. Klipp, and W. Liebermeister. 2010. "Annotation and merging of SBML models with semanticSBML". *Bioinformatics* 26 (3): 421–422.

Lazebnik, Y. 2002. "Can a biologist fix a radio?–Or, what I learned while studying apoptosis.". *Cancer Cell.* 2 (3): 179–182.

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

Marquez-Lago, T. T., and K. Burrage. 2007. "Binomial tau-leap spatial stochastic simulation algorithm for applications in chemical kinetics". *The Journal of Chemical Physics* 127 (10).

Maus, C. 2013. *Toward Accessible Multilevel Modeling in Systems Biology: A Rule-based Language Concept*. Logos, Berlin. PhD thesis (University of Rostock, Germany).

Maus, C., S. Rybacki, and A. M. Uhrmacher. 2011. "Rule-based multi-level modeling of cell biological systems". *BMC Systems Biology* 5 (166).

Mazemondet, O., R. Hubner, J. Frahm, D. Koczan, B. Bader, D. Weiss, A. Uhrmacher, M. Frech, A. Rolfs, and J. Luo. 2011. "Quantitative and kinetic profile of Wnt/β-catenin signaling components during human neural progenitor cell differentiation". *Cellular & Molecular Biology Letters* 16 (4): 515–538.

Mazemondet, O., M. John, S. Leye, A. Rolfs, and A. M. Uhrmacher. 2012. "Elucidating the Sources of β-Catenin Dynamics in Human Neural Progenitor Cells". *PLoS ONE* 7 (8): e42792.

Mazemondet, O., M. John, C. Maus, A. M. Uhrmacher, and A. Rolfs. 2009. "Integrating Diverse Reaction Types into Stochastic Models - A Signaling Pathway Case Study in the Imperative pi-Calculus". In *Proceedings of the 2009 Winter Simulation Conference*, edited by A. Dunkin, R. G. Ingalls, E. Yücesan, M. D. Rossetti, R. Hill, and B. Johansson, 932–943: Piscataway, NJ: IEEE.

Milner, R. 2006. "Pure bigraphs: Structure and dynamics". *Information and Computation* 204 (1): 60–122.

Moss, S., H. Gaylard, S. Wallis, and B. Edmonds. 1998. "SDML: A Multi-Agent Language for Organizational Modelling". *Computational and Mathematical Organization Theory* 4 (1): 43–69.

Nähring, S., R. Ewald, A. Uhrmacher, and C. Maus. 2013. "From standardized modeling formats to modeling languages and back - An exploration based on SBML and ML-Rules". In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S. H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, 1359–1370: Piscataway, NJ: IEEE.

Nievergelt, J., H. Hinterberger, and K. C. Sevcik. 1984. "The Grid File: An Adaptable, Symmetric Multikey File Structure". *ACM Transactions on Database Systems* 9 (1): 38–71.

Noble, D. 2008. *The MUSIC of LIFE: Biology Beyond Genes*. Oxford University Press.

Odersky, M., L. Spoon, and B. Venners. 2011. *Programming in Scala*. 2nd ed. Artima.

Pawlikowski, K., H.-D. Jeong, and J.-S. Lee. 2002. "On Credibility of Simulation Studies of Telecommunication Networks". *IEEE Communications Magazine* 40 (1): 132–139.

Peng, D., R. Ewald, and A. M. Uhrmacher. 2014. "Towards Semantic Model Composition via Experiments". In *Proceedings of the 2014 Workshop on Principles of Advanced and Distributed Simulation*, 151–162.

Peng, D., A. Steiniger, T. Helms, and A. M. Uhrmacher. 2013. "Towards Composing ML-Rules Models". In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S. H. Kim, A. Tolk, R. Hill, and M. E. Kuhl: Piscataway, NJ: IEEE.

Randhawa, R., C. a. Shaffer, and J. J. Tyson. 2009. "Model aggregation: a building-block approach to creating large macromolecular regulatory networks.". *Bioinformatics (Oxford, England)* 25 (24): 3289–3295.

