Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds.

FROM STANDARDIZED MODELING FORMATS TO MODELING LANGUAGES AND BACK — AN EXPLORATION BASED ON SBML AND ML-RULES

Sebastian Nähring
Roland Ewald
Adelinde M. Uhrmacher

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

Carsten Maus

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

ABSTRACT

Standardized model exchange formats give practitioners the freedom to choose the most suitable tool and facilitate both cross-validation and reproduction of simulation results. On the other hand, standardization necessarily implies a compromise between the capabilities of individual modeling languages and a common ground of concepts and underlying assumptions of the given application domain. This compromise often leads to a mismatch of expressiveness between modeling language and exchange format, which should be resolved automatically, e.g., by offering a transformation. We explore the challenges of such an approach for the Systems Biology Markup Language (SBML), a well-established model format in systems biology, and ML-Rules, a rule-based modeling language for describing cell biological systems at multiple interrelated levels. Our transformation approach can be extended both in terms of the heuristics it employs and in terms of the modeling formalisms it supports.

1 INTRODUCTION

The introduction of standardized modeling formats or languages facilitates the reuse of models between different tools (Mattson and Elmquist 1998). They give practitioners the freedom to choose the most suitable tool and facilitate cross-validation and reproduction of simulation results. As a community effort, language features are identified that are considered as standard and thus shall be supported by modeling and simulation tools in this area, or that appear promising or dearly needed in the future. In systems biology, different such standards have been proposed, e.g., CellML (Cuellar et al. 2003) and the Systems Biology Markup Language (SBML, Hucka et al. 2010). They are the basis for extensive curated model repositories, such as the BioModels database (Li et al. 2010). Thus a wide adoption of such standards allows not only to use different tools for the modeling effort at hand, but facilitates sharing models between researchers beyond the lifetime of the software used to create them. A pre-requisite for harvesting those benefits is a clear separation between model, executing those models, and experiments.

As standards are intended to capture the essentials of different modeling approaches and are based on negotiations within the community, they tend to lag behind the progress in modeling methods and do not necessarily capture all features of a modeling language. To transform a model from a modeling language into a standard format may thus lead to rather complicated model structures, to a partially translated model representation (consisting only of compatible model fragments), or to a failure if too little support exists. In addition, a language might not (yet) support all features of a standard. Standards change only slowly, which refers to including new features as well as getting rid of no longer needed ones. A concrete language

might focus on specific aspects of modeling and languages tend to evolve over time. Thus, not all languages might (wish to) support all features of a standard, even if only at the beginning.

Hence, there is typically a mismatch between features supported in the language and those of the standard. We will explore possibilities of how to transform models between language and standard format based on ML-Rules (Maus et al. 2011) and SBML. This endeavor is of particular interest for the systems biology community as ML-Rules supports some features announced to be subject of future extensions of SBML, i.e., multistate species, dynamically nested structures, and discrete spatial processes (cf. http://sbml.org/Documents/Specifications#SBML_Level_3_Packages). However, the transformation is quite a challenge: In contrast to SBML, reactions in ML-Rules are defined in terms of rule schemata, whose variables are bound during execution and thus over an infinite simulation run theoretically an infinite number of reactions can be generated. Furthermore, since ML-Rules combines this with dynamically nested structures, reactions might occur in infinitely many compartments.

2 STANDARD AND LANGUAGE IN A NUTSHELL

2.1 SBML

Over the last decade, SBML has emerged as the *de facto* standard format for computational models in systems biology (Strömbäck 2006). Like many other exchange formats, e.g., CellML or MathML, SBML is based on XML. Note that examples in this paper are given in the SBML shorthand notation from Wilkinson (2006) to enhance their readability. In addition, self-explanatory keywords for initial assignments (`@inits`) and functions (`@functions`) are used.

The development of SBML is structured into levels and versions. Whereas levels signalize bigger changes, e.g., the way how to encode mathematical formulas, versions denote smaller updates of the standard. As the diversity of modeling languages in systems biology (see, e.g., de Jong 2002 and Bittig and Uhrmacher 2010) asks for more and more extensions of the standard, the community now distinguishes between core SBML and different extension packages. At the most recent SBML specification (level 3, version 1), a model contains a finite set of species as well as a fixed number of compartments with predefined volumes. Each species is assigned to one of those compartments, however, this assignment is static and can not change during simulation. Reactions between species are defined to take place in those compartments and can have arbitrary kinetic rate expressions. In addition, time-triggered and situation-triggered events can be defined. So called rules allow to specify algebraic expressions between species.

2.2 ML-Rules

In biology, molecules, individual cells, and cell populations denote different levels of an organizational and functional hierarchy, each of which with its own dynamics. Multilevel modeling is concerned with describing a system at such different levels and relating their dynamics, which has been the motivation for developing multilevel modeling approaches like ML-Rules, EMSY (Uhrmacher 1995), ML-DEVS (Uhrmacher et al. 2007), or colored stochastic multilevel multiset rewriting (Oury and Plotkin 2011). Hence, multilevel modeling implies hierarchical nesting of model entities and explicit support for upward and downward causation between the dynamics at these different levels.

