Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 293 (2013) 51–65

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Software Interface Between the Narrative Language and Bio-PEPA

Anastasis Georgoulas[1]

SynthSys – Synthetic and Systems Biology, University of Edinburgh, Edinburgh, United Kingdom

Maria Luisa Guerriero[2]

Sytems Biology Ireland, University College Dublin, Dublin, Ireland

Abstract

We present a software tool for the automatic translation of models from the Narrative Language, a semi-formal language for biological modelling, into the Bio-PEPA process algebra. This provides biologists with an easy way to describe systems and at the same time gives them access to the simulation and analysis techniques provided by Bio-PEPA. We present details of the translation algorithm and its integration into existing software, and discuss ways in which this idea could be further explored.

Keywords: automatic translation, high level modelling languages, process algebra, Narrative Language, Bio-PEPA

1 Introduction

The use of formal methods for modelling the behaviour of biological systems has been a subject of intense research in recent years. In particular, process algebras [6] have been successfully shown to be a useful formalism for the description of biological processes and their analysis, using established techniques such as stochastic simulation [11] and model-checking [14].

Mathematical formalisms such as process algebras are not an easy modelling language to adopt for non-experts, and in this respect they are in stark contrast to the textual and graphical descriptions traditionally used in biology, which are often informal, ambiguous and not amenable to automatic manipulation, but intuitive and easy to use.

[1] Email: anastasis.georgoulas@ed.ac.uk

[2] Email: maria.guerriero@ucd.ie

doi:10.1016/j.entcs.2013.02.018

Biological modelling, especially in the context of systems biology, is increasingly becoming an interdisciplinary field, and often biologists are directly involved in the development of models. However, because of the complexity of formal methods, despite their attractive characteristics and the analysis capabilities, biologists often prefer to continue describing systems in an informal way and rely on mathematicians and computer scientists to translate these informal descriptions into computational models. Unfortunately, this translation process is prone to a number of errors, including misinterpretation of the informal representations.

In order for scientists with different backgrounds to collaborate efficiently and to fully take advantage of existing and emerging techniques, they need a common vocabulary and modelling language and tools that are accessible to all of them. An ideal modelling platform would be one that combines an interface that is easy to use by wet-lab biologists with the strong analysis capabilities of formal languages. Such platform would hide from the user the process of translating the biologists' intuitive system descriptions into formal computational models.

In this paper we present a tool for translating a semi-formal modelling language for biology [12] into the process algebra Bio-PEPA [7]. Our tool is implemented as an extension of the Bio-PEPA Eclipse plugin [5], an existing software platform for formal modelling and analysis of biological systems.

The remainder of the paper is structured as follows: Section 2 gives some background information on the two modelling languages involved in the translation we present; Section 3 describes the translation procedure, while Section 4 contains an example of its application; details of the implementation of the tool and its integration with existing software are given in Section 5; finally, Section 6 presents ideas for further work.

2 Background

The ideas laid out in the previous section were the reasoning behind the introduction of the Narrative Language (NL) [13]. The NL describes the events that can occur in a system using a syntax that, although constrained, approximates natural language descriptions. It also provides a predefined vocabulary of biochemical terms (such as "binds" or "phosphorylates") that are similar to those normally used by biologists and allow one to model a variety of biological interactions. As demonstrated in [12], the model essentially takes the form of a few tables, each containing information about the locations, species and reactions of the system. In the original work, this semi-formal description was translated into the Beta-Binders process algebra [15]. One can thus enjoy the benefits of formal modelling without having to express the model in a language that is potentially not approachable by biologists.

Bio-PEPA [7] is a stochastic process algebra introduced for the purpose of being applied to biochemical systems. It adopts a reagent-centric view in which each biochemical species is abstracted as a process. A model is then composed in a modular way through the interactions between the processes.

Although a translation to Beta-Binders already exists, the translation to Bio-

PEPA is worthwhile for a number of reasons. Firstly, having another target language was a motivation to re-evaluate and extend the syntax of the NL (as described in the next section). Additionally, the new translation has been integrated into an existing actively-maintained software platform, the Bio-PEPA Eclipse plugin, which provides modellers with a number of interesting simulation and analysis methods. For example, one can perform static analysis of a model's invariants or export it to a format suitable for model-checking.

3 Translation

Because of the various conceptual differences between Beta-Binders and Bio-PEPA, the algorithm we propose is not a modification on the original translation to Beta-Binders. Rather, it is designed with Bio-PEPA in mind as the target language and tailored to the specific features of that formalism.

3.1 Input language

The fact that the NL was designed having in mind that it would be translated to Beta-Binders influenced its original syntax and characteristics. Since our target language is different, we took this opportunity to introduce some new language features that expand the expressive capabilities of the NL, which were not supported by Beta-Binders.

One of the features of Bio-PEPA is that it supports the use of arbitrary kinetic laws, whereas in Beta-Binders (and, accordingly, in the initial version of the NL), only mass-action kinetics were supported. In the input language for our translation, a reaction can be defined as occurring at a constant rate or following mass-action, Michaelis-Menten or Hill kinetics, with a corresponding set of parameters.

An additional change is the ability to define constants, which can then be used in the model whenever an explicit numerical value is expected, e.g. as kinetic parameters or initial concentrations. Despite being a trivial change, this greatly simplifies the process of specifying and modifying models.

A description of the full language syntax can be found in Appendix C.

3.2 Preprocessing

Before the translation itself begins, a series of checks are performed to ascertain whether the model is valid. These consist mainly in "sanity checks" in order to ensure the internal consistency of the model, i.e. that the definitions of its elements are not contradicted by their subsequent use. For example, if an event states "*if A is active then A relocates to 2*", then the definition of component A must state at least two things: that A can exist in an active or inactive state; and that A is (potentially) found in compartment *2*.