Ribault, J., O. Dalle, D. Conan, and S. Leriche. 2010. "OSIF: A Framework to Instrument, Validate, and Analyze Simulations". In *Proc. 2010 ICST Int. Conf. on Simulation Tools and Techniques*, 56:1–56:9.

Röhl, M., and A. M. Uhrmacher. 2008. "Definition and analysis of composition structures for discrete-event models". In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler, 942–950: Piscataway, NJ: IEEE.

Rohr, C., W. Marwan, and M. Heiner. 2010. "Snoopy—a unifying Petri net framework to investigate biomolecular networks". *Bioinformatics* 26 (7): 974–975.

Rybacki, S., F. Haack, K. Wolf, and A. M. Uhrmacher. 2014. "Developing simulation models - from conceptual to executable model and back - an artifact-based workflow approach". In *Proceedings of the 2014 International ICST Conference on Simulation Tools and Techniques*.

Sandmann, W. 2009. "Streamlined Formulation of Adaptive Explicit-Implicit Tau-Leaping with Automatic Tau Selection". In *Proceedings of the 2009 Winter Simulation Conference*, edited by A. Dunkin, R. G. Ingalls, E. Yücesan, M. D. Rossetti, R. Hill, and B. Johansson, 1104–1112: Piscataway, NJ: IEEE.

Silverman, E., J. Bijak, J. Hilton, V. D. Cao, and J. Noble. 2013. "When Demography Met Social Simulation: A Tale of Two Modelling Approaches". *Journal of Artificial Societies and Social Simulation* 16 (4): 9.

Sneddon, M. W., J. R. Faeder, and T. Emonet. 2011. "Efficient modeling, simulation and coarse-graining of biological complexity with NFsim". *Nature Methods* 8 (2): 177–183.

Strömbäck, L. 2006. "A Method for Comparison of Standardized Information Within Systems Biology". In *Proceedings of the 2006 Winter Simulation Conference*, edited by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 1603–1610: Piscataway, NJ: IEEE.

Sutton, R. S., and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Szabo, C., and Y.-M. Teo. 2009. "An Approach for Validation of Semantic Composability in Simulation Models". In *Proc. of the 2009 Workshop on Principles of Advanced and Distributed Simulation*, 3–10.

Tian, T., and K. Burrage. 2004. "Binomial leap methods for simulating stochastic chemical kinetics". *The Journal of Chemical Physics* 121 (21): 10356–10364.

Uhrmacher, A. M., R. Ewald, M. John, C. Maus, M. Jeschke, and S. Biermann. 2007. "Combining micro and macro-modeling in DEVS for computational biology". In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 871–880: Piscataway, NJ: IEEE.

Van Deursen, A., P. Klint, and J. Visser. 2000. "Domain-Specific Languages: An Annotated Bibliography". *Sigplan Notices* 35 (6): 26–36.

Wilkinson, D. J. 2011. *Stochastic Modelling for Systems Biology*. 2nd ed. CRC Press.

Winsberg, E. 2013. "Computer Simulations in Science". In *The Stanford Encyclopedia of Philosophy* (Summer 2013 ed.)., edited by E. N. Zalta.

Zeigler, B. P., T. G. Kim, and H. Praehofer. 2000. *Theory of Modeling and Simulation*. 2nd ed. Academic Press, Inc.

AUTHOR BIOGRAPHIES

TOBIAS HELMS is a PhD student at the Institute of Computer Science at the University of Rostock. His email address is tobias.helms@uni-rostock.de.

CARSTEN MAUS holds a PhD in Computer Science from the University of Rostock. Currently, he is a post-doc at the Division of Theoretical Systems Biology of the German Cancer Research Center (DKFZ) in Heidelberg, Germany. His email address is c.maus@dkfz-heidelberg.de.

FIETE HAACK is a PhD student at the Institute of Computer Science at the University of Rostock. His email address is fiete.haack@uni-rostock.de.

ADELINDE M. UHRMACHER is professor at the Institute of Computer Science of the University of Rostock and head of the Modeling and Simulation Group. She holds a PhD in Computer Science from the University of Koblenz and a Habilitation in Computer Science from the University of Ulm. Her email address is adelinde.uhrmacher@uni-rostock.de.