In the area of systems biology, rule-based modeling approaches are gaining increasingly attention due to enabling a concise and compact reaction-centric description of biochemical systems (Faeder 2011). ML-Rules presents a rule-based multilevel modeling approach, where species may have assigned arbitrary attributes (states) and solutions (multisets of enclosed species) to describe a hierarchy of nested entities. Rule schemata are dynamically instantiated and applied to various solutions, i.e., to nested species as well as species nested within others. Thereby, arbitrary reaction rate kinetics and constraints allow for specifying state transitions in a flexible manner.

3 TRANSFORMATION FROM SBML TO ML-RULES

Many basic elements from SBML can straightforwardly be mapped to ML-Rules, such as parameters, compartments, species, and initial assignments.

Parameters are supported by ML-Rules in the form of constants, so they can be expressed directly. ML-Rules also allows mathematical expressions for the definition of their values, so initial assignments from an SBML model can be easily integrated as well. Compartments can be expressed by ML-Rules species that have an attribute representing the compartment's volume. Species from SBML itself are mapped to attribute-less species in ML-Rules. In ML-Rules, the initial solution (shown as `>>INIT[...]`) defines the initial state of the model. Each species from SBML that has an initial value greater than zero (either through their set value or an initial assignment) will be added to this solution. To represent the assignment of species to compartments, transformed species can be nested within the solution of the species representing the compartment.

Reactions given in SBML can be transformed to rule schemata by using the transformed species as reactants and products. If the rate expression of the reaction refers to elements that are not reactants (e.g., modifiers or elements with assignment rules), they have to be added to the reactants in ML-Rules as this is the only way to access their value. To prevent those species from degradation, they also have to be added to the product side.

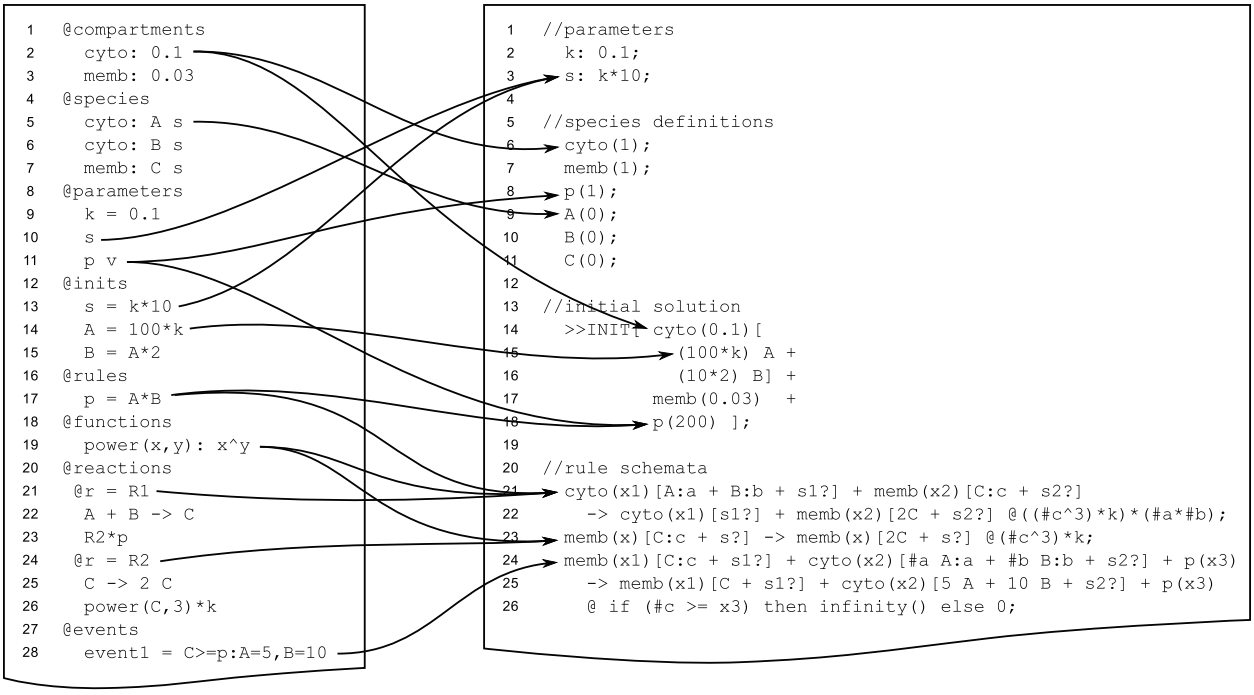

Figure 1: Example transformation from SBML to ML-Rules. SBML compartments and species are transformed to species in ML-Rules. Initial assignments will be preserved if possible and included in the initial solution. Since the parameter `p` is variable (indicated through letter v) it will become a species in ML-Rules, thereby representing its value as a possibly dynamic attribute. Species `memb` and `C` had to be included in the rule schemata because their kinetic expression refer to the molecule count of `C`. In ML-Rules one can only reference a species amount if it is defined as a reactant within the rule schema. So there is also an additional product species `C` to preserve the added species. Function definitions are evaluated during transformation.