3.3 The algorithm

Generally speaking, the NL follows a rule-based modelling style, so the emphasis is on the description of events. On the other hand, Bio-PEPA is reagent-centric, with the building block being the definitions of species. Therefore our work is mainly to process each NL event to collect the reactions in which each species participates and its role in them. We will now briefly describe how each event is processed.

Identifying the participating components

The first step is to find the components that take part in the event, which can happen in two ways: a component can be either *affected* if it undergoes some change as a result of the event, or simply *involved* if it facilitates such a change without changing itself. We make this distinction to differentiate between the roles a species can have in a reaction in Bio-PEPA: affected components correspond to reactants and products, while involved ones to modifiers (enzymes or inhibitors).

Consider for example the following event: "*if A is inactive then B activates A*". In this bimolecular reaction, A undergoes a change (i.e. becomes active) and is therefore an affected component, while B does not undergo any change and is therefore an involved component.

Getting the variants of the components

A crucial difference between the NL and Bio-PEPA is that in the NL a component is associated with a number of binary states (e.g. (un)phosphorylated, (in)active), whereas Bio-PEPA species have no internal state. This means that a NL component with n states can correspond, in the translation, to 2^n Bio-PEPA species. In practice, the number of these variants is constrained by the conditions of each event, which can impose restrictions on the state of components. We will denote by $var(a, C)$ the set of variants of component a that satisfy the set of conditions C.

Continuing the previous example, let us assume that B has no internal states while A has two: it can be phosphorylated or not, and active or not. This implies that B can only have one variant, also denoted B. A can have four variants, denoted $A\_active\_phosphorylated$, $A\_active\_unphosphorylated$, $A\_inactive\_phosphorylated$ and $A\_inactive\_unphosphorylated$. However, the event imposes a condition c_1 on A, which limits the applicable variants to two; i.e. formally, we have that this event can only be applied to the set $var(A, \{c_1\}) = \{A\_inactive\_phosphorylated, A\_inactive\_unphosphorylated\}$.

Adding the reactions

Each combination of variants of the participating components can give rise to a different reaction. Let $A = \{a_k\}$ and $I = \{i_k\}$ be the set of affected and involved components, respectively. Then the output model will contain one reaction for each element of R, where

$$R = \prod\{var(a, C)|a \in A\} \times \prod\{var(i, C)|i \in I\} \ .$$

The products of every such reaction depend on the affected variants and the type of the event (e.g. activation, binding, etc.). Let the product species be denoted by $out(A', e)$, where e is a type of event and A' is a tuple of variants of the affected species, i.e.

$$A' \in \prod\{var(a, C)|a \in A\} \ .$$

In summary, for every element of R, we add one Bio-PEPA reaction for which the reactants are A', the activators are I' (defined similarly to A') and the products are $out(A', e)$. It is obvious that the resulting model will in general have more species and reactions than the number of NL components and events, respectively. For a single event, in the worst case the increase could be exponential in the number of states of the participating components. This is an unavoidable consequence of the nature of Bio-PEPA, specifically its lack of internal state for species, as mentioned above.

In the example, there are two combinations between $var(A, \{c_1\})$ and $var(B, \{c_1\})$, so

$$R = \{(A\_inactive\_phosphorylated, B), (A\_inactive\_unphosphorylated, B)\} \ .$$

This event will therefore be translated into two reactions, $r1$ and $r2$, one for each combination. For $r1$,

$$A' = A\_inactive\_phosphorylated,\ I' = B$$

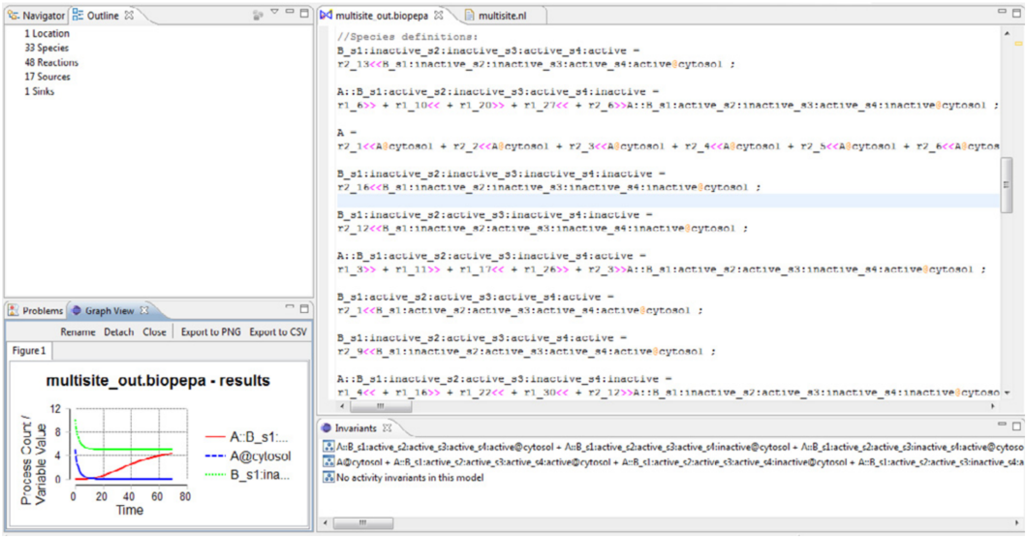
and $out(A', activation) = \{A\_active\_phosphorylated\}$. Reaction $r2$ is defined similarly.

