A simple calculus for proteins and cells

Cosimo Laneve, Fabien Tarissan

▶ **To cite this version:**

Cosimo Laneve, Fabien Tarissan. A simple calculus for proteins and cells. Theoretical Computer Science, 2008, Membrane Computing and Biologically Inspired Process Calculi, 404 (1-2), pp.127-141. 10.1016/j.tcs.2008.04.011 . hal-01217855

HAL Id: hal-01217855
https://hal.science/hal-01217855v1

Submitted on 20 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A simple calculus for proteins and cells

Cosimo Laneve, Fabien Tarissan,

Dipartimento di Scienze dell'Informazione, Università di Bologna

In memory of Nadia Busi

Abstract

The use of process calculi to represent biological systems has led to the design of different formalisms such as brane calculi and κ-calculus. Both have proved to be useful to model different types of biological systems. As an attempt to unify the formalisms, we introduce the `bio`κ*-calculus* , a simple calculus for describing proteins and cells, in which bonds are represented by means of shared names and interactions are modelled at the domain level. In `bio`κ-calculus, protein-protein interactions have to be at most binary and cell interactions have to fit with sort constraints.

In this contribution we define the semantics of `bio`κ-calculus, analyse its properties, discuss the expressivity of the calculus by modelling two significant examples – a signalling pathway and a virus infection –, and study an implementation in Milner's π-calculus.

1 Introduction

One problem when dealing with molecular biology is to extract a functional meaning out of the mass of current knowledge. This problem has pleaded for the development of specific tools that describe biology in a faithful way. Among these tools, process calculi have been proved powerful enough to formalise some of crucial activities of biological systems, to render in a natural way the massive parallelism and concurrency of interactions, and to analyse the overall behaviour.

There are two features of process calculi that are particularly relevant to biological systems. The first one is about the syntax. In process calculi, the syntax of the terms determines their capacity to interact with the environment. This is in strong analogy with molecular biology, where the structure of an object affects its activities. For instance, the three-dimensional structure of a protein, as well as the fact that it is part of an amino-acid sequence, may enable

some reactions and disable other ones. The second feature is about the semantics. Process calculi have models that have been thoroughly studied and that support automatic verification tools. These models may turn out essential in order to formalise the behaviour of biological systems (with qualitative measures, such as rate of reactions) and the reachability of problematics molecular configurations.

Two different process calculi families have brought some interest and various results. A first family is based on Milner's π-calculus [1], following principles proposed in [2]. According to these principles, bonds between proteins are represented in π-calculus by shared channel names and protein interactions are modelled by process communications. The κ-calculus [3] belongs to this family and it has been showed convenient for representing mechanisms such as signalling pathways or regulatory networks. The second family of calculi relies on Mobile Ambient [4], following ideas proposed by Paun [5]. These calculi use actions and co-actions capabilities located on the surface of cell membranes for representing molecular transports and virus infections. The *brane calculi* [6] and *bio-ambients* [7] belong to this family. As these two families turn out to commit to different paradigms, it is compelling to develop a unique formalism able to handle the two types of systems. The challenge is to enucleate few basic realistic primitives that are expressive enough to cover mechanisms of κ-calculus and brane calculi. Such a formalism is the aim of this contribution.

The calculus presented in this contribution – the $\texttt{bio}\kappa$-calculus – retains denotations for proteins, cells, and solutions. Protein reactions are *complexations* and *decomplexations* of *two* proteins. These reactions follow the same pattern of those of κ-calculus. Actually, they are similar, but even simpler, to those of $\texttt{m}\kappa$-calculus [3], an intermediate formalism introduced to ease the translation of κ-calculus in π-calculus. Cells are compartments consisting of a *membrane* and a *cytoplasm* – a solution contained into the membrane. Cellular reactions permit to fuse two cells. Fusions open the cell to other cells. In particular, the cytoplasms, whose interactions with the external solution are mediated by the membrane, may directly interact with another solution – the cytoplasm of the fusing cell – after the fusion. Complexations and decomplexations of a reactant that belongs to a cell membrane may change the capability of the membrane, thus preparing the cell to future fusions, endo- or exocytosis. The formal rendering of these phenomena is done in $\texttt{bio}\kappa$-calculus by admitting protein reactions with side-effects on membrane types. Other relevant cellular reactions are considered in $\texttt{bio}\kappa$-calculus, such as *translocations*, which transport proteic material inside the cells, and *phagocytosis*, which allows a cell, such as a virus, to enter other cells. Phagocytosis is particularly difficult to model because the entering cell is enclosed into a membrane that is pull out the host cell membrane. This means that the membrane of the host cell must have enough material for the new one. This mechanism is modelled by admitting matches of patterns of proteins.