Table 1: Overview of transformation from SBML to ML-Rules. (Left column) SBML level 3 core components, (middle) specific properties, (right) mapping to corresponding ML-Rules elements.

SBML component	Restriction	Representation in ML-Rules
Function definitions		none, will be expanded and replaced during transformation
Unit definitions		none, will be ignored
Compartments		as species with the compartment size as its attribute
Species		as attribute-less species, initial concentration or amount will be transformed to particle count
Parameters	`constant` = **true**	as constant parameter
	`constant` = **false**	as attributed species
Initial assignments		as part of the initial solution
Rules	Algebraic	not supported yet
	Assignment	partly supported, species with assignment rules will be added to respective rule schemata
	Rate	partly supported, rate of change of species amounts, compartment sizes, and variable parameters will be transformed to respective rule schemata with discrete firing rates
Validity constraints		not supported yet
Reactions		as rule schemata
	`fast` = **true**	not supported yet
	`reversible` = **true**	will be split into two rule schemata with adjusted rate expressions
Events	general	as rule schemata, firing is constrained by the trigger condition
	with priority, delay or timed trigger	not supported yet

Some SBML events can be transformed to rule schemata, as ML-Rules allows the definition of constraints within rate expressions. If the condition within such a rate does not hold, the rate can be set to zero and thereby an event trigger condition can be mimicked. To ensure that a rule may be selected as the next rule, the special rate value `infinity()` can be used. This tells the simulator that the rule shall be executed immediately without time advancement. If there are multiple rules with this rate at a given time, one of them is chosen randomly. A sample transformation from SBML to ML-Rules is shown in Figure 1.

Some of the SBML components can not be expressed in ML-Rules due to missing equivalent features. These are algebraic rules (as a linear equation systems can not be expressed in ML-Rules), events with delays or priorities, and mathematical expressions using the simulation time (like trigger conditions of events), as no modeling construct exists in the current version that allows to access the simulation time. Overall, for the curated models in the BioModels database (version of July the 9th, 2012) 303 of the 421 available models can be fully transformed into ML-Rules models. An overview of the components and their transformation is shown in Table 1.

4 TRANSFORMATION FROM ML-RULES TO SBML

As with the transformation in the other direction, some ML-Rules elements can be mapped to SBML easily. Parameters can be straightforwardly mapped (with an additional initial assignment), as already shown in Figure 1. Another easy to map element is the initial solution of an ML-Rules model, which defines the model's state at the beginning of its execution. In SBML this is represented by either special values within the definitions of species, compartments, and parameters, or with the use of initial assignments. To transform the initial solution to SBML, we decided to use initial assignments if the initial value is determined by a mathematical expression and to use the dedicated attributes if it is instead represented by a single number. In contrast to SBML, species in ML-Rules can have arbitrary attributes by which they are distinguished. Therefore, each possible assignment of attribute values has to be mapped to an individual SBML species. This implies that *every possible* attribute value combination is known for every ML-Rules species at the time of the transformation. An explanation on how this can be achieved is presented in the

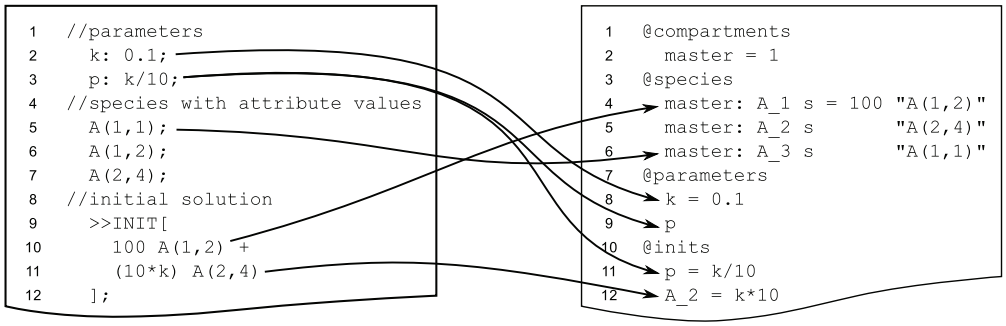

Figure 2: Transformation of basic ML-Rules elements to SBML. Constants are mapped directly and their value or an initial assignment is set accordingly. Each species with a defined combination of attribute values is transformed to a species in SBML. The initial solution determines the initial assignments in the SBML model.