After the event is translated, we will have the following roles for the relevant species.

$$\begin{aligned} A\_inactive\_phosphorylated &= r1 \downarrow \\ A\_active\_phosphorylated &= r1 \uparrow \\ A\_inactive\_unphosphorylated &= r2 \downarrow \\ A\_active\_unphosphorylated &= r2 \uparrow \\ B &= r1 \oplus \ + r2 \oplus \end{aligned}$$

4 Example and effect on model size

As a further simple example, we consider a molecule B with four activation sites which can be activated in any order. Another molecule A can bind to B regardless of B's activation state. The description of this system in the NL contains two components and five reactions (four activations and one binding). The full model can be found in Appendix A.



```
//Species definitions:
B_s1:inactive_s2:inactive_s3:active_s4:active =
r2_13<<B_s1:inactive_s2:inactive_s3:active_s4:active@cytosol ;

A::B_s1:active_s2:inactive_s3:active_s4:inactive =
r1_6>> + r1_10<< + r1_20>> + r1_27<< + r2_6>>A::B_s1:active_s2:inactive_s3:active_s4:inactive@cytosol ;

A =
r2_1<<A@cytosol + r2_2<<A@cytosol + r2_3<<A@cytosol + r2_4<<A@cytosol + r2_5<<A@cytosol + r2_6<<A@cytos

B_s1:inactive_s2:inactive_s3:inactive_s4:inactive =
r2_16<<B_s1:inactive_s2:inactive_s3:inactive_s4:inactive@cytosol ;

B_s1:inactive_s2:active_s3:inactive_s4:inactive =
r2_12<<B_s1:inactive_s2:active_s3:inactive_s4:inactive@cytosol ;

A::B_s1:active_s2:active_s3:inactive_s4:active =
r1_3>> + r1_11>> + r1_17<< + r1_26>> + r2_3>>A::B_s1:active_s2:active_s3:inactive_s4:active@cytosol ;

B_s1:active_s2:active_s3:active_s4:active =
r2_1<<B_s1:active_s2:active_s3:active_s4:active@cytosol ;

B_s1:inactive_s2:active_s3:active_s4:active =
r2_9<<B_s1:inactive_s2:active_s3:active_s4:active@cytosol ;

A::B_s1:inactive_s2:active_s3:inactive_s4:inactive =
r1_4<< + r1_16>> + r1_22<< + r1_30<< + r2_12>>A::B_s1:inactive_s2:active_s3:inactive_s4:inactive@cytoso
```

```
A::B_s1:active_s2:active_s3:active_s4:active@cytosol + A::B_s1:active_s2:active_s3:active_s4:inactive@cytosol + A::B_s1:active_s2:active_s3:inactive_s4:active@cytoso
A@cytosol + A::B_s1:active_s2:active_s3:active_s4:active@cytosol + A::B_s1:active_s2:active_s3:active_s4:inactive@cytosol + A::B_s1:active_s2:active_s3:inactive_s4:a
No activity invariants in this model
```

Fig. 1. The translated model opened in the Bio-PEPA plugin with some analysis results.

Appendix B contains the result of the translation, and Figure 1 shows the file as displayed in the Bio-PEPA plugin. The resulting model has 33 species and 48 reactions, in contrast to the compact size of the original one. This increase in size is a consequence of the inability to express internal states in Bio-PEPA: the complexity held by the existence of sites and states in the NL model leads directly to this combinatorial explosion during the translation. In the worst case, the number of Bio-PEPA species corresponding to a NL component can be exponential in the number of the component's sites, with the number of reactions growing accordingly. This example shows another reason why developing models in the NL and automatically translating them into Bio-PEPA can be advantageous compared to developing them in Bio-PEPA directly.

While this increase in model complexity is inevitable in the translation, there may be ways of alleviating its effect by exploiting the structure of the system. For instance, some combinations of states might be implicitly prohibited by the sequence of events or the initial state, and we can thus remove the corresponding species from the final model through a reachability analysis. Additionally, we can lump together species with identical behaviour; for instance, in the above example, if all activation sites are indistinguishable, we can treat all variants that have exactly one active site as equivalent, thus reducing the number of resulting species.

5 Implementation and software integration

Our translation procedure has been implemented as an extension of the existing Bio-PEPA plugin [5] for the Eclipse IDE [1] (the code can be found at [3] and will be included in the next major release of the Bio-PEPA plugin). Specifically, we have added a new menu item that prompts the user to select an input file and an output location and then performs the translation. If the translation is completed successfully without any errors, the user then has the choice to open the newly created Bio-PEPA file in the editor, from where they can use the analysis methods available for any standard Bio-PEPA file.

In the case of errors being found during the validation procedure, the course of action depends on their severity. If the translation cannot continue, e.g. if the parsing has failed because an essential element is missing, the process is aborted. If, however, the problems are less severe, the user has the choice of continuing, although the resulting file may not be valid in its entirety.

The tool is written in Java, using packages from the standard library, with two exceptions. Parsing of the model is performed using Xtext [4], a tool for specifying and handling domain-specific languages. To use Xtext, we specify the rules describing the NL grammar in a format similar to EBNF. Xtext then produces an object-oriented model of the grammar, automatically generating classes for each term defined, and also provides classes for parsing a file according to the grammar. This greatly simplifies the low-level work needed to retrieve the model described. Secondly, the graphical interface is built using the Eclipse API [2] for its various elements.

6 Future work

We are interested in continuing this work and believe there is room for further improvement, at both a theoretical and practical level. For the former, we plan to explore possible optimisations of the translation algorithm that would result in a reduced output model. This could be done by analysing the Bio-PEPA model once it is generated and using existing theoretical results concerning bisimilarity in the language, e.g. [10], or approaches like symmetry detection [9]. Translations could also be developed for other target languages, in particular rule-based languages such as Kappa [8].

On the practical side, we believe that our tool would benefit from a graphical interface that makes the NL even easier to access. One idea, for instance, would be to use Xtext to implement a text editor for the Narrative Language, which could be integrated into the Bio-PEPA plugin. Another option would be to have a more elaborate and user-friendly GUI for specifying NL models, using forms or a spreadsheet-like application. This could be either standalone or part of the plugin. One could even hide the Bio-PEPA model altogether: a user could describe the model and choose an analysis method using a purpose-built GUI, then the tool would silently perform the translation and run the desired analysis, before presenting the user with the results.