The formal models of bioκ-calculus are labelled transition systems where labels carry information about the reactants and the biological rule that is being used. It is folklore in process calculi that such systems are too descriptive (too *intentional objects*) because they separate solutions that no experiment may distinguish. A number of equivalences over transition systems have been proposed in the literature. In this paper we consider *weak bisimulation* [8] that equates two systems if they simulate each other. We demonstrate that weak bisimulation is a congruence in bioκ-calculus: two weakly bisimilar biological systems behave in the same way when put in *every* solution.

The bioκ-calculus is simple. It has atomic elements – the proteins – and two syntactic constructors – grouping of solutions and cells. Reactions are interactions between two proteins or between two membranes. Notwithstanding this simplicity, the calculus is expressive enough. It is possible to describe significant biological examples: the RTK-MAPK pathway and a virus infection. This simplicity is also crucial for providing a faithful implementation of core bioκ-calculus into π-calculus. The compilation proposed in this paper encodes complexations and decomplexations into exactly one π-calculus interaction; conversely, every interaction corresponds exactly to one reaction.

The contribution is structured as follows. In the next section we define a core version of bioκ-calculus where reactions are restricted to be between two proteins. We study the semantics of core bioκ-calculus and analyse a signal transduction mechanism. In Section 3 we define an extensional semantics for bioκ-calculus and study its properties. In Section 4 we extend the basic interaction mechanisms with fusions that make structural modifications in the hierarchical organisation of a system. In Section 5 we discuss other extensions of the calculus for modelling translocations and phagocytosis. In Section 6 we study an implementation of core bioκ-calculus in π-calculus. Section 7 draws few concluding remarks and hints at possible future works.

2 The core of the bioκ-calculus

In this section we present a core version of the bioκ-calculus where interactions never change the hierarchical organisation of biological solutions. We define its syntax, its operational semantics, and the weak bisimulation. We also analyse its expressiveness by encoding the RTK-MAPK pathway.

Three disjoint countable sets of names will be used in bioκ-calculus. A set of *protein names* P, ranged over by A, B, C, ... to denote the *type* of a protein; a set of *edge names* E, ranged over by x, y, z, ... to describe the bonds between proteins; a set of *membrane names* M, ranged over by M, N, ..., to denote the type of a membrane. Protein names are sorted according to the number of *sites* they possess. Let $\mathfrak{s}(\cdot)$ be the function yielding the sites of proteins. The sites of a protein A are indicated by the natural numbers in the set $\{1, \cdots, \mathfrak{s}(\mathrm{A})\}$.

Sites may be in three possible states: *bound* to another site, sharing then an element in E with this site, *visible*, i.e. not connected to other sites, or *hidden*, i.e. not connected to other sites but not useful for interactions. Let s be a site, we write s^x if it is connected through the name x, $\bar{s}$ if it is hidden, and simply s if it is visible. Formally, the state of sites are defined by maps, called *interfaces*. Interfaces, ranged over by σ, σ', ..., are partial functions from naturals to $\mathsf{E} \cup \{v, h\}$ (we are assuming that $v, h \notin \mathsf{E}$). For instance, $[1 \mapsto x; 2 \mapsto v; 3 \mapsto h]$ is an interface. In order to simplify the reading, we write this map as $1^x + 2 + \overline{3}$. In the following, when we write $\sigma + \sigma'$ we assume that the domains of σ and σ' are disjoint. We always assume that interfaces are injective on E. The sites of a protein A are completely defined by interfaces that are *total* on $\{1, \cdots, \mathfrak{s}(\mathrm{A})\}$.

