Mary-Ann Blätke[1]

BioModelKit – An Integrative Framework for Multi-Scale Biomodel-Engineering

[1] Leibniz Institute of Plant Genetics and Crop Plant Research (IPK), Department of Molecular Genetics, Corrensstrasse 3, 06466 Seeland OT Gatersleben, Germany, E-mail: blaetke@ipk-gatersleben.de. http://orcid.org/0000-0002-4790-7377.

Abstract:
While high-throughput technology, advanced techniques in biochemistry and molecular biology have become increasingly powerful, the coherent interpretation of experimental results in an integrative context is still a challenge. *BioModelKit* (*BMK*) approaches this challenge by offering an integrative and versatile framework for biomodel-engineering based on a modular modelling concept with the purpose: (i) to represent knowledge about molecular mechanisms by consistent executable sub-models (modules) given as Petri nets equipped with defined interfaces facilitating their reuse and recombination; (ii) to compose complex and integrative models from an *ad hoc* chosen set of modules including different *omic* and abstraction levels with the option to integrate spatial aspects; (iii) to promote the construction of alternative models by either the exchange of competing module versions or the algorithmic mutation of the composed model; and (iv) to offer concepts for (*omic*) data integration and integration of existing resources, and thus facilitate their reuse. *BMK* is accessible through a public web interface (www.biomodelkit.org), where users can interact with the modules stored in a database, and make use of the model composition features. *BMK* facilitates and encourages multi-scale model-driven predictions and hypotheses supporting experimental research in a multilateral exchange.

Keywords: model database, model composition, model mutation, spatiotemporal modelling, Petri nets

DOI: 10.1515/jib-2018-0021

Received: March 16, 2018; **Revised:** May 21, 2018; **Accepted:** June 7, 2018

1 Introduction

In the mid-1990s, systems biology emerged as a new discipline in biosciences aiming to achieve a systems-level understanding by encouraging the development of advanced high-throughput technologies to investigate molecular mechanisms on various *omic* levels. The consistent interpretation and integration of the resulting big data, which is not only numerous but also complex and diverse, is still a challenging field, which demands integrative modelling and analysis frameworks.

The well-established formalism of *Petri nets* (*PNs*) has been shown to ideally support multidisciplinary research [1], [2], [3], [4]. By their definition, PNs allow for an intuitive representation of reaction networks including concurrency and synchronisation [5], [6].

Mary-Ann Blätke is the corresponding author.

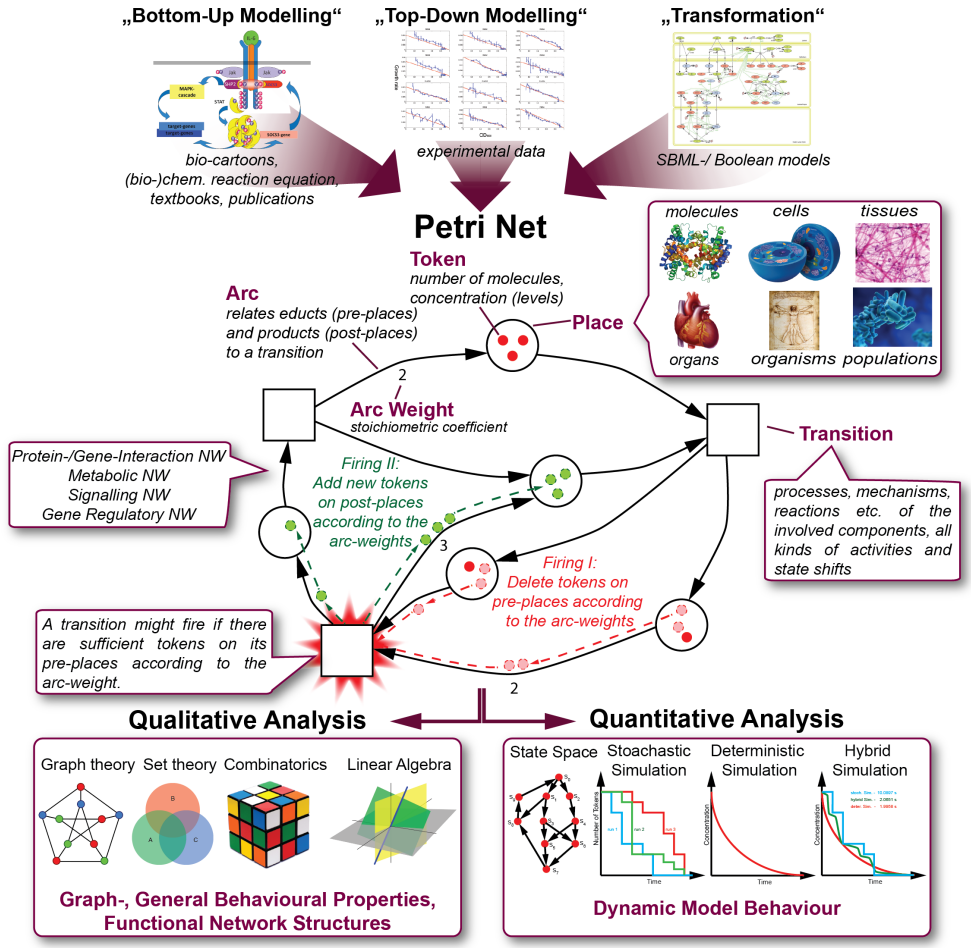

Figure 1: Overview of the PN framework.

A PN is defined by two sets of nodes, *places* representing conditions (e.g. molecules) and *transitions* describing events (e.g. reactions), that are connected via *directed arcs* carrying *arc-weights* to reflect the stoichiometry. *Tokens*, which reside only in places, indicate the value of a condition. A transition is a called *enabled* and may *fire* if the number of tokens (marking) on the pre-places is equal to or greater than the corresponding arc-weights. Only one transition can fire at a time. During the firing, the chosen enabled transition removes tokens from its pre-places and adds tokens to its post-places, both according to the arc-weight. After firing a new *state* is reached. Playing the so-called *token game* allows exploring the state space.

The *standard, i.e. discrete and time-free PNs* have been initially used for structural analysis, see, e.g. [7], [8], [9], [10], [11]. Moreover, PNs offer a unifying framework, which empowers the quantitative (i.e. stochastic, continuous, hybrid) interpretations of a qualitative model [12], [13]. *Coloured PNs*, combine PNs with a programming language to obtain a scalable modelling language for concurrent systems. The combination of PNs with a programming language provides the primitives for encoding data manipulation and creating compact and parameterizable models [14]. Therefore, coloured PNs offer a convenient framework to model complex multi-scale [15], multi-level and multi-dimensional aspects of a biomolecular system, see also [16] for a recent review.

For our purposes, we decided to use the compatible tools:

- *Snoopy* – a tool for modelling and animation/simulation of hierarchical uncoloured/coloured qualitative, stochastic, continuous and hybrid PNs [17], [18], [19];
- *Charlie* – an analysis tool of standard PN properties [20];
- *Marcie* – a symbolic reachability analysis tool, including CTL and CSL and model checking [21].

which support all aspects of PNs as described above and more.

Modelling concepts need to be designed in a way to provide a clear and precise outline of the modelled molecular mechanisms and to adapt the model content according to the analysis focus flexibly. These two claims involve the reuse and recombination of models or parts thereof. Here, the fundamental concept of modularisation that has been widely used in engineering to reduce the complexity of problems by decomposing a

system into functional parts with defined interfaces comes into focus. *BioModelKit* (short, *BMK*) seizes on the idea of modularisation in the context of modelling; we summarize related work in Section 2.

BMK is a web-based platform (www.biomodelkit.org) that relies on an *a priori* molecule-centred modularisation concept, where sub-models (*modules*) are designed consistently with matching interface networks to allow for the automatic composition of complex and coherent models, see Section 3. Molecule-centred means that a module describes the function and interaction of one particular component. Section 3.1 provides all necessary terminology of the used modularisation concept. *BMK* makes use of different module types to integrate molecular mechanisms and correlations on the main *omic* level (metabolome, proteome, transcriptome, genome), i.e. we distinguish *mechanistic modules* (protein, protein degradation, mRNA, and gene modules) and *causal modules* (causal influence and allelic influence modules), see Figure 2 – Box 1 and Section 3.2 for details. A module is given as a PN, which relates the molecular structure of the component and different conformational states thereof (places) to molecular events (transitions) interfering with the molecular function, see Section 3.3.

The module coupling is not restricted to a particular *omic* or abstraction level allowing to aggregate molecular mechanisms and correlations on various levels in a composed model consistently. Modules can be recombined arbitrarily to compose alternative models with competing module versions or reused in the context of a different biological system. The arbitrary recombination of modules enables the user to decide also about which *omic* levels to include in a composed model. Section 4 describes the details of the model composition based on an *ad hoc* chosen set of modules.