Acknowledgement

The authors would like to thank Jane Hillston, Stephen Gilmore and Allan Clark for their help and advice. SynthSys (formerly "Centre for Systems Biology at Edinburgh") is a Centre for Synthetic and Systems Biology funded by BBSRC and

EPSRC, ref. BB/D019621/1.

References

[1] *Bio-PEPA*, http://www.biopepa.org/.

[2] *Eclipse Platform API Specification*, http://help.eclipse.org/helios/topic/org.eclipse.platform.doc.isv/reference/api/overview-summary.html.

[3] *GitHub*, https://github.com/ageorgou/Bio-PEPA.

[4] *Xtext*, http://www.eclipse.org/Xtext/.

[5] Ciocchetta, F., A. Duguid, S. Gilmore, M. L. Guerriero and J. Hillston, *The Bio-PEPA Tool Suite*, in: *Sixth International Conference on the Quantitative Evaluation of Systems, QEST'09*, 2009, pp. 309–310.

[6] Ciocchetta, F. and J. Hillston, *Process Algebras in Systems Biology*, in: *SFM'08*, Lecture Notes in Computer Science **5016**, Springer-Verlag, 2008 pp. 265–312.

[7] Ciocchetta, F. and J. Hillston, *Bio-PEPA: A framework for the modelling and analysis of biological systems*, Theoretical Computer Science **410** (2009), pp. 3065–3084.

[8] Danos, V., J. Feret, W. Fontana, R. Harmer and J. Krivine, *Rule-Based Modelling of Cellular Signalling*, in: *CONCUR 2007 - Concurrency Theory*, Lecture Notes in Computer Science **4703**, Springer Berlin/Heidelberg, 2007 pp. 17–41.

[9] Donaldson, A. and A. Miller, *Automatic symmetry detection for model checking using computational group theory*, in: *FM 2005: Formal Methods*, Lecture Notes in Computer Science **3582**, Springer Berlin/Heidelberg, 2005 .

[10] Galpin, V., *Equivalences for a biological process algebra*, Theoretical Computer Science **412** (2011), pp. 6058–6082.

[11] Gillespie, D. T., *Exact stochastic simulation of coupled chemical reactions*, Journal of Physical Chemistry **81** (1977), pp. 2340–2361.

[12] Guerriero, M. L., A. Dudka, N. Underhill-Day, J. K. Heath and C. Priami, *Narrative-based computational modelling of the Gp130/JAK/STAT signalling pathway*, BMC Systems Biology **3** (2009), p. 40.

[13] Guerriero, M. L., J. K. Heath and C. Priami, *An Automated Translation from a Narrative Language for Biological Modelling into Process Algebra*, in: *Proceedings of Computational Methods in Systems Biology (CMSB'07)*, Lecture Notes in Computer Science **4695** (2007), pp. 136–151.

[14] Kwiatkowska, M., G. Norman and D. Parker, *Using probabilistic model checking in systems biology*, ACM SIGMETRICS Performance Evaluation Review **35** (2008), pp. 14–21.

[15] Priami, C. and P. Quaglia, *Operational patterns in Beta-binders*, Transactions on Computational Systems Biology **1** (2005), pp. 50–65.

A NL model of the example system

Constants
(N,5)

Compartments
(1, cytosol, 1.0, , 3,)

Components
(1, A, , , bound:FALSE, 1:(100, 100), (N, 100), (0,0), (0,0))
(2, B, , s1:active:FALSE;s2:active:FALSE;s3:active:FALSE;s4:active:FALSE, bound:FALSE,1:(100, 100), (10, 100), (0,0), (0,0))

Reactions
(1, activation, “single site activation”, (fMA(0.05), 50), (1.0, 50))
(2, binding, “binding A-B”, (fMA(0.05),100),(1.0,100))

Narrative

```
Process "activation of B"
(1, if A is bound and B is bound and B.s1 is not active then A activates B on s1, "", 1, , )
(2, if A is bound and B is bound and B.s2 is not active then A activates B on s2, "", 1, , )
(3, if A is bound and B is bound and B.s3 is not active then A activates B on s3, "", 1, , )
(4, if A is bound and B is bound and B.s4 is not active then A activates B on s4, "", 1, , )
(5, if A is not bound and B is not bound then A binds B, "",2, ,)
```

