Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 326 (2016) 27–49

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Formal Biochemical Space with Semantics in Kappa and BNGL[1]

T. Děd, D. Šafránek, M. Troják, M. Klement, J. Šalagovič, L. Brim

Faculty of Informatics, Masaryk University
Brno, Czech Republic

Abstract

Biochemical Space (BCS) has been introduced as a semi-formal notation for reaction networks of biological processes. It provides a concise mapping of mathematical models to their biological description established at a desired level of abstraction. In this paper, we first turn BCS into a completely formal language with rigorously defined semantics by means of a simplified Kappa calculus. On the practical end, we support BCS with translation to BNGL, a well-known practically used rule-based language. Finally, we show the current status of BCS defined for cyanobacteria processes.

Keywords: Kappa, Biochemical space, CMP, cyanobacteria processes

1 Introduction

To provide a rigorous representation of complex biological processes without congesting the users with overcomplicated syntax, we have enriched our online platform for modelling of cyanobacteria processes, e-cyanobacterium[2], with a semi-formal textual notation called *Biochemical Space* (BCS) [10]. BCS represents reaction networks of the studied processes and provides a concise mapping of mathematical models to a precise biological description that is established at a consortium-agreed level of abstraction.

The concept of BCS makes a crucial methodological part of *Comprehensive Modelling Platform* (CMP), a general platform for computational modelling and analysis of biological processes, first introduced in [15] as a concept for unambiguous representation of internally consistent reduced mathematical models of oxygenic photosynthesis [17] and further refined to a general online modelling platform as described in [11]. In general, the main goal of BCS as a part of CMP is to simplify

[1] This work has been supported by the Czech Science Foundation grant No. GA15-11089S.

[2] http://www.e-cyanobacterium.org, http://www.cyanoteam.org

http://dx.doi.org/10.1016/j.entcs.2016.09.017

systems level model-building tasks by providing simple and clear way of notation easily understandable by *in silico* modellers on the one end, and experimental biologists on the other end.

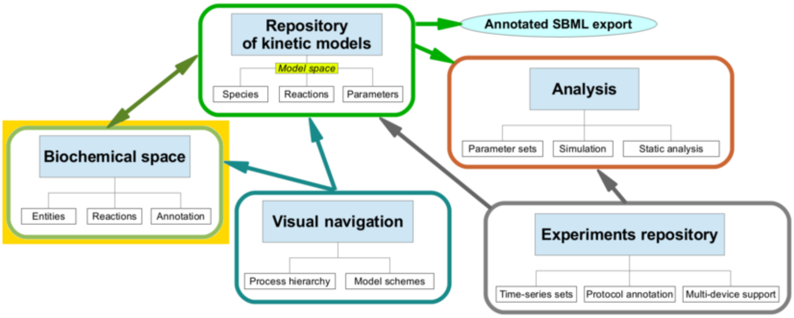

Fig. 1. Graphical representation of Comprehensive Modelling Platform (CMP).

In [1] we have shown that rule-based methods can be directly used for rewriting existing kinetic models of oxygenic photosynthesis into a compact non-redundant form obtained by applying a set of automatised syntactic reductions defined in Kappa [4]. That achievement lead us further to employ rule-based definition of biological processes as the framework for qualitative description of the consortium-agreed understanding of chemical reactions behind the processes. Existing quantitative models can be then mapped onto the qualitative rule-based BCS.

BCS borrows concepts from two worlds – the formal rule-based languages and semi-formal reaction network annotation bases such as KEGG [9]. The BCS language is defined with a clear relation to BNGL [6], a practical tool-supported rule-based language compatible with Kappa. Since the most important requirement of the consortium-driven modelling platform is a simple-to-use format well-adjusted to the suitable level of abstraction employed for biological process description, we were not able to directly employ any of the well-established rule-based languages and rather defined a new language with a clear relation to the existing formats.

In particular, for our purpose BNGL and Kappa consider too many details. The most important fact is that BNGL requires to specify bindings inside the complex structures. This demands binding sites specification for each molecule and unique labelling for each interaction. In BCS, these structural details are abstracted out. It is enough just to know that molecules interact and form a complex while abstracting from the details. Another issue is the fact that existing formalisms consider biological entities as agents all defined at the same level of abstraction. In BCS we allow hierarchical construction of agents from simple molecules to composite structures and complexes. Finally, the algebraic representation of Kappa and BNGL goes quite far from common chemical notation and is not human readable. BCS attempts to avoid this.

In [10] we have presented general ideas behind BCS. The language has been defined as a semi-formal notation. In this paper, we turn BCS into a completely formal language with clearly defined syntax and semantics (by embedding to Kappa). We define the relation of BCS and BNGL which allows us to translate specification between both languages. In Section 7, we show the current status of BCS description implemented for cyanobacteria.

1.1 Related Work

On the bioinformatics side, the closest format to BCS is KEGG [9]. In contrast to BCS, KEGG does not support rule-based description allowing compact representation of combinatorial states. Moreover, it does not support logical organization of entities and reactions into an organism-specific hierarchy that may significantly simplify understanding of the complex processes driving the organisms physiology and its interaction with the environment. Since the notation relies on a simple textual base and focuses on a simple but still reasonably precise and compact description maintainable by biologists, the format of BCS specification is compliant with KEGG.

BCS should be also compared to the well-acclaimed standard provided by SBML [8,13] that might be also used for representation of a biochemical space. BCS completely avoids issues related with dynamical models. As an annotation platform purely focused on process-level description, BCS goes beyond SBML level 2 in generalization of entities to hierarchical agents, in introducing entity states, and in dealing with related combinatorial explosion. These issues are solved in detail by rule-based approaches [4,7] and there is a draft of a package for SBML level 3 in preparation [18] (multi).

In comparison with process algebraic languages treating chemical reactions mechanistically as communicating concurrent processes [2,5], BCS keeps a purely qualitative level of description closed to chemical reactions and remains as simple as possible to cover the consortium-agreed level of abstraction. The language defined in [16] targets a similar level of abstraction as BCS. However, it is intended more as a programming language for biological systems than an annotation format.

2 Background

We define simplified Kappa (*kappa*$_s$) using a process-like notation as is presented in [3], syntax and the notions of structural equivalence and matching are entirely taken from [3]:

expression	$E ::= \emptyset \mid a, E$	site	$s ::= n_\iota^\lambda$
agent	$a ::= N(\sigma)$	site name	$n ::= x \in \mathcal{S}$
agent name	$N ::= A \in \mathcal{A}$	internal state	$\iota ::= \epsilon \mid m \in \mathbb{V}$
interface	$\sigma ::= \emptyset \mid s, \sigma$	binding state	$\lambda ::= \epsilon \mid i \in \mathbb{N}$

where $\mathcal{A}$ is a finite set of agent names, $\mathcal{S}$ is a finite set of site names, $\mathbb{V}$ is a finite set of values representing modified states of the sites. We use notation $\sigma(a)$ for a signature associated to an agent a.

An agent is denoted by its name and its interface. Interface consists of a sequence of sites. x_ι^λ denotes a site x with internal state ι and binding state λ. If the binding state is ϵ then the site is free, otherwise it is bound. By convention, when a binding or internal site is not specified, ϵ is considered.

Note that full Kappa is richer. It allows a binding state meaning a *free or bound site*, denoted by a question mark. We also omit rates from the rules.

Definition 2.1 *An expression is* well-formed *if a site name occurs only once in an interface and if each binding state ($\neq \epsilon$) present in the expression occurs exactly twice. The* set of all well-formed expressions *is denoted as $\mathcal{E}$.*

We assume a standard structural equivalence on well-formed expressions that treats as equivalent all expressions differing in order of sites in interfaces, order of agents in expression, and naming of binding sites.

A *rule* is a pair of expressions E_l, E_r (usually written as $E_l \to E_r$). The set of all rules is denoted as $\mathcal{R}$. The left hand side E_l of the rule describes the solution taking part in the reaction and the right hand side E_r describes the effects of the rule. The rule can be either a binding rule or a modification rule. A binding (unbinding) rule binds two free sites together (or unbinds two bound sites). A modification rule modifies some internal state [3].

Matching is a relation denoted as $\models \subseteq \mathcal{E} \times \mathcal{E}$ and defined inductively in the left column below. *Replacement* is a function $\mathcal{E} \times \mathcal{E} \to \mathcal{E}$ defined in the right column below:

$$\begin{array}{cc}
n_\iota^\lambda \models n_\iota^\lambda & n_\iota^\lambda[n_{\iota_r}^{\lambda_r}] = n_{\iota_r}^{\lambda_r} \\[1ex]
n_\iota^\lambda \models n^\lambda & n_\iota^\lambda[n^{\lambda_r}] = n_\iota^{\lambda_r} \\[1ex]
\sigma \models \emptyset & \sigma[\emptyset] = \sigma \\[1ex]
\dfrac{s \models s_l \quad \sigma \models \sigma_l}{s, \sigma \models s_l, \sigma_l} & s, \sigma[s_r, \sigma_r] = s[s_r], \sigma[\sigma_r] \\[2ex]
\dfrac{\sigma \models \sigma_l}{N(\sigma) \models N(\sigma_l)} & N(\sigma)[N(\sigma_r)] = N(\sigma[\sigma_r]) \\[2ex]
E \models \emptyset & E[\emptyset] = E \\[1ex]
\dfrac{a \models a_l \quad E \models E_l}{a, E \models a_l, E_l} & (a, E)[a_r, E_r] = a[a_r], E[E_r]
\end{array}$$

A replacement can be applied only if the corresponding matching is satisfied.