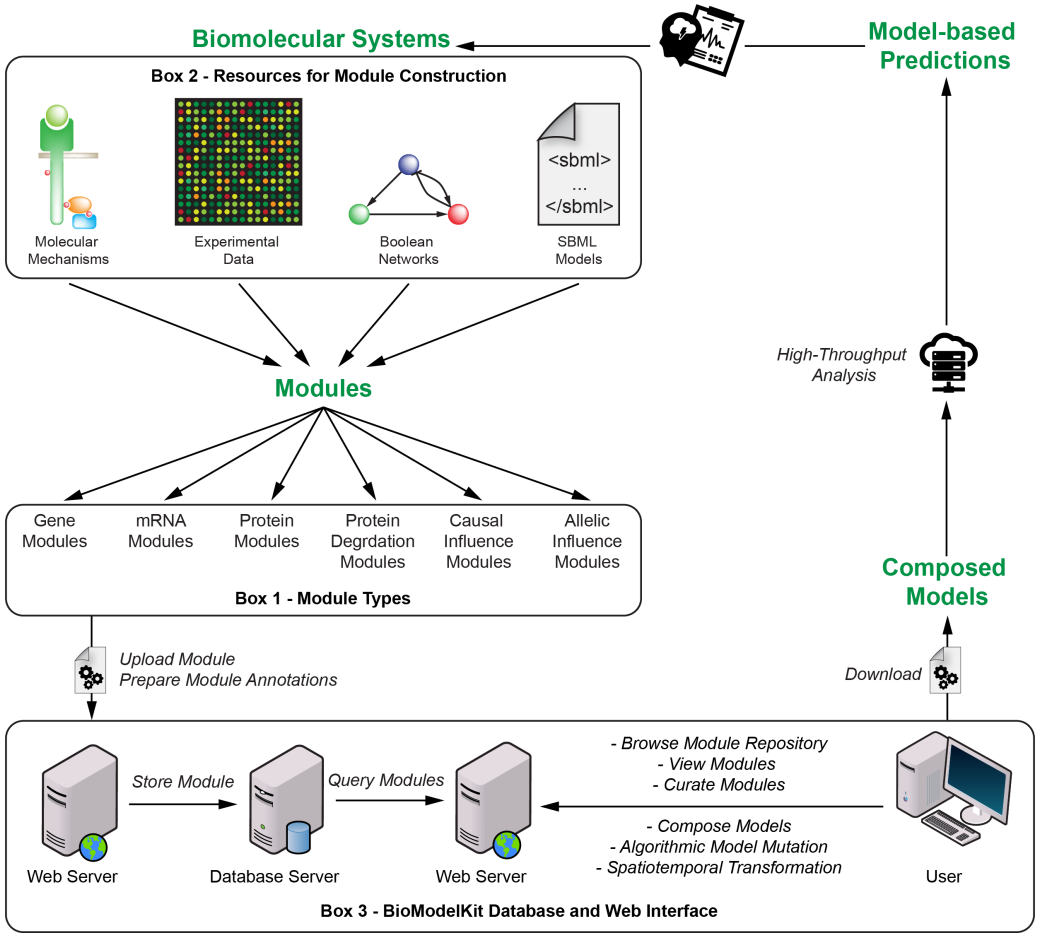

Figure 2: Overview of the BioModelKit framework.

BMK offers several additional features to enhance the versatility of the proposed framework. First, an annotation file in the *BMK markup language* (*BMKml*) complements the PN graph of each module, holding details about the authorship and modelling process, documentation of all model elements, and cross-references to resources. So far, we exploit three ways to obtain modules, see Figure 2 – Box 2: (i) direct engineering of modules based on published knowledge; (ii) reverse engineering of modules from complex (*omic*) data sets; and (iii) modularisation of gene regulatory networks [22] and SBML encoded models [23]. The three approaches are discussed in Section 5.2. The *algorithmic mutation* is another feature of *BMK* to generate alternative models automatically by mimicking the effect of gene knockouts and structure-function mutations. This approach allows for in-depth *in silico* mutation studies to determine variations in a molecular mechanism reproducing

the desired function, see Section 5.3. Another recent addition to *BMK* is the spatiotemporal transformation that adds a position to all components in the composed model, which allows tracking the movement of the components and their interaction in a defined space (1-, 2-, or 3-dimensional) with restriction to the assumed cellular arrangement [24], details are explained in Section 5.4.

The accessibility of modules and the model composition, as well as the features mentioned above, are crucial components of *BMK*, which is realised through the web interface and the database, see Figure 2 – Box 3, Section 6 discusses aspects of their implementation.

A brief discussion and outlook on *BMK* is given in Section 7.

2 Related Work

In system biology, modularisation approaches have mostly been applied to decompose complex models for analysis purposes, see [25]. Only a few approaches like the one from Cooling et al. [26] suggest composing models of *ab inito* built standard virtual parts (SVPs), which are accessible via CellML [27]. In all cases, manual adjustments are required to couple the obtained modules.

The public dissemination of models is another critical issue in life sciences to not only reproduce results but also to accelerate their reuse. Next to CellML, which is a repository for a wide range of biological models, the Biomodels database [28], BIGG Models [29] and the SEEK platform [30] offer an infrastructure to collect and exchange published models.

While some of the model repositories mentioned above support the simulation of models, none of them supports the automatic composition of models from a set of existing models. The major reasons inferring the automated coupling of models are the use of diverse modelling formalisms, the incompatibility of granularities, the lack of naming convention standards of model elements (reactions, species, parameters etc.) and the inconsistent formulation of assumptions and constraints used in a model concomitant with insufficient model documentation [31].

As already introduced, *BMK* applies the concept of modularisation to compose models from a set of modules, that are given as reuseable Petri net submodels with interfaces, which are specifically and consistently designed for model composition in an automated fashion. To the best of our knowledge, the framework established with *BMK* is unique. There exist no comparable frameworks or tools.

3 Modularisation

BMK employs an *a priori* molecule-centred modularisation concept, where sub-models (modules) are designed consistently with matching interface networks to allow for the automatic model composition. Molecule-centred means that a module describes the function of one particular component by a PN. The PN relates the molecular structure of the component and its conformational states (places) to molecular events (transitions) interfering with the molecular function. Below, we introduce the formalisation used within *BMK*, compare Figure 3, which exemplifies the modularisation and model composition concept on a simple example consisting of a kinase and its transcription factor as a feedback activator.

3.1 Module Terminology

In this section, we introduce the module terminology of the molecule-centred modularisation approach as in [32]. This formalisation is essential for the realisation of *BMK* including all of its features and functionalities. A brief example follows every term about the running example depicted in Figure 3. Figure 3(A) provides a short description of the molecular mechanisms, while Figure 3(B) depicts the corresponding modules, respectively the composed model.

A *component* is any chemical compound that contributes to a biological process at the molecular level.

Running Example

Components in the running example are kinase protein *pK*, kinase mRNA *mK*, kinase gene *gK*, transcription factor protein *pTF*.

A *genetic component* is a component with genetic information like genes, mRNAs, and proteins.

Running Example
All of the components above are genetic components.

A *non-genetic component* is a component without genetic information like metabolites, second messengers, energy equivalents, ions, etc.

Running Example
None of components above is a non-genetic component.

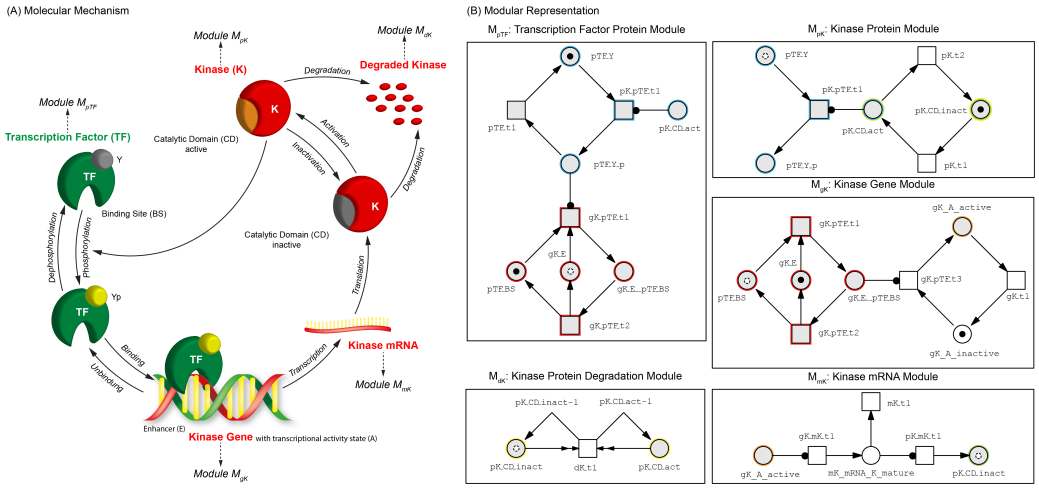

Figure 3: (A) Molecular mechanism: Active kinase (K) phosphorylates its transcription factor (TF), the phosphorylated TF binds to the K gene, which starts the transcription, followed by protein translation of K. (B) Modular representation by PNs: Two protein modules for the TF and K, one gene, mRNA and protein degradation module for K according to the molecular mechanism in (A). Nodes highlighted in the same colour define a shared interface network. Modules share their tokens after model composition and thus, the dotted tokens appear on shared places enabling the execution of interactions.

A *module* M_{c_0} describes the functionality of one particular genetic component c_0, called *main component*, including its interactions with $n_{IC} \geq 0$ components. Here, $C(M_{c_0}) = \{c_0, \ldots, c_{n_{IC}}\}$ defines the total set of components of module M_{c_0}. We distinguish mechanistic module types (protein, protein degradation, mRNA and gene modules) and causal module types (allelic influence and causal influence modules), see Section 3.2 for details.