B Bio-PEPA translation of example system

```
//Constants:
N = 5;
//Compartments:
location cytosol : size = 1.0, type = compartment;
//Kinetic rate laws:
kineticLawOf r1_1 : fMA(0.05);
kineticLawOf r1_2 : fMA(0.05);
kineticLawOf r1_3 : fMA(0.05);
kineticLawOf r1_4 : fMA(0.05);
kineticLawOf r1_5 : fMA(0.05);
kineticLawOf r1_6 : fMA(0.05);
kineticLawOf r1_7 : fMA(0.05);
kineticLawOf r1_8 : fMA(0.05);
kineticLawOf r1_9 : fMA(0.05);
kineticLawOf r1_10 : fMA(0.05);
kineticLawOf r1_11 : fMA(0.05);
kineticLawOf r1_12 : fMA(0.05);
kineticLawOf r1_13 : fMA(0.05);
kineticLawOf r1_14 : fMA(0.05);
kineticLawOf r1_15 : fMA(0.05);
kineticLawOf r1_16 : fMA(0.05);
kineticLawOf r1_17 : fMA(0.05);
kineticLawOf r1_18 : fMA(0.05);
kineticLawOf r1_19 : fMA(0.05);
kineticLawOf r1_20 : fMA(0.05);
kineticLawOf r1_21 : fMA(0.05);
kineticLawOf r1_22 : fMA(0.05);
kineticLawOf r1_23 : fMA(0.05);
kineticLawOf r1_24 : fMA(0.05);
kineticLawOf r1_25 : fMA(0.05);
kineticLawOf r1_26 : fMA(0.05);
kineticLawOf r1_27 : fMA(0.05);
kineticLawOf r1_28 : fMA(0.05);
kineticLawOf r1_29 : fMA(0.05);
kineticLawOf r1_30 : fMA(0.05);
kineticLawOf r1_31 : fMA(0.05);
kineticLawOf r1_32 : fMA(0.05);
kineticLawOf r2_1 : fMA(0.05);
kineticLawOf r2_2 : fMA(0.05);
kineticLawOf r2_3 : fMA(0.05);
kineticLawOf r2_4 : fMA(0.05);
kineticLawOf r2_5 : fMA(0.05);
kineticLawOf r2_6 : fMA(0.05);
kineticLawOf r2_7 : fMA(0.05);
kineticLawOf r2_8 : fMA(0.05);
kineticLawOf r2_9 : fMA(0.05);
kineticLawOf r2_10 : fMA(0.05);
kineticLawOf r2_11 : fMA(0.05);
kineticLawOf r2_12 : fMA(0.05);
kineticLawOf r2_13 : fMA(0.05);
kineticLawOf r2_14 : fMA(0.05);
kineticLawOf r2_15 : fMA(0.05);
kineticLawOf r2_16 : fMA(0.05);
//Species definitions:
B_s1:inactive_s2:inactive_s3:active_s4:active =
r2_13<<B_s1:inactive_s2:inactive_s3:active_s4:active@cytosol;
A::B_s1:active_s2:inactive_s3:active_s4:inactive =
r1_6>>+ r1_10<<+ r1_20>>+ r1_27<<+ r2_6>>A::B_s1:active_s2:inactive_s3:active_s4:inactive@cytosol;
A =
r2_1<<A@cytosol + r2_2<<A@cytosol + r2_3<<A@cytosol + r2_4<<A@cytosol + r2_5<<A@cytosol +
r2_6<<A@cytosol + r2_7<<A@cytosol + r2_8<<A@cytosol + r2_9<<A@cytosol + r2_10<<A@cytosol
+ r2_11<<A@cytosol + r2_12<<A@cytosol + r2_13<<A@cytosol + r2_14<<A@cytosol +
r2_15<<A@cytosol + r2_16<<A@cytosol;
B_s1:inactive_s2:inactive_s3:inactive_s4:inactive =
r2_16<<B_s1:inactive_s2:inactive_s3:inactive_s4:inactive@cytosol;
B_s1:inactive_s2:active_s3:inactive_s4:inactive =
r2_12<<B_s1:inactive_s2:active_s3:inactive_s4:inactive@cytosol;
A::B_s1:active_s2:active_s3:inactive_s4:active =
r1_3>>+ r1_11>>+ r1_17<<+ r1_26>>+ r2_3>>A::B_s1:active_s2:active_s3:inactive_s4:active@cytosol;
B_s1:active_s2:active_s3:active_s4:active =
r2_1<<B_s1:active_s2:active_s3:active_s4:active@cytosol;
B_s1:inactive_s2:active_s3:active_s4:active =
r2_9<<B_s1:inactive_s2:active_s3:active_s4:active@cytosol;
A::B_s1:inactive_s2:active_s3:inactive_s4:inactive =
r1_4<<+ r1_16>>+ r1_22<<+ r1_30<<+ r2_12>>A::B_s1:inactive_s2:active_s3:inactive_s4:inactive@cytosol;
A::B_s1:inactive_s2:inactive_s3:inactive_s4:inactive =
r1_8<<+ r1_16<<+ r1_24<<+ r1_32<<+ r2_16>>A::B_s1:inactive_s2:inactive_s3:inactive_s4:inactive@cytosol;
A::B_s1:inactive_s2:inactive_s3:active_s4:active =
r1_5<<+ r1_13<<+ r1_23>>+ r1_31>>+ r2_13>>A::B_s1:inactive_s2:inactive_s3:active_s4:active@cytosol;

A::B_s1:active_s2:inactive_s3:inactive_s4:active =
r1_7>>+ r1_11<<+ r1_19<<+ r1_28>>+ r2_7>>A::B_s1:active_s2:inactive_s3:inactive_s4:active@cytosol;

A::B_s1:active_s2:inactive_s3:inactive_s4:inactive =
r1_8>>+ r1_12<<+ r1_20<<+ r1_28<<+ r2_8>>A::B_s1:active_s2:inactive_s3:inactive_s4:inactive@cytosol;

B_s1:inactive_s2:active_s3:inactive_s4:active =
r2_11<<B_s1:inactive_s2:active_s3:inactive_s4:active@cytosol;

B_s1:active_s2:inactive_s3:active_s4:inactive =
r2_6<<B_s1:active_s2:inactive_s3:active_s4:inactive@cytosol;

B_s1:inactive_s2:active_s3:active_s4:inactive =
r2_10<<B_s1:inactive_s2:active_s3:active_s4:inactive@cytosol;

A::B_s1:inactive_s2:active_s3:inactive_s4:active =
r1_3<<+ r1_15>>+ r1_21<<+ r1_30>>+ r2_11>>A::B_s1:inactive_s2:active_s3:inactive_s4:active@cytosol;

A::B_s1:active_s2:active_s3:active_s4:inactive =