In order to apply a rule $E_l \to E_r$ to a solution $[E]$ the expression E representing the solution must first be reordered to an equivalent expression E' that matches E_l (according to the definition of matching stated above). E' is then replaced with $E'[E_r]$ (also defined above).

Rule application is a mapping $t : \mathcal{E} \times \mathcal{R} \to \mathcal{E}$ such that $t([E], (E_l, E_r)) = [E'[E_r]]$ whenever $\exists E' \in [E] : E' \models E_l$. Rules yield a *transition system* between solutions containing an edge $[E] \to_{E_l, E_r} [E'[E_r]]$ whenever $\exists E' \in [E]. E' \models E_l$.

An *agent signature* (Σ, I) is a pair of mappings $\Sigma : \mathcal{A} \to 2^{\mathcal{S}}$ and $I : \mathcal{A} \times \mathcal{S} \to 2^{\mathbb{V}}$. Informally, Σ restricts for each agent name $A \in \mathcal{A}$ the set of site names that can occur in an agent with name A and I restricts the set of internal states a particular site can attain. Additionally, expressions are treated as complete if their agents employ all sites and states of the signature. For formal definitions see [1] or the original paper [3].

A *rule-based model* $\mathcal{M}$ is a tuple $(\Sigma, I, \mathcal{R})$ such that $\mathcal{R}$ satisfies the signature (Σ, I). An *initialised model* $\mathcal{M}_0$ is a pair $(\mathcal{M}, E_i)$ where $\mathcal{M} = (\Sigma, I, \mathcal{R})$ is a rule-based model and E_i is an expression representing the *initial solution* such that E_i is complete for the signature (Σ, I).

Definition 2.2 *A* state space *of an initialised model* $\mathcal{M}_0 = (\mathcal{M}, E_i)$ *is a pair* $(\mathsf{Solutions}(\mathcal{M}_0) \subseteq \mathcal{E}, \mathsf{Reactions}(\mathcal{M}_0) \subseteq \mathcal{E} \times \mathcal{E})$ *defined inductively as follows:*

(i) $[E_i] \in \mathsf{Solutions}(\mathcal{M}_0)$

(ii) $[E] \in \mathsf{Solutions}(\mathcal{M}_0)$ *and* $\exists r \in \mathsf{Rules}(\mathcal{M}).t([E], r) = [E']$
if and only if $[E'] \in \mathsf{Solutions}(\mathcal{M}_0)$ *and* $([E], [E']) \in \mathsf{Reactions}(\mathcal{M}_0)$

In BNGL, agents are called molecules and they are specified in a similar manner as in *kappa*$_s$. An example of a molecule is `A(x~n!1)` where the site `x` has an internal state `n` (separated from the site by a tilde) and a binding state is `1` (separated by the exclamation mark). The BNGL alternatives to agent signatures are called molecule types and they are defined using the notation demonstrated in the following example: `A(x~n~b, y~n~a)`. Here, the allowed internal states of the individual sites are separated by tildes (site `x` can have an internal state `n` or `b`). Rules are described by the *lhs* `->` *rhs* notation (or *lhs* `<->` *rhs* in the case of reversible rules). The individual model components (molecule types, reaction rules, seed species, observables) are in BNGL separated by the *begin* `keyword` and *end* `keyword`.

3 Biochemical Space

BCS provides well described biological background for mathematical models of processes taking place in specific organism. Complete BCS model provides a connection between existing ontologies and partial mathematical models. A BCS model is represented in a form of a textual file. This file offers a human readable format of BCS which can be easily edited in a dedicated editor and visualised on the website. First part of a BCS model is represented by a set of *entities* (to be compliant with process-algebraic frameworks we call entities *agents*), while the second part contains *rules* (abstractly represented chemical reactions defined over the set of entities). In our case study, a consortium of scientists is involved in modelling several cyanobacterial processes and in establishing of the respective BCS model.

When building the BCS model, emphasis is put on well-defined and complete annotations. Therefore, links to relevant ontologies must be specified for each entity and rule. Unique IDs provided by ontologies can help to automatically detect duplicities. IDs are also used to create hypertext links to related ontologies on the web, thus providing a one part of the already mentioned connection between ontologies and models. At this moment, links to KEGG, ChEBI, CyanoBase [14] and other databases are supported. A single entity or a rule can have multiple links to several external databases. An example is presence of a particular entity in ChEBI as well as in KEGG. In the case of annotating enzymatic rules, an EC number (here acting as a descriptor of the rule mechanism behind the catalytic reaction) is associated to the enzyme via a respective KEGG ID. For an entity that represents a protein, annotation can be enriched with a sequence of genes that encode the protein. A single link (in our case to genome browser in CyanoBase) is created for every gene separately. If more than one gene sequence is present, additional information about every particular sequence is specified in terms of notes. In general, NOTES records carry internal information about an entity or a rule. Finally, a comma is used as a separator between records within links and notes fields. In most cases, ontologies contain general information about entities and about rule mechanisms. If this is not available, verbal description of the role of an entity or a rule can be specified directly within the particular record.

Example 3.1 *Description, links, and notes information for an entity.*

DESCRIPTION:	Protein involved in hydrolysis of N-acetylated amino acids
LINKS:	KEGG::ec3.5.1.14, CBS::slr1653, CBS::sll0100
NOTES:	ChEBI link is missing

The fact that most fields in entity and rule definitions are tightly coupled with information from linked ontologies is the reason why we have started with describing annotation attributes. In the first place, one of these attributes is ENTITY NAME, which is taken from ontologies or follows the standard naming conventions. ENTITY ID of every entity is fixed by the consortium. KEGG ID, ChEBI ID or internal ID is used if no reasonable ID is available. IDs of rules are internal and assigned automatically.

Example 3.2 *Complete information given for an atomic entity.*

ENTITY ID:	HCO3
STATES:	{-, +}
LOCATIONS:	cyt, liq
COMPOSITION:	
ENTITY NAME:	hydrogencarbonate
CLASSIFICATION:	small molecule
DESCRIPTION:	Plays major role in carbon concentrating mechanism (CCM).
LINKS:	CHEBI::17544
ORGANISM:	Synechococcus elongatus PCC 7942

An entity in our interpretation is a bounded space (a so-called *compartment*) or a structural part of a specific organism. BCS covers a hierarchy of entities ranging from small molecules (*atomic entities (agents)*) through composite structures (*structure entities (agents)*) to large complex molecules (*complex entities (agents)*). Our goal is to make BCS as simple as possible. In existing ontologies, entities residing in several different states (oxidised, reduced, etc.) are usually treated as separate entities, thus causing the total number of entities to be enormous. To reduce this complexity, the concept of STATES is defined in BCS. All states are enclosed in curly brackets and they are comma-separated. The relationship entity–state is of the form parent–child. All information about an entity is inherited by its states unless it is specified explicitly for a particular state. The ID of an entity and its state in curly brackets form together a unique *entity identifier*. If no state is specified, the default value is the ‘neutral’ (ground) state.

BCS extends the traditional concept of compartmentalisation with a hierarchy at the level of entities. A particular entity can reside in several different compartments as specified in the LOCATIONS field. Additionally, the CLASSIFICATION field specifies the type of an entity in a sense of functional or structural characterisation.

An entity can be a part of a structurally more complex entity. We consider two kinds of composite entities: structure and complex entities. Structure entity represents partially specified composite species (we employ the partial composition operator '|', e.g., $ps2(chl|yz|oec)$), a photosystem complex partially specified with prosthetic groups of interest $ps2(chl|yz|oec)$). Complex entity represents fully specified composite species (we employ the full composition operator '.', e.g., a homodimer $KaiC.KaiC$). The composition of a composite entity is given in the field COMPOSITION. We employ a so-called *localisation operator* '::' to express the fact that an entity plays a role of a location for the structurally simpler entity (e.g., chlorophyl *chl* located in a photosystem $ps2$ is written $chl :: ps2$). In Example 3.3 there is a protein KaiC specified as a partial composition of two amino acids of interest – serine (S) and threonin (T). In such a configuration, serine-phosphorylated state of KaiC can be written as $S\{p\} :: KaiC$.