The following figure illustrates the all the possible state of protein domain and the notation for the complexes:

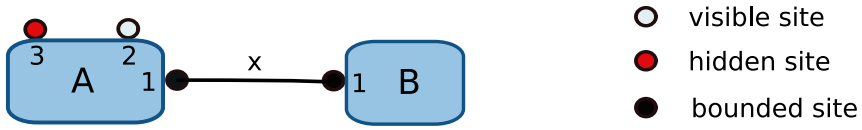

Fig. 1. Graphical notation for the complex $\mathrm{A}(1^x + 2 + \overline{3})$, $\mathrm{B}(1^x)$

Biological reactions modify interfaces, thus altering proteins' capabilities. In addition to the creation or suppression of a bond between proteins, reactions may also modify sites that are either hidden or visible to visible or hidden, respectively. In order to model these changes, let a *v-h-map*, ranged over $\phi, \psi, \cdots$, be every partial interface onto $\{v, h\}$. Let the *swap* of a v-h-map ϕ, written $\overline{\phi}$, be the v-h-map:

$$\overline{\phi}(i) \;=\; \begin{cases} h & \text{if } \phi(i) = v \\ v & \text{if } \phi(i) = h \end{cases}$$

Definition 1 *The syntax of the core-*bioκ*-calculus defines (biological)* solu-

tions S *by the grammar:*

$$\mathsf{S},\mathsf{T} \;::=\; \underset{\textit{(empty)}}{\mathbf{0}} \;\mid\; \underset{\textit{(protein)}}{\mathrm{A}(\sigma)} \;\mid\; \underset{\textit{(cell)}}{\mathrm{M}(\!|\,\mathsf{S}\,|\!)[\mathsf{T}]} \;\mid\; \underset{\textit{(group)}}{\mathsf{S}\,,\,\mathsf{T}}$$

Solutions can be either empty, or a *protein* $\mathrm{A}(\sigma)$ indicating a protein name and its interface, or a *cell* $\mathrm{M}(\!|\,\mathsf{S}\,|\!)[\mathsf{T}]$, that is a solution T, called *cytoplasm* [1], surrounded by another solution S, called *membrane*, or a *group* of solutions S, T. The following figure illustrates the syntactical notation for a cell:

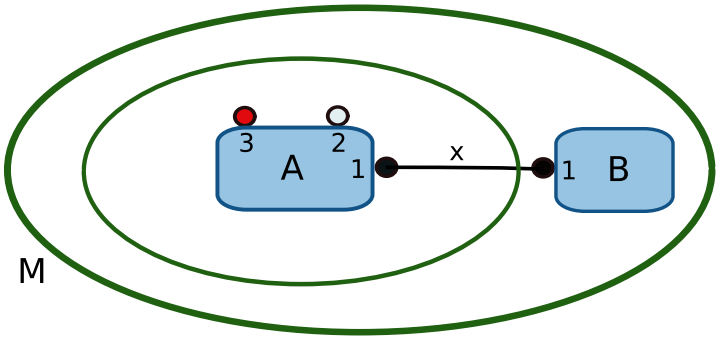

Fig. 2. Graphical notation for the cell $\mathrm{M}(\!|\,\mathrm{B}(1^x)\,|\!)[\mathrm{A}(1^x+2+\overline{3})]$

Three auxiliary functions will be applied to solutions and interfaces:

$\mathtt{en}(\cdot)$ returns the set of *edge names* occurring in the argument;
$\mathtt{de}(\cdot)$ returns the set of *dangling edge names* of the argument, namely those names that occur exactly once;
$\mathtt{be}(\cdot)$ returns the set of *bound edge names* of the argument, namely those names that occur at least twice.