Running Example
Modules related to

- kinase *K*: protein module M_{pK}, protein degradation module M_{dK}, gene module M_{gK}, and mRNA module M_{mK};
- transcription factor *TF*: protein module M_{pTF}

A *functional unit* u is a defined part of a component $c \in C(M_{c_0})$ with an assigned function that is of importance for the functionality of the component. A genetic component $c \in C(M_{c_0})$ consists of $n_{FU} \geq 1$ functional units. We assume that non-genetic components cannot be decomposed into more than one functional unit. Furthermore, $U(c) = \{u_1, \ldots, u_{n_{FU}}\}$ defines the total set of functional units of component $c \in C(M_{c_0})$, $\mathcal{U}(M_{c_0}) = \bigcup_{c \in C(M_{c_0})} U(c)$ defines the total set of functional units in module M_{c_0}.

Running Example
Functional units of the involved components are

- kinase protein *pK*: catalytic domain (*CD*) part of M_{pK}, M_{dK} and M_{mK}, as well as M_{pTF} due to its interaction;
- kinase gene *gK*: (i) enhancer (*E*) part of M_{gK}, as well as M_{pTF} due to its interaction, (ii) artificial functional unit for transcriptional activity (*A*) part of M_{gK}, as well as M_{mK} because *A* triggers the mRNA synthesis;
- kinase mRNA *mK*: RNA sequence (*mRNA_K*) part of M_{mK}; and
- transcription factor protein *pTF*: (i) tyrosine residue (*Y*) part of M_{pTF}, as well as M_{pK} due to its interaction, (ii) DNA binding site (*BS*) part of M_{pTF}, as well as M_{gK} due to its interaction.

A *molecular state s* describes a specific physical constitution of a functional unit $u \in \mathcal{U}(M_{c_0})$. A functional unit $u \in U(c)$ can adopt $n_{MS} \geq 1$ different molecular states. Furthermore, $S(u) = \{s_1, \ldots, s_{n_{MS}}\}$ defines the total set of molecular states of a functional unit $u \in U(c)$ and $\mathcal{S}(M_{c_0}) = \bigcup_{u \in \mathcal{U}(M_{c_0})} S(u)$ defines the summation of all sets $S(u)$ over $u \in \mathcal{U}(M_{c_0})$. Each molecular state $s \in \mathcal{S}(M_{c_0})$ can be mapped to at least one functional unit according to the function $\lambda^u(s) = \{u \in \mathcal{U}(M_{c_0}) \mid s \in S(u)\}$ and to at least one component according to the function $\lambda^c(s) = \{c \in C(M_{c_0}) \mid s \in \bigcup_{u \in U(c)} S(u)\}$.

Running Example
Molecular states of functional units

- catalytic domain *CD* of kinase protein *pK*: active (*pK_CD_act*) or inactive (*pK_CD_inact*, initial/active molecular state in M_{pK});
- enhancer *E* of kinase gene *gK*: free (*gK_E*, initial/active molecular state in M_{gK}) or bound to DNA binding site *BS* of transcription factor protein *pTF* (*gK_E__pTF_BS*);
- transcriptional activity of the kinase gene *A*: active (*gK_A_act*) or inactive (*gK_A_inact*, initial/active molecular state in M_{gK});
- RNA sequence *mRNA_K* of kinase mRNA *mK*: mature (*mK_mRNA_K_mature*);
- tyrosine residue *Y* of transcription factor protein *pTF*: phosphorylated (*pTF_Y*, initial/active molecular state in M_{pTF}) or unphosphorylated (*pTF_Y_p*); and
- DNA binding site *BS* of transcription factor protein *pTF*: free (*pTF_BS*, initial/active molecular state in M_{pTF}) or bound to enhancer *ES* of kinase gene *gK* (*gK_E__pTF_BS*).

An *interaction state* s_{IS} is a molecular state $s_{IS} \in \mathcal{S}(M_{c_0})$ if it can be mapped to more than one component $|\lambda^c(s)| > 1$ defining a complex $k = \lambda^c(s)$. Accordingly, $\mathcal{S}_{IS}(M_{c_0}) = \{s \in \mathcal{S}(M_{c_0}) \mid |\lambda^c(s)| > 1\}$, $\mathcal{S}_{IS}(M_{c_0}) \subseteq \mathcal{S}(M_{c_0})$ defines the total set of interaction states.

Running Example
The complex of the transcription factor protein *pTF* and the kinase gene *gK* defines an interaction state *gK_E__pTF_BS* in M_{pTF} and M_{gK}.

The term *active molecular state* defines the current state of a functional unit $u \in \mathcal{U}(M_{c_0})$, only one molecular state can be active at each time point. The function $\alpha : S(u) \to \{0, 1\}$ determines the active molecular state with the constraint $\sum_{s \in S(u)} \alpha(s) = 1$. If $\alpha(s) = 1$ the molecular state s is active, otherwise if $\alpha(s) = 0$ the molecular state s is inactive.

Running Example
See above, examples for molecular states.

The *initial state* $s_{0,u} \in S(u)$ with $\alpha(s_{0,u}) = 1$ and $s_{0,u} \notin \mathcal{S}_{IS}(M_{c_0})$ is assumed to be the ground state of a functional unit, which can not be an interaction state. This assumption is necessary to exclude inconsistencies during the composition of models. The union of all initial states is given by $S_0(M_{c_0}) = \bigcup_{u \in U(c_0)} s_{0,u}$.

Running Example
See above, examples for molecular states.

A *molecular event e* represents an action changing the molecular states of the functional units. The relations between molecular states in $\mathcal{S}(M_{c_0})$ are described by $n_{ME} \geq 1$ molecular events. Here, $\mathcal{E}(M_{c_0}) = \{e_1, \ldots, e_{n_{ME}}\}$ defines the total set of molecular events.

Running Example
Molecular events are

- *pK_t*1 in M_{pK}: activation of catalytic domain *pK_CD_inact* to *pK_CD_act*;
- *pK_t*2 in M_{pK}: inactivation of catalytic domain *pK_CD_act* to *pK_CD_inact*;
- *pK_pTF_t*1 in M_{pK} and M_{pTF}: phosphorylation of tyrosine residue *pTF_Y* to *pTF_Y_p* triggered by the active catalytic domain of the kinase *pK_CD_act*;
- *pTF_t*1 in M_{pK} and M_{pTF}: dephosphorylation of tyrosine residue *pTF_Y_p* to *pTF_Y*;

- *gK_pTF_t*1 in M_{gK} and M_{pTF}: binding of *pTF_BS* and *gK_E* forming the complex *gK_E__pTF_BS*;
- *gK_pTF_t*2 in M_{gK} and M_{pTF}: dissociation of complex *gK_E__pTF_BS* to *pTF_BS* and *gK_E*;
- *gK_pTF_t*3 in M_{gK}: activation of *gK_A_inact* to *gK_A_act* triggered by the complex *gK_E__pTF_BS*;
- *gK_t*1 in M_{gK}: inactivation of *gK_A_act* to *gK_A_inact*;
- *gK_mK_t*1 in M_{mK}: synthesis of mature kinase mRNA *mK_mRNA_K_mature*;
- *mK_t*1 in M_{mK}: degradation of mature kinase mRNA ewgdsfgg *mK_mRNA_K_mature*;
- *pK_mK_t*1 in M_{mK}: synthesis of kinase protein defined by *pK_CD_inact*; and
- *dK_t*1 in M_{dK}: degradation of kinase protein defined by *pK_CD_inact* and *pK_CD_act*.

The *stoichiometric coefficient* $\nu(e,s) \in \mathbb{N}_0$ (products) and $\nu(s,e) \in \mathbb{N}_0$ (educts) defines the stoichiometry with which a molecular state $s \in \mathcal{S}(M_{c_0})$ mediates a molecular event $e \in \mathcal{E}(M_{c_0})$. The set of molecular states participating in a molecular event e can be distinguished into a set of educts $\bullet e = \{s \in \mathcal{S}(M_{c_0}) \mid \nu(s,e) \neq 0\}$, and a set products $e\bullet = \{s \in \mathcal{S}(M_{c_0}) \mid \nu(e,s) \neq 0\}$. Vice versa, $\bullet s = \{e \in \mathcal{E}(M_{c_0}) \mid \nu(e,s) \neq 0\}$ defines the set of molecular events, where the molecular state s is a product, and $s\bullet = \{e \in \mathcal{E}(M_{c_0}) \mid \nu(s,e) \neq 0\}$ defines the set of molecular events, where molecular state s is an educt. A molecular event $e \in \mathcal{E}(M_{c_0})$ can be defined as $e = \{(\varepsilon(\bullet e), \varepsilon(e\bullet)), \varepsilon(\bullet e) \rightarrow \varepsilon(e\bullet)\}$, where $\varepsilon(\bullet e) = \sum_{s \in \bullet e} \nu(s,e) \cdot s$ and $\varepsilon(e\bullet) = \sum_{s \in e\bullet} \nu(e,s) \cdot s$. Each molecular event $e \in \mathcal{E}(M_{c_0})$ occurs with a rate $r(e)$.

Running Example
All stoichiometric coefficients are equal to one.

An *interaction event* e_{IS} is a molecular event $e_{IS} \in \mathcal{E}(M_{c_0})$ that involves more than one component, $|\lambda^c(\bullet e \cup e\bullet)| > 1$. The total set of interaction events is given by $\mathcal{E}_{IA}(M_{c_0}) = \{e : |\lambda^c(\bullet e \cup e\bullet)| > 1\}$.

Running Example
Interaction events given by interaction of (i) kinase and transcription factor protein (*pK_pTF_t*1), (ii) kinase gene and transcription factor protein (*gK_pTF_t*1, *gK_pTF_t*2, *gK_pTF_t*3), (iii) kinase gene and mRNA (*gK_mK_t*1), and (iv) kinase mRNA and protein (*pK_mK_t*1).