Example 3.3 *Complete information given for a structure entity.*

ENTITY ID:	KaiC
STATES:	
LOCATIONS:	cyt
COMPOSITION:	S | T
ENTITY NAME:	circadian clock protein kinase KaiC
CLASSIFICATION:	enzyme
DESCRIPTION:	Monomer component representing a core component of the circadian clock system.
LINKS:	uniprot::Q79PF4, cyanobase::Synpcc7942_1216
ORGANISM:	Synechococcus elongatus PCC 7942

Rules are specified by rule equations enriched with additional annotation information. When defining a rule equation, identifiers of substrates and products are used to make the notation of rules compact. Every entity appearing in a RULE EQUATION has to be followed by the localisation operator associating it with a particular compartment. This is important especially for rules that act on both sides of a membrane. That way, a rule is always precisely localised in/inbetween compartments. A natural stoichiometric coefficient can be placed before any entity in a rule equation. Irreversible and reversible rules are distinguished by the operators '⇒', '⇔'. The '+' symbol is used as a separator between individual substrates and individual products. A rule can also have an assigned classification. Rule classification assigns a list of higher level biophysical processes in which the rule is involved.

Example 3.4 *Complete information for a rule.*

RULE ID:	NADPH oxid.
RULE EQUATION:	$NADPH :: cyt + 5h\{\mathbf{+}\} :: cyt + pq :: cym \Rightarrow$ $\Rightarrow NADP\{\mathbf{+}\} :: cyt + 4h\{\mathbf{+}\} :: pps + pqh2 :: cym$
MODIFIER:	NDH1
RULE NAME:	plastoquinone reduction in the cytoplasmic membrane
CLASSIFICATION:	reduction, oxidation
DESCRIPTION:	Oxidation of NADPH and reduction of plastoquinone in the cytoplasmic membrane.

In some cases, emphasis on a detailed description leads to very complex BCS models. Abstraction of some processes is therefore needed to keep BCS models as simple as possible. To this end, rules expressing enzymatic reactions are considered in a simplified form. In fact, there should be at least two different rules describing an enzymatic reaction (one for a substrate binding and another for a catalytic step). Instead, since an enzyme is not affected during the reaction, it is affiliated to the rule as a MODIFIER. However, it is difficult to define precise meaning of a modifier in this case. We rather treat the modifier field informally as an entity *which has to be present* for the rule to be enabled. The exact reaction mechanism of an enzyme is not always clear and therefore it is abstracted out (see Example 3.4).

Example 3.5 *A rule employing structure entity state change.*

RULE ID:	FGFR2 phosph.
RULE EQUATION:	$Thr\{u\} :: FGF :: FGFR2 :: cyt \Leftrightarrow Thr\{p\} :: FGF :: FGFR2 :: cyt$
MODIFIER:	NDH1
RULE NAME:	FRRG2 threonine residue (de)phosphorylation
CLASSIFICATION:	phosphorylation, dephosphorylation
DESCRIPTION:	FGF enzyme is phosphorylated on threonine residue in FGFR2 complex.

Higher abstraction comes into account when several electrons play ‘musical chairs’ inside protein complexes. The issue is that parts of processing protein complex can have different unstable states during a short period of time. When one tries to define all rules among these proteins, combinatorial explosion of the number of states of the complex arises. Not all of these combinations are biologically correct. Even when excluding biologically inadmissible cases, the number of states is still enormous. For the purpose of BCS, we introduce a solution inspired by the enzymatic rule mentioned above. We treat a protein complex as a structure entity on which structurally simpler entities change its state (not necessarily proteins) and we abstract from background processes. We can see a particular rule as a change of a state of a structure entity (see Example 3.5).

4 Formal Definition of Biochemical Space

At the general level, BCS is a complex annotation format for description of the reaction network including textual annotation and links to existing annotation bases. The rigorous (rule-based) core of the language is made by declaration of chemical entities and reaction rules. The annotation part has been described in [10]. Here we define the formal core of BCS and associate it with a formal semantics by means of translating BCS rules into $kappa_s$.

Model in BCS is defined in similar way as a $kappa_s$ model. First, we define syntax of expressions describing *agents* formally in BCS. Next, the notion of *agent signature* is defined that allows to specify restrictions on the general expressions. Finally, agents are used as elementary constructs in definition of BCS *rules*.

4.1 BCS Agents

Let $\mathcal{N}_a$, $\mathcal{N}_T$, $\mathcal{N}_x$, $\mathcal{N}_c$, $\mathcal{N}_s$ be mutually exclusive finite sets of atomic, structure, complex, compartment, and state names, respectively.

Agents are defined hierarchically starting from *atomic agents* that are of two kinds: *class atoms* representing (abstract) class agents and *object atoms* representing (concrete) object agents. Class atomic agents allow us to represent compactly objects that can reside in several selected (or even all possible) states whereas object atomic agents represent concrete objects specified with the particular state. Every atomic agent must be accompanied with a physical compartment within which it is considered.

Atomic agent expressions have the following syntax:

atomic agent	$\mathsf{a} ::= \mathsf{a}_{\Box} \mid \mathsf{a}_{\diamond}$	state signature	$\delta ::= \delta,\ s \mid s$
class atom	$\mathsf{a}_{\Box} ::= \alpha^{\delta} :: \mathsf{c}$	state	$s ::= n \in \mathcal{N}_s$
object atom	$\mathsf{a}_{\diamond} ::= \alpha\{s\} :: \mathsf{c}$	compartment	$\mathsf{c} ::= n \in \mathcal{N}_c$
atom name	$\alpha ::= n \in \mathcal{N}_a$		

From now on, we restrict ourselves to atomic agents where the state signature can be treated as a set (a state cannot occur more than once in a state signature). This restriction is motivated by the aim to keep the language as simple as possible. Treating the state signatures as multisets would lead to confusions and is actually not needed to clearly represent biological objects.

Definition 4.1 Let a, a' be atomic agents. We define the *structural equivalence of atomic agents* by claiming $\mathsf{a} \equiv \mathsf{a}'$ whenever a, a' are (i) two identical object atoms or (ii) two identical class atoms that differ only in the order of states in the state signature.

Notation 4.2

- *We denote $s \in \delta$ the fact that s is included in the state signature δ.*
- *For better readability of class atomic agents, we enclose non-trivial state signatures into curly brackets. I.e., we write $\alpha^{\{\delta\}}$ instead of α^{δ} whenever δ contains more than one state.*

Since our notion of atomic agents considers concrete objects as well as general classes of objects, we need to formally relate a class with concrete objects that instantiate it. To this end, we define compatibility relation $\lhd$ that is stronger than structural equivalence.

Definition 4.3 *Let a, a' be atomic agents. We say a is* compatible with a'*, written*

$\mathsf{a} \lhd \mathsf{a}'$, *iff* $\mathsf{a} \equiv \mathsf{a}'$ *or iff there exist* $\alpha, \alpha', s, \delta, \mathsf{c}$ *such that* $\mathsf{a} = \alpha\{s\} :: \mathsf{c}$, $\mathsf{a}' = \alpha'^{\delta} :: \mathsf{c}$, $s \in \delta$, *and* $\alpha = \alpha'$.

An example of a class and object atomic agents is given in Table 1. In particular, the class atom $S^{\{u,p\}}$ represents a serine amino acid that can be considered in two different states. An object atom $S\{u\}$ represents the unphosphorylated form of serine.

Next we proceed with defining *structure agents*. A structure agent represents a biochemical object that is composed from several known atomic agents provided that we know that such a composition is abstract and not necessarily complete. To incorporate such an abstraction of biological structures into our language, a structure agent is defined to be labelled with a unique name and it is constructed only from atomic agents considered in the same physical compartment.

The key construct of a structure agent is *partial composition* defined as a list of atomic agents which are considered to be relevant parts of the structure agent. We allow this list to be empty, in that case the meaning is a biological structure for which we do not know its composition.

A typical example of a structure agent is a protein where the atomic agents are individual amino acids that are of interest in the particular setting. In Table 1 there is an example of a cyanobacteria clock protein $KaiC$ specified with an interest put to the serine amino acid (here denoted by the class atomic agent S).

$$\begin{array}{ll} \text{structure agent} & \mathtt{T} ::= \tau(\gamma_p) :: \mathsf{c} \\ \text{structure name} & \tau ::= n \in \mathcal{N}_T \\ \text{partial composition} & \gamma_p ::= \emptyset \mid \mathsf{a} \shortmid \gamma_p \end{array}$$

We restrict the language to structure agents where the partial composition does not contain replicated agents (stoichiometry is not considered at this level). More precisely, in every partial composition there is always at most one occurrence of an atomic agent with a name $n \in \mathcal{N}_a$. The main motivation for such a simplification is again the practical purpose of our language. The concept of partial composition is primarily considered as a rigorous identification of relevant parts of the non-trivial biochemical entity (most typically a protein). These parts are possibly subject to state changes.

Note that a compartment of a structure agent is uniquely given by the compartment specified in its parts. We restrict ourselves to structure agents where all atomic agents in the partial composition have the same compartment. Assuming this restriction, we can shorten the notation by omitting compartments in the atomic agents of a partial composition.