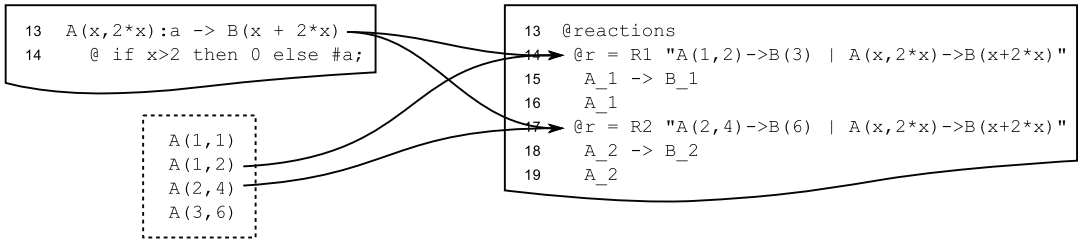

Figure 3: Transformation from rule schemata to reactions. On the left a sample rule as well as a list of possible attribute values of species A is shown. Each combination is inserted into the rule and tested for validity. Species `A(1,1)` does not fulfill the rule's condition, hence no reaction is created for it. Similarly, the rate for species `A(3,6)` evaluates to zero and therefore in this case also no reaction is created.

paragraph below on flat models. An example of the transformation of some basic elements is given in Figure 2.

Given the possible attribute values of each species, we can also map rule schemata to reactions. However, to do so requires some efforts, as rule schemata in ML-Rules are defined on variables and those need to be instantiated by concrete species in SBML to form the reactions. Therefore we put in every possible combination of attribute values for variables and check whether they produce valid rules. Some rules may restrict possible combinations in such a way that, e.g., the second attribute of a species has to be twice as high as its first attribute. If an attribute value combination is valid, the rate expression can be calculated. A positive rate suggests that the rule can be executed with this combination and a reaction can be created for it. The rule can not be executed if its rate becomes zero, thus no reaction will be created for it. An example for a rule schema transformation is given in Figure 3 and an overview of the mapping between ML-Rules and SBML can be found in Table 2.

4.1 Flat models

As mentioned above, transforming the species requires to know what attribute combinations can occur within a model. Values are defined in two ways: First the initial solution defines the values for species at the beginning of model execution. New values can only be produced through products within rule schemata or more specific through the assignments to attributes within each species in such schemata. To now calculate all possible attribute combinations we construct a directed bipartite graph based on attributes and their value assignments. One node type represents the attributes of each species (now called attribute nodes) whereas the other node type represents mathematical expressions given in a product of a rule (expression

Table 2: Overview of transformation from ML-Rules to SBML. (Left column) ML-Rules model elements, (middle) specific properties, (right) mapping to corresponding SBML components.

ML-Rules component	Restriction	Representation in SBML
Constant parameter	general	as parameter with `constant` = **true**
	math. expr. as value	as parameter with an additional initial assignment
Species definition		none, only concrete species are relevant
Concrete species	general	as species with `name` attribute representing the attribute values
	Nested species (contained solution)	each possible combination of species nested within the solution becomes a concrete species
Initial solution		a concrete species with an initial assignment
Rule Schema	general	set of reactions, each concrete and unique attribute combination of reactants becomes a single reaction
	constrained rates (independent on species counts)	same as before, but combinations with a zero-rate will be ignored
	constrained rates (dependent on species counts)	with two additional events changing a newly added parameter depending on the outcome of the condition, each such rule will be split into two reactions with adjusted rates

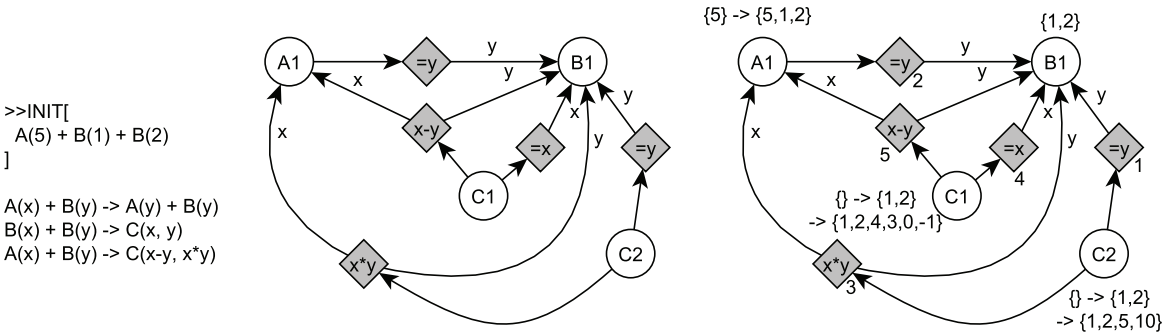

Figure 4: Dependency graph for calculating attributes. This figure shows the graph that was created by the model excerpt given on the left. The letters on edges represent the role an attribute plays within the connected expression node. The right graph shows the steps needed to calculate all possible attributes. Numbers below each expression node determine their order of execution. On each attribute node the values in braces represent possible values for this attribute after each step of execution.