r1_2>>+ r1_10>>+ r1_18>>+ r1_25<<+ r2_2>>A::B_s1:active_s2:active_s3:active_s4:inactive@cytosol;

A::B_s1:inactive_s2:active_s3:active_s4:active =
r1_1<<+ r1_13>>+ r1_21>>+ r1_29>>+ r2_9>>A::B_s1:inactive_s2:active_s3:active_s4:active@cytosol;

B_s1:active_s2:inactive_s3:inactive_s4:active =
r2_7<<B_s1:active_s2:inactive_s3:inactive_s4:active@cytosol;

A::B_s1:active_s2:inactive_s3:active_s4:active =
r1_5>>+ r1_9<<+ r1_19>>+ r1_27>>+ r2_5>>A::B_s1:active_s2:inactive_s3:active_s4:active@cytosol;

A::B_s1:inactive_s2:inactive_s3:inactive_s4:active =
r1_7<<+ r1_15<<+ r1_23<<+ r1_32>>+ r2_15>>A::B_s1:inactive_s2:inactive_s3:inactive_s4:active@cytosol;

A::B_s1:active_s2:active_s3:inactive_s4:inactive =
r1_4>>+ r1_12>>+ r1_18<<+ r1_26<<+ r2_4>>A::B_s1:active_s2:active_s3:inactive_s4:inactive@cytosol;

B_s1:active_s2:active_s3:active_s4:inactive =
r2_2<<B_s1:active_s2:active_s3:active_s4:inactive@cytosol;

B_s1:active_s2:inactive_s3:inactive_s4:inactive =
r2_8<<B_s1:active_s2:inactive_s3:inactive_s4:inactive@cytosol;

B_s1:active_s2:active_s3:inactive_s4:inactive =
r2_4<<B_s1:active_s2:active_s3:inactive_s4:inactive@cytosol;

B_s1:inactive_s2:inactive_s3:inactive_s4:active =
r2_15<<B_s1:inactive_s2:inactive_s3:inactive_s4:active@cytosol;

B_s1:inactive_s2:inactive_s3:active_s4:inactive =
r2_14<<B_s1:inactive_s2:inactive_s3:active_s4:inactive@cytosol;

A::B_s1:active_s2:active_s3:active_s4:active =
r1_1>>+ r1_9>>+ r1_17>>+ r1_25>>+ r2_1>>A::B_s1:active_s2:active_s3:active_s4:active@cytosol;

B_s1:active_s2:active_s3:inactive_s4:active =
r2_3<<B_s1:active_s2:active_s3:inactive_s4:active@cytosol;
```

```
B_s1:active_s2:inactive_s3:active_s4:active =
r2_5<<B_s1:active_s2:inactive_s3:active_s4:active@cytosol;

A::B_s1:inactive_s2:inactive_s3:active_s4:inactive =
r1_6<<+ r1_14<<+ r1_24>>+ r1_31<<+ r2_14>>A::B_s1:inactive_s2:inactive_s3:active_s4:inactive@cytosol;

A::B_s1:inactive_s2:active_s3:active_s4:inactive =
r1_2<<+ r1_14>>+ r1_22>>+ r1_29<<+ r2_10>>A::B_s1:inactive_s2:active_s3:active_s4:inactive@cytosol;

//Model component:
B_sub ::=
B_s1:inactive_s2:inactive_s3:inactive_s4:inactive@cytosol[10] <*>
B_s1:active_s2:active_s3:active_s4:active@cytosol[0] <*>
B_s1:active_s2:active_s3:active_s4:inactive@cytosol[0] <*>
B_s1:active_s2:active_s3:inactive_s4:active@cytosol[0] <*>
B_s1:active_s2:active_s3:inactive_s4:inactive@cytosol[0] <*>
B_s1:active_s2:inactive_s3:active_s4:active@cytosol[0] <*>
B_s1:active_s2:inactive_s3:active_s4:inactive@cytosol[0] <*>
B_s1:active_s2:inactive_s3:inactive_s4:active@cytosol[0] <*>
B_s1:active_s2:inactive_s3:inactive_s4:inactive@cytosol[0] <*>
B_s1:inactive_s2:active_s3:active_s4:active@cytosol[0] <*>
B_s1:inactive_s2:active_s3:active_s4:inactive@cytosol[0] <*>
B_s1:inactive_s2:active_s3:inactive_s4:active@cytosol[0] <*>
B_s1:inactive_s2:active_s3:inactive_s4:inactive@cytosol[0] <*>
B_s1:inactive_s2:inactive_s3:active_s4:active@cytosol[0] <*>
B_s1:inactive_s2:inactive_s3:active_s4:inactive@cytosol[0] <*>
B_s1:inactive_s2:inactive_s3:inactive_s4:active@cytosol[0];

A_sub ::= A@cytosol[5];

Complexes_sub ::=
A::B_s1:active_s2:inactive_s3:active_s4:inactive@cytosol[0] <*>
A::B_s1:active_s2:active_s3:inactive_s4:active@cytosol[0] <*>
A::B_s1:inactive_s2:active_s3:inactive_s4:inactive@cytosol[0] <*>
A::B_s1:inactive_s2:inactive_s3:inactive_s4:inactive@cytosol[0] <*>
A::B_s1:inactive_s2:inactive_s3:active_s4:active@cytosol[0] <*>
A::B_s1:active_s2:inactive_s3:inactive_s4:active@cytosol[0] <*>
A::B_s1:active_s2:inactive_s3:inactive_s4:inactive@cytosol[0] <*>
A::B_s1:inactive_s2:active_s3:inactive_s4:active@cytosol[0] <*>
A::B_s1:active_s2:active_s3:active_s4:inactive@cytosol[0] <*>
A::B_s1:inactive_s2:active_s3:active_s4:active@cytosol[0] <*>
A::B_s1:active_s2:inactive_s3:active_s4:active@cytosol[0] <*>
A::B_s1:inactive_s2:inactive_s3:inactive_s4:active@cytosol[0] <*>
A::B_s1:active_s2:active_s3:inactive_s4:inactive@cytosol[0] <*>
A::B_s1:active_s2:active_s3:active_s4:active@cytosol[0] <*>
A::B_s1:inactive_s2:inactive_s3:active_s4:inactive@cytosol[0] <*>
A::B_s1:inactive_s2:active_s3:active_s4:inactive@cytosol[0];

B_sub <*>A_sub <*>Complexes_sub
```