Notation 4.4

- *We denote* $\tau(...\shortmid\mathsf{a}\shortmid...) :: \mathsf{c}$ *a structure agent of the name* τ *such that an atomic agent* a *makes its part.*
- *We denote* $\mathsf{a} \in \gamma_p$ *the fact that* γ_p *includes the atomic agent* a.
- *The agent of the form* $\tau(\emptyset) :: \mathsf{c}$ *is usually written as* $\tau :: \mathsf{c}$.
- *A structure agent* $\tau(\gamma_p) :: \mathsf{c}$ *is usually written* $\tau(\alpha_1\shortmid\alpha_2\shortmid...\shortmid\alpha_n) :: \mathsf{c}$ *where* $\alpha_1, ..., \alpha_n$

are names of all agents in γ_p such that $\gamma_p = \mathtt{a}_1|\ldots|\mathtt{a}_n$ where each agent $\mathtt{a}_i$ is either of the form $\mathtt{a}_i = \alpha_i^{\delta_i} :: \mathtt{c}$ or $\mathtt{a}_i = \alpha_i\{s_i\} :: \mathtt{c}$ for some δ_i a state signature, s_i a state, and $\mathtt{c}$ a compartment shared among all agents in γ_p.

Definition 4.5 *Let $\mathtt{T}, \mathtt{T}'$ be structure agents. We define the* structural equivalence of structure agents *by claiming $\mathtt{T} \equiv \mathtt{T}'$ iff there exist $\tau, \tau', \gamma_p, \gamma_p', \mathtt{c}$ such that $\mathtt{T} = \tau(\gamma_p) :: \mathtt{c}, \mathtt{T}' = \tau'(\gamma_p') :: \mathtt{c}$, $\tau = \tau'$ and γ_p, γ_p' are equal or differ only in the order of its components (the operator '$|$' is considered associative and commutative).*

As a representative of a class of structurally equivalent structure agents we consider the agent $\tau(\gamma_p)$ where the agents in γ_p are lexicographically ordered by names. Since atomic agents cannot be repeated in a structure agent, such an order is total.

Definition 4.6 *Let $\mathtt{T}, \mathtt{T}'$ be structure agents. We say* $\mathtt{T}$ is compatible with $\mathtt{T}'$, *written $\mathtt{T} \lhd \mathtt{T}'$, iff either $\mathtt{T} \equiv \mathtt{T}'$ or for each atomic agent $\mathtt{a}$ such that $\mathtt{T} = \tau(\ldots|\mathtt{a}|\ldots) :: \mathtt{c}$ there exists an atomic agent $\mathtt{a}'$ such that $\mathtt{T}' = \tau'(\ldots|\mathtt{a}'|\ldots) :: \mathtt{c}$, $\tau = \tau'$ and $\mathtt{a} \lhd \mathtt{a}'$.*

In the following we define the last step in the hierarchy of agents. In particular, we define *complex agents.* A complex agent represents a non-trivial composite biochemical object that is (inductively) constructed from already known biological objects. In common rule-based languages this is typically defined by introducing some kind of bonds between individual biochemical objects. In BCS we abstract from detailed specification of bonds and we rather assume a complex as a coexistence of certain objects in a particular group. Such a group can be optionally referred to by a unique name. A complex agent is constructed from structure agents where all are required to reside in the same compartment $\mathtt{c}$.

A complex agent is given either directly as an expression inductively built by applying coexistence operator '.' to structure agents or indirectly as a name referring to a separate set of definitions of complex agents (incorporated in the notion of agent signature). We use that approach because we do not want to overcomplicate complex agent expressions.

The key element of a complex agent is *full composition* describing inductively constructed coexistence expressions from existing agents. We restrict ourselves to full compositions where all agents reside in the same compartment.

complex agent	$\mathtt{X} ::= \gamma_f :: \mathtt{c} \mid n \in N_x :: \mathtt{c}$
full composition	$\gamma_f ::= \mathtt{T}.\mathtt{T} \mid \mathtt{T}.\gamma_f$

In contrast to partial compositions, we allow replication at the level of full compositions (an agent of a certain name can appear more than once in a full composition). Moreover, names of complex agents are not associated with particular full compositions at the level of agent expressions. This is done at the level of agent signatures (see Section 4.2).

Note that in similar way as in the case of structure agents, we restrict the formalism to complex agents where the compartment is the same for all agents inside the respective full composition.

Notation 4.7

$\mathtt{a}_\square$ *atomic class agent*	$S^{\{u,p\}} :: cyt$ Serine (S) in two possible states phosphorylated (p) and unphosphorylated (u) existing in compartment cytosol (cyt).
$\mathtt{a}_\diamond$ *atomic object agent*	$S\{u\} :: cyt$ Serine (S) in state unphosphorylated (u) existing in compartment cytosol (cyt).
$\mathtt{T}$ *structure agent*	$KaiC(S^{\{u,p\}}) :: cyt$ Protein KaiC containing Serine (S) in its partial composition γ_p. It is possible to obtain two different derivations $KaiC(S\{u\}) :: cyt$ and $KaiC(S\{p\}) :: cyt$.
$\mathtt{X}$ *complex agent*	$KaiC(S^{\{u,p\}}).KaiC(S^{\{u,p\}}).KaiC(S^{\{u,p\}}).KaiC(S^{\{u,p\}}).KaiC(S^{\{u,p\}}).KaiC(S^{\{u,p\}}) :: cyt$ Complex of six KaiC structure agents (order does no matter).

Table 1
Examples of different forms of an agent.

- *Let* $\mathtt{X} = \gamma_f :: \mathtt{c}$ *for some full composition* γ_f*. We denote* $\mathtt{T} \in \mathtt{X}$ *the fact that* $\mathtt{T}$ *is a structure agent included in* γ_f*. Moreover, we denote* $\#\mathtt{T}[\mathtt{X}]$ *the number of occurrences of* $\mathtt{T}$ *in* γ_f*.*
- *For a complex agent* $\mathtt{X} = \gamma_f :: \mathtt{c}$ *where each item* $x \in \mathtt{X}$ *is an agent assigned to a compartment* $\mathtt{c}$*, we can use simplified notation that omits the compartment suffix '*$:: \mathtt{c}$*' in individual agents of* γ_f*.*

Next we define structural equivalence of complex agents. We employ set-based approach to aggregate complex agents into equivalence classes. In particular, at that level we achieve commutativity and associativity of the operator '.'.

Definition 4.8 *Let* $\mathtt{X}, \mathtt{X}'$ *be complex agents. We define* structural equivalence *of* complex agents *by claiming* $\mathtt{X} \equiv \mathtt{X}'$ *iff either of the following conditions holds:*

(i) *There exist a compartment* $\mathtt{c}$ *and* $n, n' \in \mathcal{N}_x$ *such that* $\mathtt{X} = n :: \mathtt{c}, \mathtt{X}' = n' :: \mathtt{c}$ *and* $n = n'$*.*

(ii) *If both* $\mathtt{X}, \mathtt{X}'$ *are specified as full compositions then the following two conditions must be satisfied:*
- *for each* $\mathtt{T} \in \mathtt{X}$ *there exists* $\mathtt{T}' \in \mathtt{X}'$ *such that* $\mathtt{T} \equiv \mathtt{T}'$ *and* $\#\mathtt{T}[\mathtt{X}] = \#\mathtt{T}'[\mathtt{X}']$*,*
- *for each* $\mathtt{T}' \in \mathtt{X}'$ *there exists* $\mathtt{T} \in \mathtt{X}$ *such that* $\mathtt{T}' \equiv \mathtt{T}$ *and* $\#\mathtt{T}'[\mathtt{X}'] = \#\mathtt{T}[\mathtt{X}]$*.*

An example of a complex agent is given in Table 1 where the given complex agent expression represents a large set of hexamers composed from $KaiC$ molecules each considered in arbitrary state.

Remark 4.9 From now on, we always consider a *lexicographically ordered agent* as a *representative of a class of structurally equivalent agents*. Since agents are defined hierarchically, lexicographical order is applied recursively to all nested agents. This allows us to always have a clearly defined unique representative.

4.2 BCS Agent Signatures

The language of agents defined in the previous section gives us a formal way how to encode biochemical objects at several levels of hierarchy and abstraction. The notion of structure agents allows to generate arbitrary partial compositions. Practically, we need to restrict the construction of composite biochemical objects by giving a set of constraints reflecting our understanding of biological objects and the desired level of abstraction. This can be achieved by assigning every structure agent name with a maximal partial composition that gives the restriction on structure agents that can be considered.

Similarly, the set of complex agents also needs to be restricted by specifying the catalogue of complex biochemical objects that can appear in the considered biochemical space. This can be achieved by assigning every complex agent name with a full composition that provides its definition. This allows us to name biological compounds, e.g., H_2O, and specify their clear definition under the coexistence abstraction, e.g., $H.H.O$.

Definition 4.10 *We say a pair* (Σ_τ, Σ_x) *is* agent signature *where* Σ_τ *and* Σ_x *are relations representing constraints on the construction of structure agents and complex agents, respectively, defined in the following way:*

- *Every* $\tau \in \mathcal{N}_T$ *is assigned some partial composition* γ_p, $(\tau, \gamma_p) \in \Sigma_\tau$.
- *Every* $n \in \mathcal{N}_x$ *is assigned some full composition* γ_f, $(n, \gamma_f) \in \Sigma_x$.