nodes) that calculate new values for attributes. An edge from an attribute node to an expression node is drawn if the expression calculates a value for this attribute. On the other hand an edge from an expression node to an attribute node is drawn if the expression needs a value from this attribute for its calculation. For example, consider the following rule:

```
A(x) + B(y) -> C(x-y, x*y)
```

A graph would consist of four attribute nodes, as species `A` and `B` have one attribute whereas `C` has two. There would be two expression nodes, one for each of the attributes of species `C`. Both expression nodes would have edges to the attribute nodes of `A` and `B` as the variables within the expression are bound to their attributes. One can now say that the attributes of `C` are dependent on the attributes of `A` and `B`. If done for all rules a complete graph of dependencies can be created as illustrated in Figure 4.

If we now wanted to calculate all possible values we would first start by populating the initial values for each attribute given in the initial solution to the attribute nodes of the graph. Then we look for sinks in the graph. Sinks are those attribute nodes that can be safely calculated, which means, all of the attribute nodes they depend on have already been calculated. Attribute nodes without outgoing edges are considered to be calculated as there are no expressions changing their values. After a sink is found we use the connected

expression nodes to calculate new values. Therefore, each variable is replaced by a value from the attribute node it refers to and the expression is evaluated. This is done for every possible combination of available values until all attribute nodes have been calculated. Figure 4 also shows the calculation steps needed for this particular example.

A problem may arise if there are cycles within the graph because then there are occasions where no sink can be found. One can distinguish two types of cycles: bound cycles and unbound cycles. Bound cycles are the ones were the calculation of an attribute depends on the attribute itself but the set of possible value can not rise infinitely. Imagine for example the following rule:

```
A(x) -> A(x+1) @ if (x > 10) then 0 else k
```

The value assignment for the attribute of `A` depends on its own value and will increase by one for each calculation step. However when the value bound to the variable `x` is greater than 10, the rate of this rule would become zero, which means it can not be executed. This restricts the values that can be inserted into the equation and determines an upper bound for the cycle. So this cycle can be calculated as there will be a time were no new values are produced. In addition, one can easily image the unbound version of this expression if the given condition is eliminated. Then the value of species `A` may rise infinitely in theory. However, the modeler may have further knowledge of the restrictions of the model. So an approach for future work might be to include user interaction into the process to guide the transformation and provide specific information that are hard or impossible to determine automatically.

To assure termination in any case, we implemented an upper bound for cycle runs. If there are still cycles that can be executed after reaching this maximum, the user will be warned that the transformation might not be complete.

4.2 Hierarchical models

Another major challenge during transformation is the handling of hierarchies within an ML-Rules model. Consider the following excerpt consisting of an initial solution and two rule schemata:

```
>>INIT[D[2 A] + D[A + B]]
D[A + s?] -> D[s?]
D[B + s?] -> D[A + s?]
```

The initial solution consists of two nested species of type `D`, i.e., `D[2 A]` and `D[A + B]`. In the first rule, a degradation of species `A` residing within a species `D` is described. The second rule transforms a species `B` into a species `A`, which also takes place within a species `D`. Although all species are attribute-less, a transformation of this model becomes problematic due to the nesting of `A` and `B` within `D`. This is because in ML-Rules species are not only distinguished by their attributes but also by their content and context, i.e., the species they enclose and the species they are enclosed by. For example, applying the second rule of the above sample model to the species `D[A + B]` yields another species of kind `D[2 A]`, i.e., the amount of `D[2 A]` would be increased from 1 to 2. On the other hand, applying the first rule to `D[A + B]` would yield `D[B]`, i.e., a new species not appeared so far. Thus, in the general case, we have to encode each possible hierarchy of species as an individual SBML species. This quickly leads to combinatorial explosion, particularly when species with attributes are involved. Moreover, it is possible that the number of hierarchy levels in the model is dynamic itself, e.g., consider repeatedly applying a rule like `A[] -> A[A[]]`. Thus, we encounter the same fundamental problem as with transforming flat models. So, to still assure termination, we used the same approach as for flat models by restricting the nesting depth.