A *molecular process* $E(u)$ defines the set of molecular events changing the molecular state of a functional unit $u \in U(c)$, such that $E(u) = \bigcup_{u \in S(u)}((\bullet s \cup s\bullet) \quad (\bullet s \cap s\bullet))$.

Running Example
Molecular processes in module

- M_{pK}: (i) *pK_t*1 and *pK_t*2 for *pk_CD*, and (ii) *pK_pTF_t*1 for *pTF_Y*;
- M_{gK}: (i) *gK_pTF_t*1 and *gK_pTF_t*2 for *pTF_CD* and *gK_E*, and (ii) *gK_pTF_t*3 and *gK_t*1 for *gK_A*;
- M_{mK}: (i) *gK_mK_t*1 and *mK_t*2 for *mK_mRNA_K_mature*, and (ii) *pK_mK_t*1 for *pK_CD*;
- M_{pTF}: (i) *pK_pTF_t*1 and *pTF_t*1 for *pTF_Y*, and (ii) *gK_pTF_t*1 and *gK_pTF_t*2 for *pTF_CD* and *gK_E*; and
- M_{dK}: (i) *dK_t*1 for *pK_CD*.

3.2 Module Types

The modularisation concept makes use of different module types to integrate molecular mechanisms and correlations on all of the main omic levels (metabolome, proteome, transcriptome, genome). Accordingly, we distinguish mechanistic module types [33], compare Figure 2 – Box 2:

A *gene module* M_{g,c_0} represents the transcriptional activity of a gene controlled by the formation of the regulatory landscape and the pre-initiation complex. These processes include complex non-covalent interactions with transcription factors or other regulatory proteins interacting with a silencer or enhancer sequences of the gene.

A *mRNA module* M_{m,c_0} represents the biosynthesis of a particular mRNA by transcription of the respective gene; the post-transcriptional modification of the mRNA including capping, (alternative) splicing and polyadenylation; the translation into the proteins encoded by the processed mRNA; and the degradation of the

mRNA including its potential control through proteins or small interfering anti-sense RNA molecules. These processes include covalent modification and non-covalent interactions of the mRNA.

A *protein module* M_{p,c_0} describes the functionality of a particular protein (single polypeptide chain), including changes in the protein conformation, non-covalent interactions with other components and covalent modifications that regulate the functionality. Thus, a protein module represents the formation and cleavage of covalent and non-covalent bonds, as well as conformational changes of the protein structure.

A *protein degradation module* M_{d,c_0} represents the degradation of a protein by proteolysis in lysosomes, ubiquitin-dependent degradation by the proteasome, or degradation by digestive enzymes or by any other possible mechanism that may lead to the degradation or inactivation of the protein. The post-translational proteolytic processing is described in the corresponding protein module.

The module types introduced above exclusively rely on known molecular mechanisms, even though some mechanistic details may not necessarily be considered. To integrate causal relationships if molecular mechanisms unknown, we introduce two more module types [33]:

An *allelic influence module* M_{ai,c_0} represents the effects of alleles (mutated versions of a gene) on molecular processes. In contrast to gene modules, the described effects are causal influences, which might be directly or indirectly mediated by unknown processes.

A *causal influence module* M_{ci,c_0} describes the influence of arbitrary entities, others than alleles, on molecular processes.

3.3 Module Petri Net

The formal description of a module M, see Section 3.1 has to be translated into a PN $\mathcal{N} = \{P, T, F, f, v, m_0\}$. Places represent molecular states and transitions represent molecular events. The function $\varrho_{p\to s} : P \to \mathcal{S}(M)$ ($\varrho_{t\to e} : T \to \mathcal{E}(M)$) maps a place $p \in P$ (transition $t \in T$) to a molecular state $s \in \mathcal{S}(M)$ (molecular event $e \in \mathcal{E}(M)$), both mappings are bijective.

The PN $\mathcal{N}(M_{c_0}) = \{P, T, F, f, v, m_0\}$ of a module M_{c_0} is defined as:

- Set of places $P = \bigcup_{c\in C(M_{c_0})} P^c$, where
  - $P^c = \bigcup_{u\in U(c)} P^u$ is the set of places representing a component $c \in C(M_{c_0})$ and $P^u = \{p : \varrho_{p\to s}(p) \in S(u)\}$ is the set of places representing a functional unit $u \in \mathcal{U}(M_{c_0})$;
- Set of transitions $T = \bigcup_{c\in C(M_{c_0})} T^c$, where
  - $T^c = \bigcup_{u\in U(c)} T^u$ is the set of transitions related to a component $c \in C(M_{c_0})$ and $T^u = \{t : \varrho_{t\to e}(t) \in E(u)\}$ is the set of transitions related to a functional unit $u \in \mathcal{U}(M_{c_0})$;
- Set of arcs $F := F_{SA} \subseteq (P \times T) \cup (T \times P)$;
- Arc-weights f: $F \to \mathbb{N}_0$, where
  - $\forall s \in \mathcal{S}(M_{c_0}), e \in \mathcal{E}(M_{c_0})$ with $s \in \bullet e$: $f_{SA}(\varrho^{-1}_{p\to s}(s), \varrho^{-1}_{t\to e}(e)) = v(s, e)$ (input arc), and
  - $\forall s \in \mathcal{S}(M_{c_0}), e \in \mathcal{E}(M_{c_0})$ with $s \in e\bullet$: $f_{SA}(\varrho^{-1}_{t\to e}(e), \varrho^{-1}_{p\to s}(s)) = v(e, s)$ (output arc);
- Set of firing rates $v : T \to H$ with $H = \bigcup_{t\in T} h(t)$, where:
  - $\forall e \in \mathcal{E}(M_{c_0}) : h(\varrho^{-1}_{t\to e}(e)) = r(e)$;
- Initial marking m_0: $P \to \mathbb{N}_0$:
  - M_{c_0} is a gene, protein, causal or allelic influence module:
    - $\forall s \in S_0(M_{c_0}) : m_0(\varrho^{-1}_{p\to s}(s)) = n_{c_0}$ (n_{c_0} is the assumed initial number of copies for component c_0, here we assume $n_{c_0} = 1$), and
    - $\forall s \in \mathcal{S}(M_{c_0}) \quad S_0(M_{c_0}) : m_0(\varrho^{-1}_{p\to s}(s)) = 0$,
  - M_{c_0} is a protein degradation or mRNA module:
    - $\forall s \in \mathcal{S}(M_{c_0}) : m_0(\varrho^{-1}_{p\to s}(s)) = 0$.

Running Example
See Figure 3(B) for the Petri net representation of the modules used in the running example in Figure 3(A).The modules are also provide as Snoopy-files in Supplementary Material 1.

The protein degradation module M_{d,c_0} is a special case with respect to the use of arcs. Assuming a transition $t \in T$ represents the molecular event of protein degradation $e = \varrho_{t\to e}(t)$ of the protein defined by module $M_{p,c_0'}$, then the set of places representing:

- none interaction states: $P^{nonIS}_{M_{p,c_0'}} = \varrho^{-1}_{p\to s}(\bigcup_{u\in U(c_0')} S(u) \quad \mathcal{S}_{IS}(M_{p,c_0'}))$ are connected with transition t using:
  - reset arcs $f_{XA}(P^{nonIS}_{M_{p,c_0'}}, t) = 1$, and
  - marking-dependent standard arcs $f_{SA}(t, P^{nonIS}_{M_{p,c_0'}}) = m_0(P^{nonIS}_{M_{p,c_0'}})$;
- interaction states: $P^{IS}_{M_{p,c_0'}} = \varrho^{-1}_{p\to s}(\bigcup_{u\in U(c_0')} S(u) \cap \mathcal{S}_{IS}(M_{p,c_0'}))$ are connected with transition t using:
  - inhibitory arcs $f_{IA}(P^{IS}_{M_{p,c_0'}}, t) = 1$.

This transformation ensures that proteins are only degraded if they are not interacting with other components and that only one copy is degraded.

Running Example
In the kinase protein degradation module M_{dK}, transitions *dK_t*1 is representing the degradation of the kinase protein *pK*. The none-interaction states of the kinase protein *pK* are *pK_CD_act* and *pK_CD_inact*. Each of the two corresponding places with the initial markings m_0(*pK_CD_act*), respectively m_0(*pK_CD_inact*), is connected with transitions *dK_t*1 via a reset arc (arc with two arrowheads), which deletes all tokens in case of firing, and a marking-dependent standard arc, which adds m_0(*pK_CD_act*) − 1 tokens to place *pK_CD_act* and m_0(*pK_CD_inact*) − 1 tokens to place *pK_CD_inact*. This subnet allows deleting exactly one copy the kinase protein *pK*, regardless of the active state of the functional unit *CD*.
The kinase protein module M_{dK} includes no interaction state. If there would exist an interaction state represented by a place p, it must have been connected with transition *dK_t*1 via an inhibitory arc (arc with an empty circle as the arrowhead). Thus, the degradation of the protein kinase *pK* could not occur as long as place p is marked.