C Narrative Language grammar

$$
\begin{array}{lcl}
\langle \mathit{model} \rangle & ::= & \langle \mathit{constants_decl} \rangle \langle \mathit{comparts_decl} \rangle \langle \mathit{compons_decl} \rangle \langle \mathit{reacts_decl} \rangle \langle \mathit{procs_decl} \rangle \\
 & & \\
\langle \mathit{constants_decl} \rangle & ::= & \text{Constants } \langle \mathit{constants_list} \rangle \\
\langle \mathit{comparts_decl} \rangle & ::= & \text{Compartments } \langle \mathit{comparts_list} \rangle \\
\langle \mathit{compons_decl} \rangle & ::= & \text{Components } \langle \mathit{compons_list} \rangle \\
\langle \mathit{reacts_decl} \rangle & ::= & \text{Reactions } \langle \mathit{reacts_list} \rangle \\
\langle \mathit{procs_decl} \rangle & ::= & \text{Narrative } \langle \mathit{procs_list} \rangle \\
 & & \\
\langle \mathit{constants_list} \rangle & ::= & \langle \mathit{constant} \rangle \\
 & | & \langle \mathit{constant} \rangle \langle \mathit{constants_list} \rangle \\
\langle \mathit{comparts_list} \rangle & ::= & \langle \mathit{compartment} \rangle \\
 & | & \langle \mathit{compartment} \rangle \langle \mathit{comparts_list} \rangle \\
\langle \mathit{compons_list} \rangle & ::= & \langle \mathit{component} \rangle \\
 & | & \langle \mathit{component} \rangle \langle \mathit{compons_list} \rangle \\
\langle \mathit{reacts_list} \rangle & ::= & \langle \mathit{reaction} \rangle \\
 & | & \langle \mathit{reaction} \rangle \langle \mathit{reacts_list} \rangle \\
\langle \mathit{procs_list} \rangle & ::= & \langle \mathit{proc} \rangle \\
 & | & \langle \mathit{proc} \rangle \langle \mathit{procs_list} \rangle \\
 & & \\
\langle \mathit{constant} \rangle & ::= & (\langle \mathit{const} \rangle, \langle \mathit{quantity} \rangle) \\
\langle \mathit{compartment} \rangle & ::= & (\langle \mathit{id} \rangle, \langle \mathit{compart_name} \rangle, \langle \mathit{opt_size} \rangle, \langle \mathit{opt_unit} \rangle, \langle \mathit{opt_dim} \rangle) \\
\langle \mathit{component} \rangle & ::= & (\langle \mathit{name} \rangle, \langle \mathit{opt_inform_descr} \rangle, \langle \mathit{opt_sites_def} \rangle, \\
 & & \langle \mathit{opt_states_def} \rangle, \langle \mathit{opt_comparts_def} \rangle, \langle \mathit{initial_quantity} \rangle) \\
\langle \mathit{reaction} \rangle & ::= & (\langle \mathit{id} \rangle, \langle \mathit{react_type} \rangle, \langle \mathit{rate} \rangle) \\
 & & \\
\langle \mathit{proc} \rangle & ::= & \text{Process } \langle \mathit{opt_inform_descr} \rangle \langle \mathit{events_list} \rangle \\
\langle \mathit{events_list} \rangle & ::= & \langle \mathit{event} \rangle \\
 & | & \langle \mathit{event} \rangle \langle \mathit{events_list} \rangle \\
\langle \mathit{event} \rangle & ::= & (\langle \mathit{id} \rangle, \langle \mathit{form_descr} \rangle, \langle \mathit{react_id} \rangle, \langle \mathit{opt_altern_event} \rangle) \\
 & & \\
\langle \mathit{opt_sites_def} \rangle & ::= & \\
 & | & \langle \mathit{sites_def} \rangle \\
\langle \mathit{sites_def} \rangle & ::= & \langle \mathit{site_def} \rangle \\
 & | & \langle \mathit{site_def} \rangle ; \langle \mathit{sites_def} \rangle \\
\langle \mathit{site_def} \rangle & ::= & \langle \mathit{name} \rangle : \langle \mathit{state_name} \rangle : \langle \mathit{is_active} \rangle \\
 & & \\
\langle \mathit{opt_states_def} \rangle & ::= & \\
 & | & \langle \mathit{states_def} \rangle \\
\langle \mathit{states_def} \rangle & ::= & \langle \mathit{state_def} \rangle \\
 & | & \langle \mathit{state_def} \rangle ; \langle \mathit{states_def} \rangle \\
\langle \mathit{state_def} \rangle & ::= & \langle \mathit{state_name} \rangle : \langle \mathit{is_active} \rangle \\
 & & \\
\langle \mathit{opt_comparts_def} \rangle & ::= & \\
 & | & \langle \mathit{comparts_def} \rangle \\
\langle \mathit{comparts_def} \rangle & ::= & \langle \mathit{compart_def} \rangle \\
 & | & \langle \mathit{compart_def} \rangle ; \langle \mathit{comparts_def} \rangle \\
\langle \mathit{compart_def} \rangle & ::= & \langle \mathit{id} \rangle : \langle \mathit{is_active} \rangle \\
 & & \\
\langle \mathit{initial_quantity} \rangle & ::= & (\langle \mathit{quantity} \rangle, \langle \mathit{opt_reliability} \rangle)
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{rate} \rangle & ::= & \mathit{rate\_const} \\
& \mid & \mathit{rate\_law} \\
\langle \mathit{rate\_const} \rangle & ::= & (\langle \mathit{rate\_value} \rangle, \langle \mathit{opt\_unit} \rangle, \langle \mathit{opt\_reliability} \rangle) \\
\langle \mathit{rate\_law} \rangle & ::= & \mathrm{fMA}(\mathit{quantity}) \\
& \mid & \mathrm{fMM}(\mathit{quantity}, \mathit{quantity}) \\
& \mid & \mathrm{fH}(\mathit{quantity}, \mathit{quantity}, \mathit{Int})
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{form\_descr} \rangle & ::= & \langle \mathit{event\_descr} \rangle \\
& \mid & \text{if } \langle \mathit{conds} \rangle \text{ then } \langle \mathit{event\_descr} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{conds} \rangle & ::= & \langle \mathit{cond} \rangle \\
& \mid & \langle \mathit{cond} \rangle \text{ and } \langle \mathit{conds} \rangle \\
\langle \mathit{cond} \rangle & ::= & \langle \mathit{names} \rangle \text{ is } \langle \mathit{state\_name} \rangle \\
& \mid & \langle \mathit{names} \rangle \text{ is not } \langle \mathit{state\_name} \rangle \\
& \mid & \langle \mathit{names} \rangle \text{ is in } \langle \mathit{id} \rangle \\
& \mid & \langle \mathit{names} \rangle \text{ is not in } \langle \mathit{id} \rangle \\
\langle \mathit{names} \rangle & ::= & \langle \mathit{name} \rangle \\
& \mid & \langle \mathit{name} \rangle.\langle \mathit{name} \rangle \\
& \mid & \langle \mathit{name} \rangle; \langle \mathit{names} \rangle \\
& \mid & \langle \mathit{name} \rangle.\langle \mathit{name} \rangle; \langle \mathit{names} \rangle \\
\langle \mathit{sites} \rangle & ::= & \langle \mathit{name} \rangle \\
& \mid & \langle \mathit{name} \rangle; \langle \mathit{sites} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{event\_descr} \rangle & ::= & \langle \mathit{complex\_name} \rangle \langle \mathit{bimol\_react} \rangle \langle \mathit{complex\_name} \rangle \text{ on } \langle \mathit{sites} \rangle \\
& \mid & \langle \mathit{complex\_name} \rangle \langle \mathit{bimol\_react} \rangle \langle \mathit{complex\_name} \rangle \\
& \mid & \langle \mathit{complex\_name} \rangle \langle \mathit{monomol\_react} \rangle \text{ on } \langle \mathit{sites} \rangle \\
& \mid & \langle \mathit{complex\_name} \rangle \langle \mathit{monomol\_react} \rangle \\
& \mid & \langle \mathit{complex\_name} \rangle \text{ relocates to } \langle \mathit{id} \rangle \\
& \mid & \langle \mathit{complex\_name} \rangle \text{ degrades} \\
& \mid & \langle \mathit{complex\_name} \rangle \text{ degrades } \langle \mathit{complex\_name} \rangle \\
& \mid & \langle \mathit{complex\_name} \rangle \text{ synthesises } \langle \mathit{complex\_name} \rangle \\
& \mid & \langle \mathit{complex\_name} \rangle \text{ homodimerizes} \\
& \mid & \langle \mathit{complex\_name} \rangle \text{ dehomodimerizes} \\
& \mid & \langle \mathit{complex\_name} \rangle \text{ dimerizes with } \langle \mathit{complex\_name} \rangle \\
& \mid & \langle \mathit{complex\_name} \rangle \text{ dedimerizes from } \langle \mathit{complex\_name} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{complex\_name} \rangle & ::= & \langle \mathit{name} \rangle \\
& \mid & \langle \mathit{name} \rangle : \langle \mathit{complex\_name} \rangle \\
\langle \mathit{id} \rangle & ::= & \mathit{Int} \\
\langle \mathit{opt\_size} \rangle & ::= & \\
& \mid & \mathit{Int} \mid \mathit{const} \\
\langle \mathit{opt\_unit} \rangle & ::= & \\
& \mid & \mathit{Str} \\
\langle \mathit{opt\_dim} \rangle & ::= & \\
& \mid & \mathit{Int} \\
\langle \mathit{name} \rangle & ::= & \mathit{Ide} \\
\langle \mathit{opt\_inform\_descr} \rangle & ::= & \\
& \mid & \mathit{Str}
\end{array}
$$