5 VALIDATION

To validate our approach we implemented a prototype within the plug-in-based modeling and simulation framework JAMES II (Himmelspach and Uhrmacher 2007). We followed the JAMES II philosophy and designed the transformation mechanism to be as flexible and reusable as possible. A transformation is therefore split into several passes, where each pass performs a specific job within the overall transformation

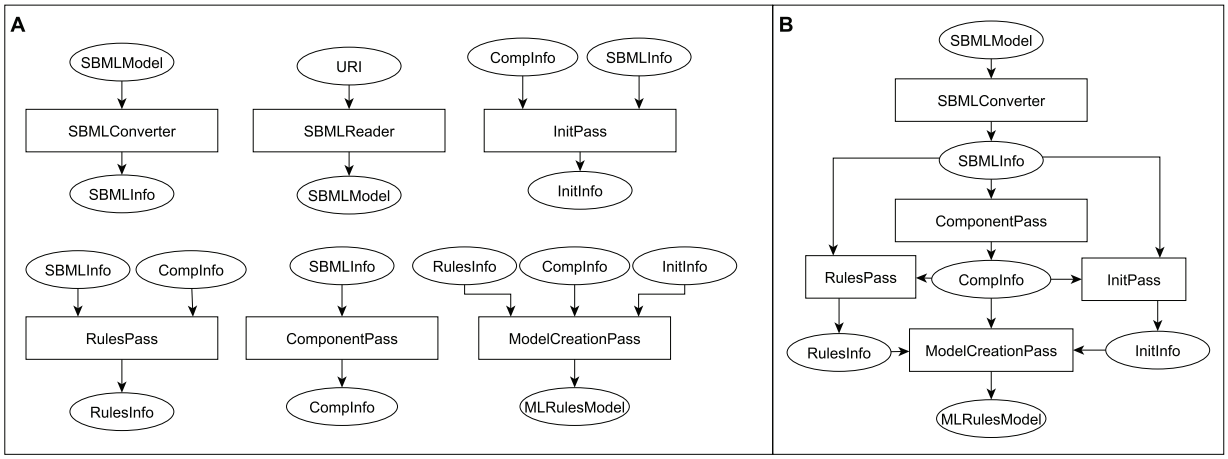

Figure 5: On-demand composition of passes. A transformation consists of several passes that can be composed to a complete process. Every pass defines what type of information it needs to run and what kind of information it produces (as seen in A). A composition of several passes is shown in B, based on the transformation of an SBML model into the ML-Rules modeling language.

process. The passes are working directly on the tree structure representing a model specified by a certain formalism, in this case SBML or ML-Rules, which is similar to abstract syntax trees. The main idea is that passes are independent of each other and can be composed on demand by analyzing the information they provide or require. Some of the passes can thus be reused to support other kind of transformations, such as a transformation from SBML to simple chemical reaction networks (Gillespie 1977) or to ordinary differential equations, which are implemented in JAMES II as well. A simple illustration of the passes involved in transforming models from SBML to ML-Rules is shown in Figure 5.

The principal idea for validating our model transformation is to execute both the original and the transformed version of a model, followed by a comparison of their results. We limited our comparison to species amounts at the end of the simulation, rather than checking whole trajectories. This allows us to work with relatively small sample sizes and thus facilitates the analysis. Besides numerical analysis, we also face-validated many smaller models, i.e., the transformations were also checked manually.

5.1 SBML to ML-Rules

Since ML-Rules employs a stochastic execution semantics, we used the *Discrete Stochastic Models Test Suite* (DSMTS, Evans et al. 2008) to test the transformation from SBML to ML-Rules. This test suite provides simple SBML level 3 models covering most of SBML's features. For each model and species, the expected mean value and standard deviation are given. We followed the test setup suggested by the DSMTS user guide (edition for SBML L3v1, https://code.google.com/p/dsmts) and calculated the distance of the empirical means and standard deviations (of species amounts at the end of the simulation) to the true mean and standard deviation, in terms of Z scores. This has been done for a subset of DSMTS models, which were transformed to ML-Rules and then replicated 10,000 times. The results are shown in Figure 6. While some tests indicate that the transformation is invalid, e.g., for `001-01(X)`, these cases have been checked individually and we found the culprit to be the observation mechanism of the simulator (currently undergoing a revision), not the transformation itself (the problem is that currently the state *after* a certain point in time is observed, but it is compared with the true state at the exact given time point).

Figure 6: Validation of the SBML to ML-Rules transformation. The result of the mean (green) and stand deviation (blue) test from the Discrete Stochastic Model Test Suite is shown, as Z scores of the empirical mean and standard deviation. Names on the x-axis refer to the models used where the affected species name is given in parentheses. The interval in which the test is considered to be valid is represented by red lines.

5.2 ML-Rules to SBML

We used a slightly different approach to validate the transformation from ML-Rules to SBML models. We constructed ML-Rules test models, transformed them into SBML, and then transformed them back into ML-Rules again. With this approach we could rely on the ML-Rules simulator for both the original and the (twice-)transformed versions, thus avoiding any bias due to the way simulation output was observed. Another advantage is that this approach validates both transformations at the same time. The test models cover the critical features discussed in Section 4. They include rule schemata with cyclic attribute definitions and with constrained rates, so that the enumeration of species of a flat model could be validated. Another set of rule schemata worked with nestedness, by allowing a species to enter or exit some other species. They allowed us to validate the transformation of nested species (the full model is omitted here for space limitations). After transformation, a single test model consists of 53 species and 99 reactions.