Definition 4.11 *We say a structure agent* $\mathtt{T} = \tau(\gamma_p)$ satisfies an agent signature (Σ_τ, Σ_x), *written* $\mathtt{T} \models (\Sigma_\tau, \Sigma_x)$, *iff for every atomic agent* $\mathtt{a} \in \gamma_p$ *there exists* $\mathtt{a}' \in \gamma_p'$ *such that* $\mathtt{a} \lhd \mathtt{a}'$ *and* $(\tau, \gamma_p') \in \Sigma_\tau$.

Remark 4.12 Note that for every $\tau \in \mathcal{N}_T$ the pair $(\tau, \gamma_p) \in \Sigma_\tau$ specifies the most general structure agent of the name τ with respect to the relation $\lhd$. The meaning of the agent $\tau(\emptyset) :: \mathtt{c}$ (simply written $\tau :: \mathtt{c}$) is a short form for the most general structure agent of the name τ specified in the signature. This is just the agent given by the partial composition γ_p. Any agent $\tau(\gamma_p')$ where γ_p' is constructed from γ_p by omitting some atomic agents makes a short form for $\tau(\gamma_p)$.

Note that with respect to Remark 4.12 we can write structure agents very compactly. E.g., assume $(\tau, \alpha_1\{s\}|\alpha_2^{\{s_1,s_2\}}) \in \Sigma_\tau$ then $\tau(\emptyset)$ is interpreted as $\tau(\alpha_1\{s\}|\alpha_2^{\{s_1,s_2\}})$, $\tau(\alpha_2\{s_1\})$ as $\tau(\alpha_1\{s\}|\alpha_2\{s_1\})$, etc.

Next we define expansion of named complex agent with respect to the given signature. In particular, we treat each pair $(n, \gamma_f) \in \Sigma_x$ as a specification of a full composition that is named n. Expansion then means replacing every complex agent name with the respective full composition. Finally, agents expanded with respect to a given signature are treated as agents satisfying the signature.

Definition 4.13 *Let* $(\Sigma_\tau,\ \Sigma_x)$ *be a signature. Every complex agent* $\mathtt{X} = n \in \mathcal{N}_x$ *is* expanded with respect to an agent signature $(\Sigma_\tau,\ \Sigma_x)$, *written* $\mathtt{X}[(\Sigma_\tau,\ \Sigma_x)]$ *and defined* $\mathtt{X}[(\Sigma_\tau,\ \Sigma_x)] = \gamma_f$ *where* $(n, \gamma_f) \in \Sigma_x$.

Definition 4.14 *We say a complex agent* $\mathtt{X} = \gamma_f$ satisfies an agent signature $(\Sigma_\tau,\ \Sigma_x)$, *written* $\mathtt{X} \models (\Sigma_\tau,\ \Sigma_x)$, *iff every structure agent* $\mathtt{T} \in \gamma_f$ *satisfies* $\mathtt{T} \models (\Sigma_\tau,\ \Sigma_x)$. *A complex agent* $\mathtt{X} = n \in \mathcal{N}_x$ satisfies a signature $(\Sigma_\tau,\ \Sigma_x)$ *iff* $\mathtt{X}[(\Sigma_\tau,\ \Sigma_x)] \models (\Sigma_\tau,\ \Sigma_x)$.

4.3 BCS Rules

At this point, we proceed to define the set of BCS rules. In contrast to $kappa_s$, a BCS rule has more complicated structure. This is due to the fact that BCS goes closer to traditional formalism of chemical reactions, in particular, BCS rules consider stoichiometry and compartmentalisation of reacting species. Moreover, to a certain extent we introduce variables in rule expressions allowing us to compact specification of repeating objects.

The list of rules $\mathtt{R}$ is defined by the following syntax:

rules	$\mathtt{R} ::= \emptyset \mid r, \mathtt{R}$
rule equation	$r ::= \Gamma \ \odot \ \Gamma$
direction	$\odot ::= \ \Rightarrow \ \mid \ \Leftrightarrow$
rule expression	$\Gamma ::= \emptyset \mid \varrho\ \epsilon :: \mathtt{c} \mid \varrho\ \epsilon :: \mathtt{c}\ +\ \Gamma$
stoichiometry	$\varrho ::= n \in \mathbb{N}^+$
rule expression item	$\epsilon ::= \epsilon_1 \mid \epsilon_2 \mid \epsilon_3$
basic rule agent	$\epsilon_1\ ::=\ \mathtt{a} \mid \mathtt{T} \mid \mathtt{X}$
shallow rule agent	$\epsilon_2\ ::=\ \mathtt{a} :: \mathtt{T} \mid \mathtt{T} :: \mathtt{X}$
deep rule agent	$\epsilon_3\ ::=\ \mathtt{a} :: \mathtt{T} :: \mathtt{X}$

We assume that a single rule cannot appear more than once in the list $\mathtt{R}$ (every rule must be unique). In relation to that, we can use the notation $r \in \mathtt{R}$ to refer to rules in $\mathtt{R}$. See Section 7 for examples of several rules.

Rule expressions allow more extensive syntax in terms of the *localisation* operator '::'. The localisation operator is intended for allowing an alternative way of expressing the hierarchically constructed agents. The main idea is to allow zooming into individual parts of a complex or a structure agent. E.g., for a structure agent $\tau(\alpha_1\{s\}|\alpha_2^{\{s,t\}}) :: \mathtt{c}$ residing in compartment $\mathtt{c}$ we can use the notation $\alpha_2\{t\} :: \tau(\alpha_1\{s\}|\alpha_2^{\{s,t\}}) :: \mathtt{c}$ to refer explicitly to a concretisation of its subagent α_2. This notation is fully equivalent with the original form $\tau(\alpha_1\{s\}|\alpha_2\{t\})$ and can be therefore considered as an alternative way to concretise a structure agent.

Similarly, the concept of localisation is applied also to complex agents. E.g., for a complex agent $A(\alpha_1\{s\}).B(\alpha_2^{\{s,t\}}) :: \mathtt{c}$ we can zoom to some of its components and express its concretisation such as $B(\alpha_2\{t\}) :: A(\alpha_1\{s\}).B(\ \alpha_2^{\{s,t\}}) :: \mathtt{c}$. In this case, the notation $B(\alpha_2\{t\}) :: A(\alpha_1\{s\}).B(\alpha_2^{\{s,t\}})$ is equivalent to the complex agent $A(\alpha_1\{s\}).B(\alpha_2\{t\})$.

In every rule subexpression $\varrho\ \epsilon :: \mathtt{c}$ the compartment $\mathtt{c}$ makes the scope for every agent appearing in ϵ. In particular, every agent inside ϵ is assumed to be assigned the compartment $\mathtt{c}$.

To simplify the resulting language to construct reasonable expressions only, we restrict ourselves to rules where the operator '::' respects constraints given in Definition 4.15.

Definition 4.15 Let ϵ be a rule expression item that appears in a rule $r \in \mathtt{R}$. The *rule expression* ϵ is *well-defined* iff the following constrains are satisfied:

(i) If $\mathtt{a} :: \tau(\gamma_p)$ is a subexpression of ϵ for some $\mathtt{a}, \tau, \gamma_p$ then there must exist $\mathtt{a}' \in \gamma_p$ such that $\mathtt{a} \lhd \mathtt{a}'$.

(ii) If $\mathtt{T} :: \mathtt{X}$ is a subexpression of ϵ for some $\mathtt{T}, \mathtt{X}$ then there must exist $\mathtt{T}' \in \mathtt{X}$ such that $\mathtt{T} \lhd \mathtt{T}'$.

Every rule agent in a shallow or deep form can be translated to an equivalent basic form. Formally, this is given in Lemma 4.16.

Lemma 4.16 (Rule Flattening) *Let (Σ_τ, Σ_x) be a signature and $\mathtt{R}$ a set of rules. Every rule $r \in \mathtt{R}$ that includes some rule agents in shallow or deep form can be reduced to a rule $r' \in \mathtt{R}$ where every rule agent is in basic form. For every rule agent ϵ in r, the reduction is done by replacing ϵ with ϵ' in the following way:*

(i) *If $\epsilon = \mathtt{a} :: \mathtt{T}$ where $\mathtt{T} = \tau(\gamma_p)$ for some τ, γ_p then there must exist $\mathtt{a}' \in \gamma_p$ such that $\mathtt{a} \lhd \mathtt{a}'$. Then we set $\epsilon' = \tau(\gamma_p')$ where γ_p' is constructed from γ_p by replacing $\mathtt{a}' \in \gamma_p$ with $\mathtt{a}$.*

(ii) *If $\epsilon = \mathtt{T} :: \mathtt{X}$ where $\mathtt{X} = \gamma_f$ then there must exist $\mathtt{T}' \in \gamma_f$ such that $\mathtt{T} \lhd \mathtt{T}'$. Then we set $\epsilon' = \gamma_f'$ where γ_f' is constructed from γ_f by replacing $\mathtt{T}' \in \gamma_f$ with $\mathtt{T}$.*

(iii) *If $\epsilon = \mathtt{a} :: \mathtt{T} :: \mathtt{X}$ then the steps (i,ii) above are applied successively.*

Definition 4.17 We say that a rule $r \in \mathtt{R}$ *satisfies agent signature* $(\Sigma_\tau,\ \Sigma_x)$, written $r \models (\Sigma_\tau,\ \Sigma_x)$, iff every structure or complex agent that appears as a rule agent in r satisfies agent signature $(\Sigma_\tau,\ \Sigma_x)$.