3.4 Interface Networks

Interface networks are shared subgraphs of modules describing an interaction between the represented components or a functional relationship (e.g. links between translation, transcription or protein degradation), in the most trivial case interface networks consist only of a single place. Identical interface networks are used to couple modules and are therefore crucial for the model composition.
A set of modules $I = \{M_1, \ldots, M_n\}$ of components involved in one particular interaction share the interface network $\mathcal{N}_I(M) = \{P^I, T^I, F^I, f^I, v^I, m_0^I(M)\}$, where $\mathcal{N}_I(M) \subseteq \mathcal{N}(M)$, $M \in I$. All nodes in an interface network $\mathcal{N}_I(M)$ are declared as logical (fusion) nodes. The set of places P^I and transitions T^I in an interface network $\mathcal{N}_I$ with e.g. $I = \{M_{c_{0,1}}, M_{c_{0,2}}\}$ depends on the type of interaction among the two modules $M_{c_{0,1}}$ and $M_{c_{0,2}}$, which can be categorised into:

- regulation – assuming $M_{c_{0,1}}$ and $M_{c_{0,2}}$ can be any module type:
  - $T^I = \{t : \varrho_{t\to e}(t) \in \mathcal{E}_{IA}(M_{c_{0,1}}) \cap \mathcal{E}_{IA}(M_{c_{0,2}})\}$, and
  - $P^I = \{p : \varrho_{p\to s}(p) \in \bullet(\mathcal{E}_{IA}(M_{c_{0,1}}) \cap \mathcal{E}_{IA}(M_{c_{0,2}})) \cup (\mathcal{E}_{IA}(M_{c_{0,1}}) \cap \mathcal{E}_{IA}(M_{c_{0,2}}))\bullet\}$;

Running Example
Transitions and places describing the molecular mechanism in the interface network, shared by the modules

- kinase protein M_{pK} and transcription factor protein M_{pTF} are $T^I = \{$*pK_pTF_t*1$\}$ and $P^I = \{$*pTF_Y*, *pTF_Y_p*, *pK_CD_act*$\}$, see Figure 3(B) nodes highlighted in blue; and

– kinase gene M_{gK} and transcription factor protein M_{pTF} are $T^I = \{gK\_pTF\_t1, gK\_pTF\_t2\}$ and $P^I = \{pTF\_BS, gK\_E, gK\_E\_\_pTF\_BS\}$, see Figure 3(B) nodes highlighted in red. In general, interface networks of protein modules do not include events affecting the transcriptional activity of a gene. Thus, transition gK_pTF_t3 is not part of the interface network of the kinase gene M_{gK} and transcription factor protein M_{pTF}.

- transcription – assuming $M_{c_{0,1}}$ is a gene module and $M_{c_{0,2}}$ is an mRNA module:
  - $T^I = \emptyset$, and
  - $P^I = \{p : \varrho_{p\to s}(p) \in \mathcal{S}(M_{c_{0,1}}) \wedge \varrho_{p\to s}(p)$ is a transcriptionally active state of $c_{0,2}\}$;

Running Example
Places in the interface network descring part of the transcription shared by the modules kinase gene M_{gK} and kinase mRNA M_{mK} are $P^I = \{gK\_A\_act\}$, see Figure 3(B) nodes highlighted in orange.

- translation – assuming $M_{c_{0,1}}$ is an mRNA module and $M_{c_{0,2}}$ is a protein module:
  - $T^I = \emptyset$, and
  - $P^I = \{p : \varrho_{p\to s}(p) \in \mathcal{S}_0(M_{c_{0,2}})\}$;

Running Example
Places in the interface network describing part of the translation shared by the modules of kinase mRNA M_{mK} and kinase protein M_{pK} are $P^I = \{pK\_CD\_inact\}$, see Figure 3(B) nodes highlighted in green.

- degradation – assuming $M_{c_{0,1}}$ is a protein module and $M_{c_{0,2}}$ is a protein degradation module:
  - $T^I = \emptyset$, and
  - $P^I \subseteq \{p : \varrho_{p\to s}(p) \in \mathcal{S}(M_{c_{0,1}})\}$.

Running Example
Places in the interface network describing part of the degradation shared by the modules of kinase protein M_{pK} and kinase protein degradation M_{dK} are $P^I = \{pK\_CD\_inact, pK\_CD\_act\}$, see Figure 3(B) nodes highlighted in yellow.

The redundant interface networks shared among modules might appear unnecessarily complicated, but they are of tremendous benefits for modules with complex interaction sites by securing the correct functioning of modules in a composed model. Even more, interface networks ensure that interactions can only be executed if all modules of components involved in the interaction are part of the composed model according to the real-world scenario, see next section.

4 Model Composition

The model composition relies on the interface networks introduced in Section 3.4. A composed model is defined by the set of modules $G = \{M_1, \dots, M_{n_M}\}$, $n_M \geq 1$, which can also be written as $G = G_g \cup G_m \cup G_p \cup G_d \cup G_{ai} \cup G_{ci}$, where:

- $G_g = \{M_{g,c_{0,1}}, \dots, M_{g,c_{0,n_g}}\}$ – subset of gene modules,
- $G_m = \{M_{m,c_{0,1}}, \dots, M_{m,c_{0,n_m}}\}$ – subset of mRNA modules,
- $G_p = \{M_{p,c_{0,1}}, \dots, M_{p,c_{0,n_p}}\}$ – subset of protein modules,
- $G_d = \{M_{d,c_{0,1}}, \dots, M_{d,c_{0,n_d}}\}$ – subset of protein degradation modules,
- $G_{ai} = \{M_{ai,c_{0,1}}, \dots, M_{ai,c_{0,n_{ai}}}\}$ – subset of allelic influence modules, and
- $G_{ci} = \{M_{ci,c_{0,1}}, \dots, M_{ci,c_{0,n_{ci}}}\}$ – subset of causal influence modules.

All module subsets G_g, G_m, G_p, G_d, G_{ai}, and G_{ci} are pairwise disjunctive.
The PN of the composed model $\mathcal{N}(G) = (P^G, T^G, F^G, f^G, v^G, m_0^G)$ is given by:

- Set of places $P^G = \bigcup_{M_{c_{0,i}} \in G} P_{M_{c_{0,i}}}$;
- Set of transitions $T^G = \bigcup_{M_{c_{0,i}} \in G} T_{M_{c_{0,i}}}$;
- Set of arcs $F^G = \bigcup_{M_{c_{0,i}} \in G} F_{M_{c_{0,i}}}$;
- Arc weights f^G: $F^G \to \mathbb{N}_0$
  - $f^G \in F^G$, where $f^G \in F^{M_1}, \ldots, F^{M_m}$ with $\{M_1, \ldots, M_m\} \subseteq G$: $f^G = f^{M_1} = \ldots = f^{M_m}$;
- Firing rates v^G: $T^G \to H^G$, $H^G = \bigcup_{M_{c_{0,i}} \in G} H_{M_{c_{0,i}}}$;
- Initial marking: $m_0^G : P^G \to \mathbb{N}_0$, where
  - $\forall p \in P^G$ with $\varrho_{p \to s}(p) \in \bigcup_{M \in G} \mathcal{S}_0(M)$: $m_0^G(p) = n_{c_0}$ with $c_0 = \lambda^c(\varrho_{p \to s}(p))$, and
  - $\forall p \in P^G$ with $\varrho_{p \to s}(p) \notin \bigcup_{M \in G} \mathcal{S}_0(M)$: $m_0^G(p) = 0$.

Running Example
See Figure 3(B) and the caption for the Petri net representation of the composed model of the running example in Figure 3(A). The composed model is also provide as Snoopy-file in Supplementary Material 2.

The definition of interface networks within modules also allows integrating various level of granularity into a composed model. Modules can be recombined arbitrarily to compose alternative models or reused in the context of a different biological system. The arbitrary recombination of modules enables the user to decide about which omic levels to include in a composed model.

Composed models can be submitted to an analysis framework using the PN tools Snoopy [18], Charlie [20] or Marcie [21]. Snoopy [18] allows converting the composed model into an SBML file to employ other analysis tools of choice as well.

5 Features

Based on the molecule-centred modularisation and model composition concept, we developed several extensions to enhance the versatility and usefulness of *BMK*, and thus to accelerate its usage and potential fields of application. Below, we briefly summarise important *BMK* features and provide references for further details.

5.1 Module Annotation

As discussed by Le Novère et al. [31] most published biological models are lost due missing access or insufficient characterisation. Based on these observations Le Novère et al. proposed a guideline for curating and encoding models called *MIRIAM* (Minimum Information Requested In the Annotation of Models). To adhere to the proposed MIRIAM guideline, a module consists not only of its underlying PN but also of an annotation file. We, therefore, defined the *BMK markup language* (*BMKml*), the *XML Schema Definition* (*XSD*) is documented in [32]. In summary, the annotation file specifies:

- module name, type and a cross-reference to the *Ensembl database* [34] of the corresponding component;
- information to clarify the authorship including contact details, creation and modification dates of the module, as well as a short description of the considered molecular mechanisms; and
- for each place, transition and parameter a description, literature references and cross-references to other databases giving more detail about the modelled molecular mechanisms or linking a *GO annotation* [35].

The module annotation provides a complete characterisation and documentation of the module to ease the understanding of the modelled molecular mechanisms and the reuse in composed models.