⟨*quantity*⟩	::=	*value* | *const*
⟨*value*⟩	::=	*Int* | *Real*
⟨*const*⟩	::=	*Ide*
⟨*opt_reliability*⟩	::=	
	|	*Int*
⟨*rate_value*⟩	::=	*quantity*
⟨*react_id*⟩	::=	*Int*
⟨*opt_altern_event*⟩	::=	
	|	alternative to ⟨*id*⟩
⟨*is_active*⟩	::=	*Bool*
⟨*compart_name*⟩	::=	nucleus | cytosol | exosol
	|	cellMembrane | nucleusMembrane | *Ide*
⟨*react_type*⟩	::=	phosphorylation | dephosphorylation
	|	binding | unbinding
	|	homodimerization | dehomodimerization
	|	dimerization | dedimerization
	|	activation | deactivation
	|	hydrolysis | dehydrolysis
	|	degradation | synthesis | relocation
⟨*state_name*⟩	::=	phosphorylated | bound | active | hydrolysed | dimer
⟨*bimol_react*⟩	::=	phosphorylates | dephosphorylates | binds | unbinds
	|	activates | deactivates | hydrolyses | dehydrolyses
⟨*monomol_react*⟩	::=	phosphorylates | dephosphorylates | hydrolyses | dehydrolyses