We simulated both the original and the twice-transformed models for 50 time units, executed 10,000 replications for each, and saved the outcomes for all species at the end of each simulation run, leading to sets of 10,000 sample values per model and species. For each model we randomly selected 20 species and compared these empirical distributions from the original and twice-transformed model by using the Wilcoxon rank-sum test (e.g., Sheskin 2007, p. 513). With $\alpha = 0.05$, our null hypotheses (that given empirical distributions are equal) could not be rejected for any of the 80 tests we carried out.

Finally, we also validated the transformation mechanism by transforming the ML-Rules models first to SBML and then to chemical reaction networks, which can be simulated by separate stochastic simulation algorithms (SSAs, Gillespie 1977) provided by JAMES II. We use the same test setup as before and again applied the Wilcoxon rank-sum test. We obtained a similar result in that no null hypotheses could be rejected for any of our 80 tests.

6 RELATED WORK

Instead of general transformation of modeling formalisms, like efforts as AToM3 (Vangheluwe and de Lara 2003), we have focused on the concrete transformation between a standard exchange format and a modeling language, i.e., a direct manipulation (Czarnecki and Helsen 2006). Future work could instead rely on toolkits like the ATLAS model management architecture (Bézivin, Jouault, and Touzet 2005), which provides a custom language to specify model transformations and is based on the Eclipse Modeling Framework (EMF). It would be also possible to use graph transformation approaches like GReAT (Agrawal, Karsai, and Shi

2003) or MOLA (Kalnins, Barzdins, and Celms 2005) which define rule-patterns for components of the meta-model of a language. These pattern then can be applied to specific components of the meta-model graph and transform them into other components, thus rewriting the meta-model graph. This is similar to using rule-based modeling languages like ML-Rules or BNGL itself. Using one of this approaches would first require to define a meta-model for ML-Rules as this has not been done yet. A meta-model for SBML already exists in its specification (Hucka et al. 2010). However, none of these approaches resolves the challenging issues we have outlined in Sections 3 and 4, as these stem from conceptual differences between SBML and ML-Rules.

A particular challenge has been the state space exploration, as in ML-Rules theoretically an infinite number of species and reactions are supported, whereas in SBML all species and reactions have to be listed explicitly. This is a comparatively old problem in computer science (see, e.g., de Souza e Silva and Ochoa 1992). During the last years in addition to rule-based languages, a series of colored modeling languages, e.g., Colored Petri Nets (Gao et al. 2011) or Attributed Process Calculi (John et al. 2010), have been introduced to support the modeling efforts in computational biology. However, concrete solutions to generate from these models SBML compliant models are still rather rare. BioNetGen offers such a concrete solution (Faeder et al. 2009). To transform the rule-based models of BioNetGen into SBML format, simulation runs are executed. Species and reactions that are generated up to a certain limit during this run enter the SBML specification. Thus, the approach in BioNetGen is experimental in nature, whereas in our approach the model structure is analyzed independently of any simulation run. The approach in BioNetGen constrains the general number of species. We support more species-specific constraints, e.g., that the attribute values of a species has to be in a specific interval or the maximum depth of nested species.

Different extensions of the core SBML standard have been proposed, e.g., the recently released Hierarchical Model Composition package specification (http://sbml.org/specifications/sbml-level-3/version-1/comp/sbml-comp-version-1-release-2.pdf). The focus of this extension is the reuse and composition of existing models rather than describing systems whose dynamics depend on nested structures. However, an initiative to extend SBML for dynamically nested structures has been announced for the future, emphasizing the importance of dealing with dynamically nested structures.

Currently each reaction from SBML is transformed into a rule in ML-Rules. This approach does not take one of the advantages of rule-based modeling languages into account, using schemata to group similar reaction into only one rule definition. Future work might extend the current transformation to search for suitable rule schemata in the given reaction network.

7 CONCLUSION

In this paper, we explore the difficulties that arise when transforming between a state-of-the-art modeling language with advanced features (here, ML-Rules) and a standard model exchange format (here, SBML). While some model entities can be transformed easily (e.g., parameters), many others require special handling and cannot be mapped one-to-one (see Sections 3 and 4). Such mismatches can often be addressed automatically, so that a modeler does not need to bother with the details of the transformation. To facilitate the development of additional transformation mechanisms, we implemented an extensible transformation mechanism currently focused on SBML import and export. It has been validated with custom test models and models from the Discrete Stochastic Model Test Suite (see Section 5).

We found that the representation of ML-Rules' attributed species and dynamic hierarchies in SBML are hard problems. So far, we only solved them with rather simple heuristics (see Section 4). Future work should therefore tackle these problems with more advanced techniques, e.g., by automatically identifying such species that could be represented by a compartment component in SBML. For transforming SBML into ML-Rules methods relying on semantic annotations (Courtot et al. 2011) might help to reduce the size of transformed models, e.g., by identifying species with different states and/or localities and representing them in ML-Rules by attributed species within a nested hierarchy. We also found that the omission of

time-triggered events is problematic (see Section 3). Therefore, efforts to support such events in ML-Rules are under way.