5.2 Module Construction

So far, we exploit three ways to obtain modules, compare Figure 2 – Box 1:

i. The direct engineering of modules in Snoopy [18] based on knowledge provided in textbooks, publications, and databases is the default approach to construct modules. We used this approach to compose models for the following case studies:

   - Pain signalling – the composed model comprises 38 modules of key players involved in molecular mechanisms of nociception, which occurs in the peripheral terminals of dorsal root ganglion neurones and is responsible for the detection of noxious and painful stimuli [36], [37], [38], [39], [40].
   - IL6-induced JAK-STAT signalling – the composed model comprises seven protein modules (JAK, STAT, IL6, IL6R, gp130, SHP2, SOCS3), as well as a protein degradation, mRNA and gene module for the feedback inhibitor SOCS3 [25], [41].
   - Phosphate regulatory network in enterobacteria – the composed model comprises nine protein modules (PhoA, PstS, PstA, PstB, PStC, PhoU, PhoR, PhoA), as well as a protein degradation, mRNA and gene module for PhoA, which sense and regulate the external Pi [33].

ii. The reverse engineering of modules from complex (*omic*) datasets to obtain causal modules or map data to module prototypes [32], [33], e.g. as in case of conceptualised gene or mRNA modules, are straightforward approaches to integrate experimental results directly into composed models and thus, to test for their effect on the model's behaviour.

iii. The modularisation and thus, the integration of existing models will be another valuable resource to obtain modules. In particular, we suggest a routine to decompose mechanistic SBML encoded models [23] into a set of related modules. Modules can also be obtained from gene regulatory models defined by Boolean networks as described in [22].

5.3 Algorithmic Mutation

Life science showed in order to understand a biological system it is necessary to study not only the native system state but also its genetic variations. Genetic variations are a consequence of mutations in the genome, where some of the mutations are indifferent, and others lead to a gain or even a loss of function. Mutations result either naturally or are experimentally enforced. Modelling approaches need to introduce equivalent mutation workflows. *BMK* suggests *algorithmic mutation* based on the modularisation concept to conveniently mimic gene deletions and structure-function mutations, which are described in detail in [32].

Removing modules from a composed model mimics gene deletions. This mutation can be performed as single, double or even multiple gene knockouts to sufficiently block the effect of redundant gene functions.

Since functional units in a module have a relation to the structure of the modelled component, their alterations can mimic structure-function mutation. By deleting the transitions in the subgraph of a functional unit, the functional unit can only adopt a subset of the previously reachable molecular states and can therefore not execute its complete functionality. As stated above, such alteration could result in an indifferent effect, but it could also possibly increase or decrease a specific function encoded by the molecular mechanism.

Both mutation algorithms can be performed either on the entire set of modules or on a subset of modules/-functional units that can be filtered through the provided references in the annotation file, see Section 5.1. The systematic mutation of a composed model yields a large set of alternative models and demands analysis workflows based on machine learning to reveal fundamental insights and identify structures with a significant effect on the system's behaviour.

5.4 Spatiotemporal Transformation

Molecular mechanisms are affected by the spatial distribution and movement of the involved components. Therefore, the cellular arrangement concerning compartments, membranes and pools, as well as the cell geometry and size, have to be considered to understand the spatiotemporal behaviour of a system. Representing biomolecular mechanisms and movement in one coherent model is a challenging task.

Based on the modularisation concept, we propose a *spatiotemporal transformation algorithm* that integrates those spatial aspects into the composed model, see [24], [42] for details. The spatiotemporal transformation algorithm extends the composed model with a spatial model controlling the movement and interaction of the

involved components without interfering the modelled molecular mechanisms. In the first step of the spatiotemporal transformation, the cellular space needs to be defined by a grid, as well as cellular substructures on that particular grid. This step includes the assignment of the components in the composed model to a set of substructures. The position of each component on the grid is encoded by a set of coordinate places (one, two or three places depending on the dimension of the grid) and the corresponding marking. The movement of the component is realised by increasing or decreasing the marking on the coordinate places by transitions with respect to the grid and its substructures. The introduced subnets responsible for the movement must further distinguish, whether the component is in an interactive state forming a complex or not, to guarantee a synchronous movement of all components forming an interactive complex. Additional rules have to be introduced to ensure that interaction between components encoded in the interface networks can only occur if they reside close to each other. The spatiotemporal transformation employs coloured PNs to allow for a scalable representation of the spatial model concerning the copy number of the involved components. The copy number of components has to be realised via colour and not via the marking, to distinguish the individual copies in respect to their position, movement and interaction.

Applying the spatiotemporal transformation to a composed model allows studying how space might affect the dynamic behaviour of a system without constituting a new additional model or framework only for this specific purpose.

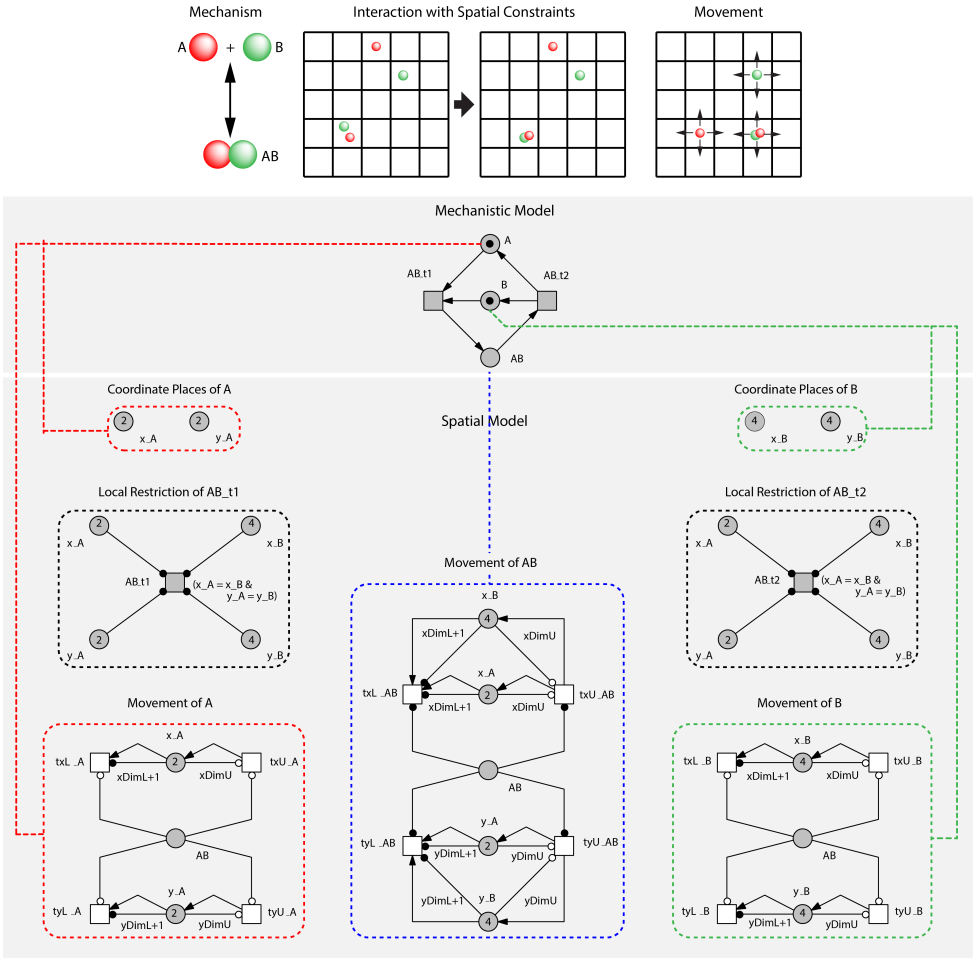

Figure 4: Basics of the spatial transformation concept.

Running Example

In the example given in Figure 4, the mechanistic model describes the binding of A and B to form complex AB, and its dissociation. The spatial model provides places encoding the (x, y)-coordinates of A and B. The marking of those places reflects the position of A and B on a 2-dimensional grid, which is defined by the parameters $(xDimL, xDimU)$ and $(yDimL, yDimU)$ (lower/upper bound at the x/y-axis). Also, the spatial model encodes spatial constraints to restrict interactions among A and B locally. Therefore, the transitions AB_{t1} and AB_{t2} are connected with the coordinates places by read arcs. The firing rates are multiplied by a Boolean expression defining a neighbourhood condition, which evaluates whether A and B are at the same position. Only if A and B stay at the same position, the interaction can be executed in the model. In the spatial model, A and B can either move as single entities or as complex AB. Thus, they can either move towards the lower bound of

an axis ($t\{x, y\}$ *L_*$\{A, B, AB\}$) or the upper bound of an axis ($t\{x, y\}$ *U_*$\{A, B, AB\}$). The marking of the coordinate place must always be greater than $xDimL/yDimL$, to move towards the lower bound of an axis. This is checked by the read arcs with the arc-weight $xDimL + 1/yDimL + 1$. Also, the marking of the coordinate place must always be less or equal than $xDimU/yDimU$, to move towards the upper bound of an axis. This is checked by the inhibitory arcs with the arc-weights $xDimU/yDimU$. In addition, A and B are only allowed to move as a single entity if the place AB is empty. This is checked by the inhibitory arcs connecting the place AB with transitions $t\{x, y\}$ *L_*$\{A, B\}$ and $t\{x, y\}$ *U_*$\{A, B\}$. In contrast, to move A and B as a complex, place AB must not be empty. This is checked by the read arcs connecting place AB with transitions $t\{x, y\}$ *L_AB* and $t\{x, y\}$*U_AB*.

6 BioModelKit Web Interface and Database

The *BMK web interface* (*BMKwi*, www.biomodelkit.org) gives public access to the modules by storing and linking all elements of the underlying PN graph and all annotations in the *BMK database* (*BMKdb*), see Figure 2 – Box 3. Thus, *BMK* is more than just a repository of hard-coded models.