To increase succinctness, we extend the language with a variable $?\nu$. A variable can be assigned to any rule in place of an agent. Evaluation of a variable within a rule is realised for every occurrence of $?\nu$. For a given signature (Σ_τ, Σ_x) we assume that after evaluating the variable, every rule agent must satisfy the signature and is well-defined. Moreover, the scope of the compartment is always uniquely given in the rule expression. The domain of a variable is assumed to be considered as a set (values are not repeated). An example is given in Example 7.2. The extended syntax is the following:

extended rule equation	$r' ::= r \mid \Gamma \ \odot \ \Gamma \ ; \ var$
variable	$var ::= \emptyset \mid ?\nu = \{\phi\} \mid ?\nu_1 = \{\phi_1\} \mid ?\nu_2 = \{\phi_2\} \mid ?\nu_3 = \{\phi_3\}$
variable value	$\phi ::= \ \phi_1 \mid \phi_2 \mid \phi_3$
atomic variable value	$\phi_1 ::= \mathtt{a},\ \phi_1 \mid \mathtt{a}$
structure variable value	$\phi_2 ::= \mathtt{T},\ \phi_2 \mid \mathtt{T}$
complex variable value	$\phi_3 ::= \mathtt{X},\ \phi_3 \mid \mathtt{X}$
extended basic rule agent	$\epsilon_1' ::= \epsilon_1 \mid ?\nu$
extended shallow rule agent	$\epsilon_2' ::= \epsilon_2 \mid ?\nu_1 :: \mathtt{T} \mid \mathtt{a} :: ?\nu_2 \mid ?\nu_2 :: \mathtt{X} \mid \mathtt{T} :: ?\nu_3$
extended deep rule agent	$\epsilon_3' ::= \epsilon_3 \mid ?\nu_1 :: \mathtt{T} :: \mathtt{X} \mid \mathtt{a} :: ?\nu_2 :: \mathtt{X} \mid \mathtt{a} :: \mathtt{T} :: ?\nu_3$

Finally, we define the notion of a BCS model that is given by a signature and a set of rules.

Definition 4.18 A *BCS model* $\mathtt{M}$ is a tuple $((\Sigma_\tau, \Sigma_x), \mathtt{R})$ such that every $r \in \mathtt{R}$ it holds that $r \models (\Sigma_\tau,\ \Sigma_x)$.

5 Translation to *kappa*$_s$

To define semantics for BCS language, we give an algorithm that translates a given BCS model $\mathtt{M}$ to a *kappa*$_s$ model $\mathcal{M}$. We assume the model $\mathtt{M}$ is normalised using the following procedures:

(i) all rules are flattened by employing Lemma 4.16,

(ii) every bidirectional rule is replaced by the two respective unidirectional rules,

(iii) every rule with variables is replaced by the set of rules generated by expanding all acceptable values for every variable.

Algorithm 1. Transform a BCS model $\mathtt{M}$ to a *kappa*$_s$ model $\mathcal{M}$.

```
function toKappa( $\mathtt{M} = ((\Sigma_\tau, \Sigma_x), \mathtt{R})$ )
    $\Sigma_\kappa,\ I,\ \mathcal{R},\ \mathtt{A} := \emptyset$        # global $kappa_s$ signature, rules and agent names
    for all $r \in \mathtt{R}$ do
        for all $\Gamma \in \{\Gamma_l, \Gamma_r\}$ such that $r = \Gamma_l \Rightarrow \Gamma_r$ do        # for both rule sides
            $\mathtt{E} := \emptyset$
            for all $\varrho\ \epsilon :: \mathtt{c} \in \Gamma$ do        # repeat $\varrho$-times
                if $\epsilon$ has the form $\mathtt{a}\{s\}$ then
                    $\mathtt{E} \leftarrow$ translateAtom($\epsilon :: \mathtt{c}$)
                if $\epsilon$ has the form $\tau(\gamma_p)$ then
                    $\mathtt{E} \leftarrow$ translateStructure($\epsilon :: \mathtt{c}$)
                if $\epsilon$ has the form $\mathtt{X}$ then
                    $\mathtt{E} :=$ translateComplex($\epsilon :: \mathtt{c}$, $\mathtt{E}$)
        construct a $kappa_s$ rule $r_\kappa$ from the two resulting sets $\mathtt{E}$ obtained for $\Gamma_l, \Gamma_r$
        $\mathcal{R} \leftarrow r_\kappa$        # extend $kappa_s$ rules
    return $\mathcal{M} = (\Sigma_\kappa,\ I,\ \mathcal{R})$
```

Algorithm 1 takes a BCS model $\mathtt{M}$ and returns a *kappa*$_s$ model $\mathcal{M}$. It uses three subroutines that modify respective types of BCS agents. Algorithm 2 translates an atomic agent directly by extending an agent name with a compartment name and adding a site $\mathtt{p}$. Algorithm 3 translates a structure agent where each atomic agent in its partial composition is encoded as a unique site. Finally, algorithm 4 translates a complex agent where each structure agent in the respective full composition is treated as a *kappa*$_s$ agent. Since BCS does not provide binding sites, we fix linear binding (see Section 6 for further discussion).

Algorithm 2. Transforms an atomic agent to a *kappa*$_s$ agent.

```
function translateAtom($\mathtt{a}\{s\} :: \mathtt{c}$)
    $\mathtt{A} \leftarrow \mathtt{a\_c}$        # extend $kappa_s$ agent names
    $\Sigma_\kappa \leftarrow (\mathtt{a\_c}, \{\mathtt{p}\})$        # introduce a new site name $\mathtt{p}$ into signature
    $\sigma(\mathtt{a\_c}) \leftarrow \mathtt{p}_s$        # add the site to the agent interface
    return $\mathtt{a\_c}(\sigma)$        # return $kappa_s$ agent
```

Algorithm 3. Function transforms structure agent to *kappa*$_s$ agent.

```
function translateStructure($\tau\{\gamma_p\} :: c$)
    $\mathtt{A} \leftarrow \tau\_\mathtt{c}$        # extend $kappa_s$ agent names
    for all $\mathtt{a}\{s\} :: \mathtt{c} \in \gamma_p$ do
        $\Sigma_\kappa \leftarrow (\tau\_\mathtt{c}, \{\mathtt{a}\})$        # introduce a new site name into the signature
        $\sigma(\tau\_\mathtt{c}) \leftarrow \mathtt{a}_s$        # add the site to the agent interface
    return $\tau\_\mathrm{c}(\sigma)$        # return $kappa_s$ agent
```

Algorithm 4. Transforms a complex agent to a *kappa*$_s$ agent.

```
function translateComplex($\mathtt{X} :: \mathtt{c}$, $\mathtt{E}$)
    $i := 0$
    for all $\mathtt{T} \in \mathtt{X}$ do
        agent $\mathtt{a} :=$ translateStructure($\mathtt{T}$)
        $\Sigma_\kappa \leftarrow (\mathtt{a}, \{l, r\})$        # add sites for binding
        if $i \neq 0$ then
            $I \leftarrow ((\mathtt{a}, l), \{i\})$
        $\sigma(\mathtt{a}) \leftarrow l^i$        # set bond with left binding partner
        $i := i + 1$
        if $i \neq \#\mathtt{T}[\mathtt{X}]$ then
            $I \leftarrow ((\mathtt{a}, r), \{i\})$
        $\sigma(\mathtt{a}) \leftarrow r^i$        # set bond with right binding partner
        $\mathtt{E} \leftarrow \mathtt{a}(\sigma)$        # extend expression $\mathtt{E}$
    return $\mathtt{E}$
```

6 Comparison of BCS and BNGL

It is necessary to note that BCS currently does not provide quantitative semantics. It just describes the system structure and the relationships between entities.

Another issue is if there is a rule containing a modifier, there are two options how to express it in BNGL. The first option is to add the modifier to both sides of the rule, the second is to include it (quantitatively) in the reaction rate function. BCS has to employ the first option. There is also an alternative solution in BCS – the field *MODIFIER* in the BCS rule annotation record.

BCS does not provide binding sites. This is caused by the fact it also does not provide specification of a bond. In BNGL, binding makes a binary operation between two components and the bond must be always specified by using operators '.' and '!' where each bond has a unique ID inside a complex. This detailed notion is not present in BCS since we want to abstract such details.