REFERENCES

Agrawal, A., G. Karsai, and F. Shi. 2003. "Graph Transformations on Domain-Specific Models". Technical Report ISIS-03-403, Vanderbilt University.

Bézivin, J., F. Jouault, and D. Touzet. 2005. "An introduction to the ATLAS Model Management Architecture". Technical Report 05.01, Université de Nantes.

Bittig, A. T., and A. M. Uhrmacher. 2010. "Spatial modeling in cell biology at multiple levels". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 608–619. Piscataway, New Jersey: IEEE.

Courtot, M., N. Juty, C. Knüpfer, D. Waltemath, A. Zhukova, A. Dräger, M. Dumontier, A. Finney, M. Golebiewski, J. Hastings et al. 2011. "Controlled vocabularies and semantics in systems biology". *Molecular Systems Biology* 7:543.

Cuellar, A. A., C. M. Lloyd, P. F. Nielsen, D. P. Bullivant, D. P. Nickerson, and P. J. Hunter. 2003, December. "An Overview of CellML 1.1, a Biological Model Description Language". *SIMULATION* 79 (12): 740–747.

Czarnecki, K., and S. Helsen. 2006, July. "Feature-based survey of model transformation approaches". *IBM Systems Journal* 45 (3): 621–645.

de Jong, H. 2002. "Modeling and Simulation of Genetic Regulatory Systems: A Literature Review". *Journal of Computational Biology* 9 (1): 67–103.

de Souza e Silva, E., and P. M. Ochoa. 1992, June. "State space exploration in Markov models". *SIGMETRICS Performance Evaluation Review* 20 (1): 152–166.

Evans, T. W., C. S. Gillespie, and D. J. Wilkinson. 2008, January. "The SBML discrete stochastic models test suite". *Bioinformatics* 24 (2): 285–286.

Faeder, J. 2011. "Toward a comprehensive language for biological systems". *BMC 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*, edited by I. V. Maly, Volume 500 of *Methods in Molecular Biology*, 113–167. Humana Press.

Gao, Q., F. Liu, D. Gilbert, M. Heiner, and D. Tree. 2011. "A Multiscale Approach to Modelling Planar Cell Polarity in Drosophila Wing using Hierarchically Coloured Petri Nets". In *International Conference on Computational Methods in Systems Biology (CMSB)*, edited by F. Fages, 209–218.

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

Himmelspach, J., and A. M. Uhrmacher. 2007. "Plug'n simulate". In *Proceedings of the 40th Annual Simulation Symposium*, 137–143: IEEE Computer Society.

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

John, M., C. Lhoussaine, J. Niehren, and A. Uhrmacher. 2010. "The Attributed Pi-Calculus with Priorities". *Transactions on Computational Systems Biology* 5945:13–76.

Kalnins, A., J. Barzdins, and E. Celms. 2005. "Model Transformation Language MOLA". In *Model Driven Architecture*, edited by U. Aßmann, M. Aksit, and A. Rensink, Volume 3599 of *LNCS*, 62–76. Springer.

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.

Mattson, S., and H. Elmquist. 1998. "An Overview of thte Modeling Language Modelica". In *Eurosim'98 Simulation Congress*. SIMS.

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

Oury, N., and G. D. Plotkin. 2011. "Coloured stochastic multilevel multiset rewriting". In *International Conference on Computational Methods in Systems Biology (CMSB)*, edited by F. Fages, 171–181: ACM.

Sheskin, D. 2007, January. *Handbook of Parametric and Nonparametric Statistical Procedures, Fourth Edition*. Chapman & Hall/CRC.

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, New Jersey: IEEE.

Uhrmacher, A. 1995. "Reasoning about changing structure: a modeling concept for ecological systems". *Applied artificial intelligence* 9 (2): 157–180.

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, New Jersey: IEEE.

Vangheluwe, H., and J. de Lara. 2003. "Foundations of multi-paradigm modeling and simulation: computer automated multi-paradigm modelling: meta-modelling and graph transformation". In *Proceedings of the 2003 Winter Simulation Conference*, edited by S. E. Chick, P. J. Sanchez, D. M. Ferrin, and D. J. Morrice, 595–603: ACM.

Wilkinson, D. J. 2006, April. *Stochastic Modelling for Systems Biology*. Mathematical and Computational Biology. Chapman & Hall/CRC.

AUTHOR BIOGRAPHIES

SEBASTIAN NÄHRING holds a BSc. in Computer Science from the University of Rostock. Currently, he is pursing a Master's degree in Computer Science at the University of Rostock. His email address is sebastian.naehring@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.

ROLAND EWALD holds a PhD in Computer Science from the University of Rostock. Currently, he is a post-doc at the Institute of Computer Science of the University of Rostock. His email address is roland.ewald@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.