BMKdb uses *MySQL* as relational database management system. The database scheme has been derived from the XSD scheme of the Snoopy file format [18] used to save the PN graphs of a module and the *BMKml* format to store the module annotation, see Section 5.1. The content of the PN graph and the annotation file of a module are not only stored as hard copies on the *BMK* file server, but are also transferred as records to the *BMKdb*, where all PN graph elements and information in the annotation file are explicitly stored and linked to each other, see Figure 5. The database scheme and its relation to the introduced module terminology and annotations are described in [32]. Storing modules in *BMKdb* allows not only organising submitted modules but also keeping track of module versioning. Therefore, the *BMKdb* can provide various module versions for one component that might display different levels of granularity, hypotheses on the modelled molecular mechanisms, or refinements and updates due to new insights.

BMKwi is running on a *Linux/Apache 2.2.14 web server* with *MySQL client version 5.1.67*, *PHP 5.2.0*, and *Javascript 1.8.5*. Its main functionality is to publish modules. Therefore, users can browse, search and inspect modules, the corresponding content is automatically generated by querying the *BMKdb*. Users registered at *BMK* can also manage their private profiles, which includes creating module collections. Module collections can be filled with *ad hoc* chosen modules published at *BMK*. Later on, modules stored in a private collection can be used for model composition, algorithmic model mutation or spatiotemporal transformation which are available as automated features. Registered users can also curate and submit modules to *BMK*, which will be available after successful approval. Figure 5 summaries the structure of the web-interface and possible user interactions.

The simple click-and-run principle makes it easy for every user to compose complex models and apply the implemented features, see the movie in Supplementary Material 3 for a demonstration.

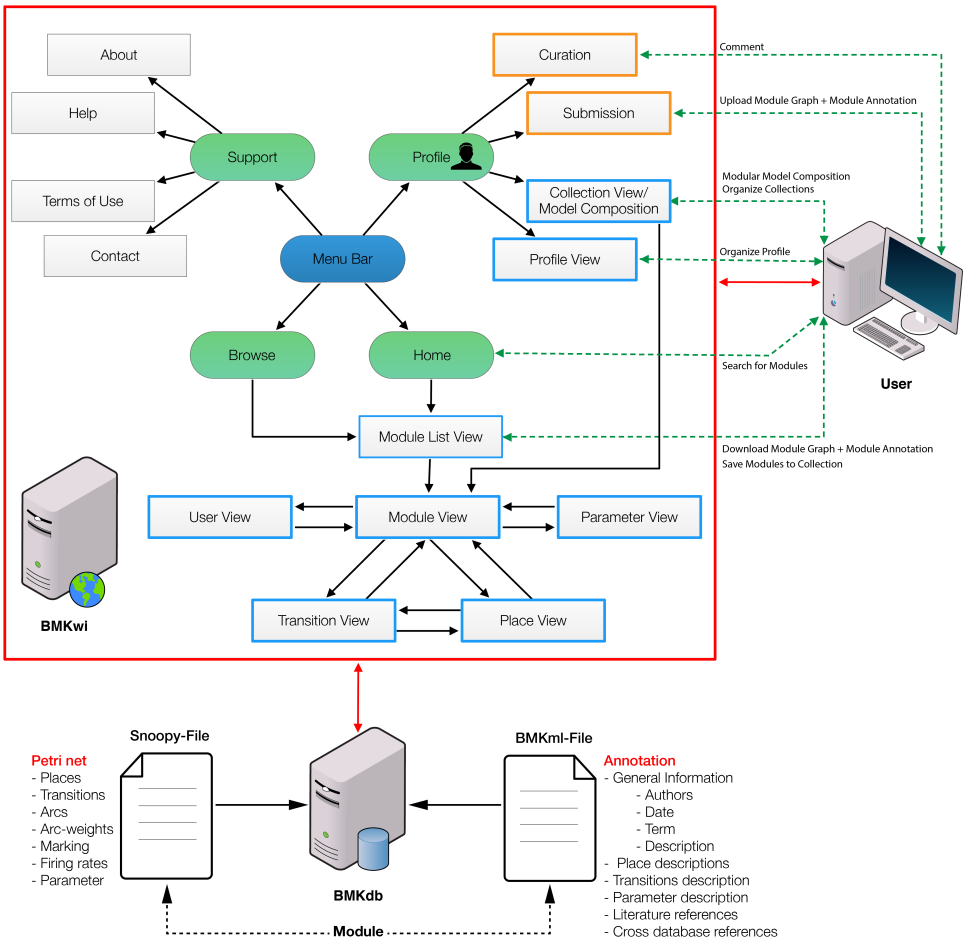

Figure 5: BioModelKit database and navigation scheme of the web-interface with features available for user interaction.

7 Discussion and Outlook

With *BMK* we propose a molecule-centred modularisation concept which is geared towards the *ab inito* design of modules, describing the functionality and interactions of one particular molecule by a PN that includes interface networks to automate the model composition. We showed that modules can be constructed from various data resources using direct and reverse engineering approaches as well as by decomposing existing models, which allows integrating molecular mechanisms and causal correlations on the metabolic, proteomic, transcriptomic and genomic level. In the scope of *BMK*, we set-up a MySQL database, that stores and organises the PN graph and all information provided in the MIRIAM-compliant annotation file of each module, which allows keeping track of the module versioning. We further designed a web interface to access modules stored in *BMKdb*. The *BMKwi* enables the user to perform the composition of models based on an *ad hoc* chosen set of modules. The model composition can be combined with mutation algorithms to generate alternative models accordingly, which allows mimicking naturally occurring genetic variations and their effect *in silico*. *BMK* also allows transforming a composed model into a spatiotemporal model integrating information about the cell geometry and arrangement concerning compartments, membranes and pools.

We demonstrated the practicability and versatility of the concept within *BMK* on several case studies, like JAK-STAT signalling [25], [41], molecular mechanisms of nociception [36], [37], [38], [39], [40] and phosphate regulation in enterobacteria [33]; additional case studies are currently in progress. So far, the *BMKdb* holds only the modules of the JAK-STAT signalling case studies, the modules of the other two case studies will be made available with the next release and update of the *BMKwi* and *BMKdb*.

BMK is still under continuous development; we aim to enhance the functionality of the web interface according to user priorities. Soon, we will release modules constructed in so far unpublished case studies, as well as modules obtained by the modularisation of existing models, to increase the number of available models in *BMK*. We intend to develop a *BMK* ontology, which addresses the naming convention of model elements (reactions, species, parameters etc.) to allow for a more consistent naming among modules stored in the *BMK*. Based

on the *BMK* ontology, we plan to develop a module questionnaire that allows for a straightforward translation of molecular mechanisms into modules compliant to the *BMK* standards. Future developments of the *BMKwi* will also focus on the implementation of modularisation workflows for automatic *omic* data integration, which facilitates the user to directly test for the effect of the experimental results on the model's behaviour.

We see great potential for *BMK* for in-depth *in silico* mutation studies to identify models reproducing a desired behaviour. This approach can considerably supplement experimental work. Another exciting field for *BMK* is *in silico* synthetic biology; module interface networks can easily be altered to add or remove functionality, which allows rewiring the network structure of the composed model synthetically and thus, to analyse the effect of desired properties. Last but not least, individualising modules or conceptualising composed models based on *omic* data might help to obtain new insights about the underlying molecular circumstances and to predict strategies to improve the molecular mechanisms. Such workflows are of particular interest in precision medicine to integrate patient-specific data and to suggest potential therapeutic strategies to treat clinical disorders or in precision crop plant breeding to recommend suitable genotypes, genetic variations or breeding strategies to increase the yield and yield stability, as well as the nutrition value.

BMK is well suited to study complex and diverse biological systems, to obtain new insights about the involved molecular mechanisms and based on that to make predictions and rise hypothesis supporting and benefiting experimental work in multilateral collaborations.

Acknowledgement

We would like to thank Monika Heiner, Mostafa Herajy, Fei Liu, Christian Rohr, and Martin Schwarick for their contributions in developing and supporting the use of Snoopy, Marcie, and Charlie; and Wolfgang Marwan for supporting the development of *BMK* by his supervision. We appreciate countless productive discussions with all of them.

Funding

This work has been partly supported by the Germany Federal Ministry of Education and Research, Funder Id: 10.13039/501100002347 (FKZ0315449F, FKZ0316177D).

Conflict of Interest Statement: All authors have read the journal's Publication ethics and publication malpractice statement available at the journal's website and hereby confirm that they comply with all its parts applicable to the present scientific work.

References

[1] Koch I, Reisig W, Schreiber F. Modeling in systems biology. The Petri Net Approach. Springer Science & Business Media, 2010.
[2] Somekh J, Peleg M, Eran A, Koren I, Feiglin A, Demishtein A, et al. A model-driven methodology for exploring complex disease comorbidities applied to autism spectrum disorder and inflammatory bowel disease. J Biomed Inform. 2016;63:366–78.
[3] Xu H, Curtis TY, Powers SJ, Raffan S, Gao R, Huang J, et al. Genomic, biochemical, and modeling analyses of asparagine synthetases from wheat. Front Plant Sci. 2018;8:2013.
[4] Zechendorf E, Vaßen P, Zhang J, Hallawa A, Martincuks A, Krenkel O, et al. Heparan sulfate induces necroptosis in murine cardiomyocytes: A medical-in silico approach combining in vitro experiments and machine learning. Front Immunol. 2018;9:885.
[5] Baldan P, Cocco N, Marin A, Simeoni M. Petri nets for modelling metabolic pathways: a survey. Nat Comput. 2010;9:955–89.
[6] Chaouiya C. Petri net modelling of biological networks. Brief Bioinform. 2007;8:210–9.
[7] Heiner M, Koch I. Petri net based model validation in systems biology. In: Proc. ICATPN 2004. vol. 3099 of LNCS. Springer, 2004:216–37.
[8] Sackmann A, Heiner M, Koch I. Application of petri net based analysis techniques to signal transduction pathways. BMC Bioinformatics. 2006;7:482.
[9] Heiner M. Understanding network behaviour by structured representations of transition invariants – a petri net perspective on systems and synthetic biology. In: Condon A, Harel D, Kok J, Salomaa A, Winfree E, editors. *Algorithmic Bioprocesses*, Natural Computing Series. Berlin, Heidelberg: Springer, 2009. p. 367–89. Available from: http://www.springerlink.com/content/m8t30720r141442m.
[10] Zevedei-Oancea I, Schuster S. Topological analysis of metabolic networks based on petri net theory. In Silico Biology. 2003;3:323–45.
[11] Koch I. Petri nets in systems biology. SoSyM. 2014;14:703–10.
[12] Blätke MA, Heiner M, Marwan W. Chapter 7 – BioModel Engineering with Petri Nets. In: Algebraic and Discrete Mathematical Methods for Modern Biology. Boston: Elsevier Inc., 2015:141–93.