However, this kind of abstraction introduces the inability to distinguish two complexes composed from the same subparts (e.g. proteins). For example, consider a protein P with a single binding site. A complex formed from n proteins P can be created from *n-1* bonds (linear conformation) to $NPI(n) = \frac{n(n-1)}{2}$, which is the maximal number of possible interactions inside the complex (assuming only one bond between two proteins is possible). It follows that a complex C_{BCS} formed from n proteins P in BCS is the set of all possible structural conformations C_i of the complex in BNGL where all proteins are considered:

$$C_{BCS} = \bigcup_{i=0}^{m} C_i, \text{ where } m \text{ is the number of connected graphs on } n \text{ nodes.}$$

Stringency of a rule makes a relevant difference. Stringency stands for degree of universality or specificity of the rule, i.e. the width of the applicability. In both languages, this can be solved by context of the rule. However, it is not always suitable to list the whole context. An example can be phosphorylation in circadian clock (Example 7.2). It can occur on each KaiC protein which is included in a complex. For this purpose there is '*site*!+' notation in BNGL which requires the protein to be in a bound state. Since BCS does not provide binding sites, this cannot be used.

To this end, we employ the localisation operator '::' in rule agents. It allows to nest rule agents to strengthen the stringency. Moreover, we have introduced *variables* in BCS. A variable in a reactant is denoted $?\nu$ and can be specified as a set of atomic, structure or complex agents to which the rule can be applied.

The last fact that is worth noting is construction of complex structures. In BNGL, each complex is identified with an exact structural notation which does not allow hierarchical construction. BCS provides the notion of structure and complex agents, this allows to form a hierarchy of the agents. Additionally, when defining a rule with quantities of interacting entities, in BNGL it is necessary to enumerate all of them whereas in BCS the stoichiometry is allowed in standard way.

6.1 BCS and BNGL translation

It is possible to translate from BCS to BNGL. This can be achieved by the application of finite set of transformation steps. The procedure is analogous to translation to $kappa_s$ (Section 5).

Translation BNGL to BCS is also possible, but the bond information is discarded in the process. In particular, all binding operations have to be removed. The only problem is the '!+' notation in BNGL which requires a bond for a entity. This kind of bond has a high level of abstraction. For this reason we cannot translate such a rule. However, every rule in BNGL with '!+' can be expanded to finite number of rules where this operator is omitted. Instead of an unknown bond, there are enumerated rules each accompanied with a known binding partner. In that case, the variable $?\nu$ is added to the BCS rule containing all the enumerated binding partners.

7 Case Study

BCS makes a part of CMP and is implemented at e-cyanobacterium.org and currently covers several functional modules of cyanobacteria. To support translation between BCS and BNGL, we have implemented a set of scripts[1] allowing to translate a BCS model to BNGL and vice versa.

7.1 Metabolism

Metabolism forms the backbone of cyanobacteria cellular processes and in BCS covers the largest part of cyanobacteria network. We distinguish two groups of entities in metabolism – enzymes and metabolites. Enzymes drive metabolic reactions and therefore are assigned to rules as modifiers. On the other hand, metabolites are small molecules playing a role of substrates or products of metabolic rules with no enzymatic function. Both groups are involved in rules which occur mostly in the cell cytoplasm, therefore the majority of their entities uses cytoplasm as a compartment.

Example 7.1 *A rule from metabolism of cyanobacteria. It is visualised in Figure 2 in the upper left part.*

RULE ID:	(S)-malate:NAD{**+**} oxidoreductase
RULE EQUATION:	$malate :: cyt + NAD\{\mathbf{+}\} :: cyt \Leftrightarrow oxaloacetate :: cyt +$ $+ NADH :: cyt + H\{\mathbf{+}\} :: cyt$
MODIFIER:	
RULE NAME:	malate oxidation
CLASSIFICATION:	oxidation, reduction
DESCRIPTION:	Process is involved in citric acid cycle. Malate is oxidised to oxaloacetate producing NADH from NAD{**+**}.

In metabolism, there are approximately 770 rules. Despite the fact that there are plenty of molecules, the rules are very specific. In our proposed rule-based language

[1] http://www.e-cyanobacterium.org/downloads/

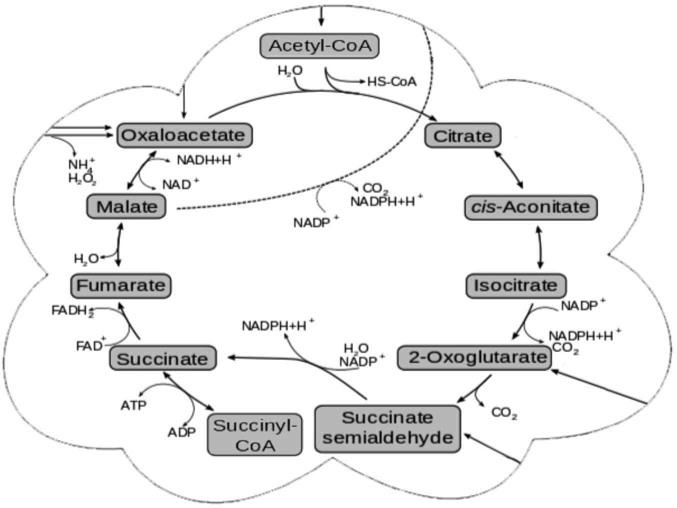

Fig. 2. Part of the reaction scheme of metabolism in cyanobacteria [12].

it means the mapping of reactions to rules is almost one-to-one (reaction-like rules). The stringency of rules is high which is what allows them to be applied only to a narrow group of molecules. It causes that compaction of metabolism in rules brings almost no benefits.

7.2 Circadian clock

Circadian clock is one of the most complex processes in cyanobacteria BCS. Its core is formed by three proteins $KaiA$, $KaiB$ and $KaiC$. Moreover, $KaiC$ contains two phosphorylation sites serine $S431$ and threonine $T432$. These sites can be phosphorylated independently, but only if $KaiC$ is in a complex. All these proteins can interact with each other in predetermined ways and form specific complexes. All processes inside the cell are then controlled by periodical formation/dissociation and (de)phosphorylation of these complexes.

Example 7.2 *Serine (de)phosphorylation on KaiC protein. In Figure 3 it is (also with threonine phosphorylation) responsible for all short cycles.*

RULE ID:	serine (de)phosph.
RULE EQUATION:	$S\{u\} :: KaiC :: ?X :: cyt \Leftrightarrow S\{p\} :: KaiC :: ?X :: cyt$; $?X = \{KaiC6, KaiA2C6, KaiB6C6, KaiA4C6, KaiA6B6C6\}$
MODIFIER:	
RULE NAME:	Serine phosphorylation and dephosphorylation
CLASSIFICATION:	phosphorylation, dephosphorylation
DESCRIPTION:	KaiC molecule is phosphorylated/dephosphorylated on serine amino acid. This process can appear whenever KaiC is in one of the complexes enumerated in variable X.

Owing to the fact the proteins can form homohexamers or smaller complexes, and each of these complexes can interact with others, it causes combinatorial ex-

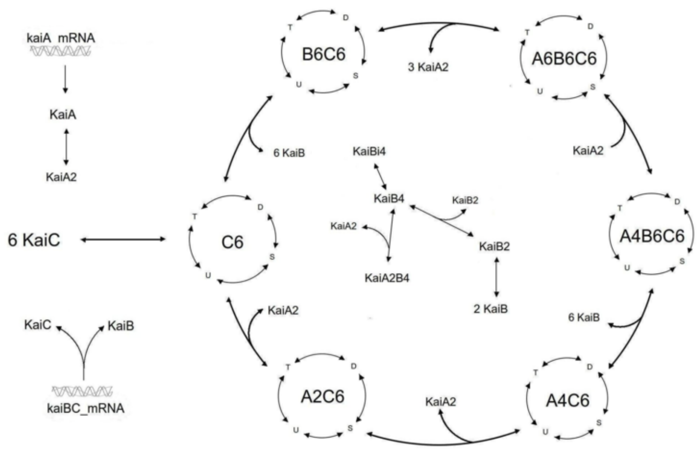

Fig. 3. Circadian clock cycle constructed by 17 BCS rules including complex formation, translation and phosphorylation.

plosion. Together there is possible formation of six different complexes containing $KaiC$: $KaiC6$, $KaiB6C6$, $KaiA2C6$, $KaiA4C6$, $KaiA4B6C6$ and $KaiA6B6C6$. Each protein $KaiC$ can occur in four different states because of the two phosphorylation sites. Considering all six complexes and also other rules in circadian clock, we obtain combinatorial explosion of different species in the system. To achieve representation of the whole system it is inefficient to enumerate each single conformation. To this end, we employ the capability of BCS rules.