[13] Gilbert D, Heiner M, Lehrack S. A unifying framework for modelling and analysing biochemical pathways using petri nets. In: Computational Methods in Systems Biology. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007:200–16.
[14] Jensen K, Kirstensen LM. Coloured petri nets: modelling and validation of concurrent systems. Berlin Heidelberg: Springer-Verlag; 2009.
[15] Gao Q, Gilbert D, Heiner M, Liu F, Maccagnola D, Tree D. Multiscale modelling and analysis of planar cell polarity in the drosophila wing. IEEE/ACM Trans Comput Biol Bioinform. 2013;10:337–51.
[16] Liu F, Heiner M, Gilbert D. Coloured petri nets for multi-level, multiscale, and multi-dimensional modelling of biological systems. Brief Bioinform. 2017;bbx150.
[17] Rohr C, Marwan W, Heiner M. Snoopy – a unifying petri net framework to investigate biomolecular networks. Bioinformatics (Oxford, England). 2010;26:974–5.
[18] Heiner M, Herajy M, Liu F, Rohr C, Schwarick M. Snoopy – A unifying Petri net tool. In: Proc. PETRI NETS 2012. vol. 7347 of LNCS. Springer, 2012:398–407.
[19] Marwan W, Rohr C, Heiner M. 2012. Petri Nets in Snoopy: A Unifying Framework for the Graphical Display, Computational Modelling, and Simulation of Bacterial Regulatory Networks. In: van Helden J, Toussaint A, Thieffry D, editors. Bacterial Molecular Networks. Methods in Molecular Biology (Methods and Protocols), vol 804. Springer, New York, NY. p. 409–37.
[20] Heiner M, Schwarick M, Wegener J. Charlie – an extensible Petri net analysis tool. In: Devillers R, Valmari A, editors. Proc. PETRI NETS 2015. vol. 9115 of LNCS. Springer, 2015:200–11.
[21] Heiner M, Rohr C, Schwarick M. MARCIE – model Checking and reachability analysis done effiCIEntly. In: Colom J, Desel J, editor(s). Proc. PETRI NETS 2013. Vol. 7927 of LNCS. Berlin, Heidelberg: Springer, 2013:389–99.
[22] Jehrke L. Modulare modellierung und graphische darstellung boolescher netzwerke mit hilfe automatisch erzeugter Petri-netze und ihre simulation am beispiel eines genregulatorischen netzwerkes [Masterthesis]; 2014.
[23] Soldmann M. Transformation monolithischer SBML-modelle biomolekularer netzwerke in Petri netz module [Masterthesis]; 2014.
[24] Blätke MA, Rohr C. BioModelKit: spatial modelling of complex multiscale molecular biosystems based on modular models. In: Advances in Biological processes and Petri nets (BioPPN). vol. 160, 1-2 of Fundamenta Informaticae. IOS Press, 2018:221–54.
[25] Blätke MA, Dittrich A, Rohr C, Heiner M, Schaper F, Marwan W. JAK/STAT signalling – an executable model assembled from molecule-centred modules demonstrating a module-oriented database concept for systems and synthetic biology. Mol Biosyst. 2013;9:1290–307.
[26] Cooling MT, Rouilly V, Misirli G, Lawson JR, Yu T, Hallinan J, et al. Standard virtual biological parts: a repository of modular modeling components for synthetic biology. Bioinformatics (Oxford, England). 2010;26:925–31.
[27] Lloyd CM, Lawson JR, Hunter PJ, Nielsen PMF. The cellML model repository. Bioinformatics (Oxford, England). 2008;24:2122–3.
[28] Li C, Donizelli M, Rodriguez N, Dharuri H, Endler L, Chelliah V, et al. BioModels database: an enhanced, curated and annotated resource for published quantitative kinetic models. BMC Syst Biol. 2010;4:1.
[29] King ZA, Lu J, Dräger A, Miller P, Federowicz S, Lerman JA, et al. BiGG models: a platform for integrating, standardizing and sharing genome-scale models. Nucleic Acids Res. 2016;44(D1):D515–22.
[30] Zhu Q, Wong AK, Krishnan A, Aure MR, Tadych A, Zhang R, et al. Targeted exploration and analysis of large cross-platform human transcriptomic compendia. Nat Methods. 2015;12:211–4.
[31] Le Novère N, Finney A, Hucka M, Bhalla DUS, Campagne F, Collado-Vides J, et al. Minimum information requested in the annotation of biochemical models (MIRIAM). Nat Biotechnol. 2005;23:1509–15.
[32] Blätke MA. BioModelKit a framework for modular biomodel-engineering. [Phd Thesis]; 2017.
[33] Blätke MA, Heiner M, Marwan W. Predicting phenotype from genotype through automatically composed petri nets. In: Proc. 10th International Conference on Computational Methods in Systems Biology (CMSB 2012), London. vol. 7605 of LNCS/LNBI. Springer, 2012:87–106.
[34] Flicek P, Aken BL, Beal K, Ballester B, Caccamo M, Chen Y, et al. Ensembl 2008. Nucleic Acids Res. 2008;36(Database issue):D707–14.
[35] Gene Ontology Consortium. The Gene Ontology Project in 2008. Nucleic Acids Res. 2008;36(Database issue):D440–4.
[36] Blätke MA. Petri-netz modellierung mittels eines modularen und hierarchischen ansatzes mit anwendung auf nozizeptive signalkomponenten. [Diploma Thesis]; 2010.
[37] Blätke MA, Meyer S, Stein C, Marwan W. Petri net modeling via a modular and hierarchical approach applied to nociception. In: Proc. 1st Int. Workshop on Biological Processes & Petri Nets (BioPPN), satellite event of Petri Nets 2010;2010:135–46.
[38] Blätke MA, Marwan W. Modular and hierarchical modelling concept for large biological Petri nets applied to nociception. In: Proc. 17th German Workshop on Algorithms and Tools for Petri Nets (AWPN 2010). vol. 643 of CEUR Workshop Proceedings. CEUR-WS.org, 2010:42–50.
[39] Blätke MA, Meyer S, Marwan W. Pain signaling – a case study of the modular Petri net modeling concept with prospect to a protein-oriented modeling platform. In: Proc. 2nd International Workshop on Biological Processes & Petri Nets (BioPPN), satellite event of Petri NETS 2011. vol. 724 of CEUR Workshop Proceedings. CEUR-WS.org, 2011:117–34.
[40] Blätke MA, Marwan W. A database-supported modular modelling platform for systems and synthetic biology. In: Proc. 3rd International Workshop on Biological Processes & Petri Nets (BioPPN), satellite event of Petri NETS 2012. vol. 852 of CEUR Workshop Proceedings. CEUR-WS.org, 2012:18–19.
[41] Blätke MA, Dittrich A, Rohr C, Heiner M, Schaper F, Marwan W. JAK–STAT signalling as example for a database-supported modular modelling concept. In: Proc. 10th International Conference on Computational Methods in Systems Biology (CMSB 2012), London. vol. 7605 of LNCS/LNBI. Springer, 2012:362–5.
[42] Blätke MA, Rohr C. A Coloured Petri net approach for spatial biomodel engineering based on the modular model composition framework Biomodelkit. In: Proc. 6th Int. Workshop on Biological Processes & Petri Nets (BioPPN 2015), satellite event of Petri Nets 2015. vol. 1373 of CEUR Workshop Proceedings. CEUR-WS.org, 2015:37–54.
[43] Liu F, Blätke M, Heiner M, Yang M. Modelling and simulating reaction–diffusion systems using coloured Petri nets. Comput Biol Med. 2014;53:297–308.
[44] Pârvu O, Gilbert D, Heiner M, Liu F, Saunders N, Shaw S. Spatial-temporal modelling and analysis of bacterial colonies with phase variable genes. ACM Trans Model Comput Simul. 2015;25:13–25.

[45] Liu F, Heiner M. Multiscale modelling of coupled Ca^{2+} channels using coloured stochastic Petri nets. IET Syst Biol. 2013;7:106–13.
[46] Blätke MA, Rohr C, Heiner M, Marwan W. A Petri-net-based framework for biomodel engineering. In: Large-Scale Networks in Engineering and Life Sciences. Modeling and Simulation in Science, Engineering and Technology. Cham: Springer International Publishing, 2014:317–66.

Supplementary Material: The online version of this article offers supplementary material (DOI: https://doi.org/10.1515/jib-2018-0021).