Example 7.3 *Formation of KaiB6C6 complex is important for circadian clock. It can be seen in the upper left part of Figure 3, where it forms the bigger cycle (with all other complex formation rules).*

RULE ID:	KaiB6C6 form./diss.
RULE EQUATION:	$6\ KaiB :: cyt + KaiC6 :: cyt \Leftrightarrow KaiB6C6 :: cyt$
MODIFIER:	
RULE NAME:	KaiB6C6 complex formation and dissociation
CLASSIFICATION:	complex formation, dissociation
DESCRIPTION:	Formation of complex from six KaiB molecules and KaiC hexamer and its dissociation. KaiC6 represents specification of complex composed from six KaiC proteins, KaiB6C6 complex of six KaiC and six KaiB respectively.
LINKS:	doi::10.1093/emboj/18.5.1137, doi::10.1016/j.febslet.2009.11.021

In BCS we have achieved complete, human readable representation of circadian clock using only 17 rules (examples are rules in Example 7.2 and Example 7.3). Regarding the defined agents, it gives us over 500 different distinguishable entities, while in BNGL similar number of rules describing the same system gives us almost 25000 entities.

7.3 Photosynthesis

Photosynthesis represents part of BCS of cyanobacteria. The process occurs in a specific folds of the cell membrane called thylakoid membrane. Photosynthesis serves as the source of energy taken from light and transferred into production of ATP and NADPH molecules with oxygen resulting as a by-product.

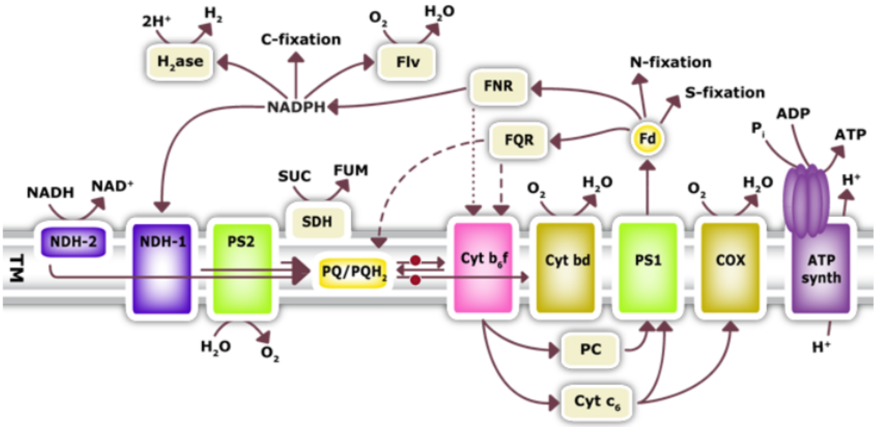

Fig. 4. Reaction scheme of photosynthesis in cyanobacteria. The lumen processes are displayed under thylakoid membrane while stroma processes are above.

Example 7.4 *A rule from photosynthesis. Oxidation reaction on PSII.*

RULE ID:	PSII oxidation
RULE EQUATION:	$ps2(oec\{3+\} \mid yz\{+\}) :: tlm \Leftrightarrow ps2(oec\{4+\} \mid yz\{n\}) :: tlm$
MODIFIER:	
RULE NAME:	oxidation from S3 to S4 of oxygen evolving complex
CLASSIFICATION:	oxidation
DESCRIPTION:	Oxidation occurring on photosystem II. Electron is transferred from oxygen evolving complex *oec* to active tyrosine *yz*.

Entities of photosynthesis BCS are represented by several complex proteins (enzymes) residing on the thylakoid membrane (tlm) in the cell. Since the thylakoid membrane encloses the inner-membrane space called lumen (lum) where H_2O molecules are processed, there are basically three locations defined for this set of entities. Rules occurring in the lumen, cytosol and in-between the thylakoid membrane and these locations have classical form. However, electron transfer reactions occurring in the structure of complex processes lead to combinatorial explosion of all possible conformations.

Photosynthesis is constructed from approximately 30 agent definitions which are interacting in over 60 rules. From the rule-based point of view, this representation is somewhere between circadian clock (Section 7.2) and metabolism (Section 7.1). It means the number of generated distinguishable entities arises compared to defined agents, but not as dramatically as in circadian clock. However, photosynthesis is a good example of rule-based process.

8 Conclusions

We have lifted the annotation format BCS to a formal language compatible with well-established rule-based languages. We have given an automated support for translating between BCS and BNGL. Currently, BCS is used on the portal e-cyanobacterium.org for description of cyanobacteria processes. In case study section we have shown the language is suitable for rule-based systems as well as reaction-based systems. For future work we plan to define an operational semantics directly without an intermediate format. This will enable implicit description of the model states space and allow to gain from the compact representation and take the advantages of on-the-fly model checking.

References

[1] Brim, L., J. Nižnan and D. Šafránek, *Compact representation of photosynthesis dynamics by rule-based models*, Electronic Notes in Theoretical Computer Science **316** (2015), pp. 17 – 27, 5th International Workshop on Static Analysis and Systems Biology (SASB 2014).

[2] 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.

[3] Danos, V., J. Feret, W. Fontana and J. Krivine, *Abstract interpretation of cellular signalling networks*, in: *Verification, Model Checking, and Abstract Interpretation, 9th International Conference, VMCAI 2008, San Francisco, USA, January 7-9, 2008, Proceedings*, 2008, pp. 83–97.

[4] Danos, V. and C. Laneve, *Formal molecular biology*, Theor. Comput. Sci. **325** (2004), pp. 69–110.

[5] Dematté, L., C. Priami and A. Romanel, *The blenx language: A tutorial*, in: M. Bernardo, P. Degano and G. Zavattaro, editors, *Formal Methods for Computational Systems Biology*, Lecture Notes in Computer Science **5016**, Springer Berlin Heidelberg, 2008 pp. 313–365.

[6] Faeder, J., M. Blinov and W. Hlavacek, *Rule-Based Modeling of Biochemical Systems with BioNetGen*, in: I. V. Maly, editor, *Systems Biology*, Methods in Molecular Biology **500**, Humana Press, 2009 pp. 113–167.

[7] Hlavacek, W. S., J. R. Faeder, M. L. Blinov, R. G. Posner, M. Hucka and W. Fontana, *Rules for modeling signal-transduction systems*, Sci. STKE **2006** (2006), p. re6.

[8] Hucka, M. et al., *The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models*, Bioinformatics **19** (2003), pp. 524–531.

[9] Kanehisa, M. and S. Goto, *Kegg: Kyoto encyclopedia of genes and genomes*, Nucleic Acids Research **28** (2000), pp. 27–30.

[10] Klement, M., T. Děd, D. Šafránek, J. Červený, S. Müller and R. Steuer, *Biochemical space: A framework for systemic annotation of biological models*, in: *Proceedings of the 5th International Workshop on Interactions between Computer Science and Biology (CS2Bio14)* (2014), pp. 31–44.

[11] Klement, M., D. Šafránek, T. Děd, A. Pejznoch, L. Nedbal, R. Steuer, J. Červený and S. Müller, *A comprehensive web-based platform for domain-specific biological models*, in: *Proceedings of the fourth International Workshop on Interactions between Computer Science and Biology (CS2Bio'13)* (2013), pp. 61–67.

[12] Knoop, H., M. Gründel, Y. Zilliges, R. Lehmann, S. Hoffmann, W. Lockau and R. Steuer, *Flux balance analysis of cyanobacterial metabolism: The metabolic network of synechocystis sp. pcc 6803*, PLoS Comput Biol **9** (2013), p. e1003081.

[13] Le Novere, N. et al., *Minimum information requested in the annotation of biochemical models (MIRIAM)*, Nat Biotech **23** (2005), pp. 1509–1515.

[14] Nakao, M., S. Okamoto, M. Kohara, T. Fujishiro, T. Fujisawa, S. Sato, S. Tabata, T. Kaneko and Y. Nakamura, *Cyanobase: the cyanobacteria genome database update 2010*, Nucleic Acids Research **38** (2010), pp. D379–D381.

[15] Nedbal, L., J. Červený and H. Schmidt, *Scaling and integration of kinetic models of photosynthesis: Towards comprehensive e-photosynthesis*, in: *Photosynthesis in silico*, Advances in Photosynthesis and Respiration **29**, Springer, 2009 pp. 17–29.

[16] Pedersen, M. and G. Plotkin, *A language for biochemical systems: Design and formal specification*, in: C. Priami, R. Breitling, D. Gilbert, M. Heiner and A. Uhrmacher, editors, *Transactions on Computational Systems Biology XII*, Lecture Notes in Computer Science **5945**, Springer Berlin Heidelberg, 2010 pp. 77–145.

[17] Šafránek, D., J. Červený, M. Klement, J. Pospíšilová, L. Brim, D. Lazár and L. Nedbal, *E-photosynthesis: Web-based platform for modeling of complex photosynthetic processes*, Biosystems **103** (2011), pp. 115–124.

[18] Zhang, F. and M. Meier-Schellersheim, “SBML Level 3 Package Specification: Multistate, Multicomponent and Multicompartment Species Package for SBML Level 3 (Version 1, Release 0.4 – Draft),” SBML.org (2015).