Clearly $\mathtt{de}(\mathsf{S}) = \mathtt{en}(\mathsf{S}) \setminus \mathtt{be}(\mathsf{S})$ and similarly for σ. For instance, in $\mathsf{S} = \mathrm{M}(\!|\,\mathrm{C}(1^y+2)\,|\!)[\mathrm{A}(1^x+2+3), \mathrm{B}(1+2^x)]$ the set $\mathtt{en}(\mathsf{S})$ is $\{y,x\}$ and the set $\mathtt{de}(\mathsf{S})$ is $\{y\}$. We abbreviate the group $\mathrm{A}_1(\sigma_1)\,,\,\cdots\,,\,\mathrm{A}_n(\sigma_n)$ with $\prod_{i\in 1..n}\mathrm{A}_i(\sigma_i)$.

In the whole paper, we identify solutions that are equal up to a renaming of edge names that are not dangling (called alpha-conversion) and we assume that all solutions meet the following *well-formedness conditions*:

- (*edge-condition)* in every solution, edge names occur at most twice;
- (*membrane-condition)* every membrane is a group of proteins, that is cells do not occur in membranes;

[1] With the term "cytoplasm" we refer to every material surrounded by a structure forbidding any interaction with the external environment. Thus, a nucleus of a cell is a cytoplasm, as well as a DNA strand surrounded by a capsid.

- (*cytoplasm-condition)* the dangling edges of nuclei of cells are connected to the corresponding membrane, that is, for every $\mathrm{M}(\!|\, \mathsf{S} \,|\!)[\mathsf{T}]$, $\mathtt{de}(\mathsf{T}) \subseteq \mathtt{de}(\mathsf{S})$.

For example $\mathrm{M}(\!|\, \mathrm{C}(1^x + 2) \,|\!)[\mathrm{A}(1^x + 2 + 3), \mathrm{B}(1 + 2^x)]$ does not meet the edge condition because the edge x has three ends (it is a multi-edge). The solution $\mathrm{M}(\!|\, \mathrm{B}(1+2) \,,\, \mathrm{C}(1+2) \,|\!)[\mathrm{A}(1+2^x+3)]$ does not meet the cytoplasm condition because the cytoplasm $\mathrm{A}(1 + 2^x + 3)$ has a dangling edge that is not connected to the membrane. In the following, solutions that are membranes will be addressed by $\mathsf{M}, \mathsf{N}, \cdots$.

2.2 Semantics

Biological reactions that we consider in this section are of two types: *complexations*, which create edges between possibly disconnected proteins, and *decomplexations*, which remove edges. For instance a complexation reaction is (we are assuming $\mathfrak{s}(\mathrm{A}) = 4$, $\mathfrak{s}(\mathrm{B}) = 3$)

$$\mathrm{A}(1^x + 2 + \overline{3} + 4) \,,\, \mathrm{B}(1 + \overline{2} + \overline{3}) \longrightarrow \mathrm{A}(1^x + 2^y + 3 + 4) \,,\, \mathrm{B}(1^y + 2 + \overline{3})$$

that creates an edge y connecting the site 2 of A and the site 1 of B. These two sites, in order that this reaction be executed, must be visible. This means that the application of a complexation must check whether the sites being connected are visible or not. For example the above reaction cannot be applied to the group $\mathrm{A}(1^x + \overline{2} + \overline{3} + 4) \,,\, \mathrm{B}(1 + \overline{2} + \overline{3})$ because the site 2 of A is hidden. Reactions in $\mathtt{bio}\kappa$-calculus may also change the state of sites that are visible or hidden in the reactants, switching them into hidden and visible, respectively. In the example above, this happens to the sites 3 of A and 2 of B. This partial interface is then used as a test to check whether the interactions is possible and at the same time, it is used to describe the effect of the interaction on the internal sites of the proteins. These serve however two different purposes. Then it seems natural to add the possibility to describe a part of the interface used as a test but keeping unchanged by the interaction. This is the case of the site 4 of A if we suppose that the interaction is impossible as soon as this site is hidden. A concise way for defining the above reaction is the schema

$$\mathtt{r}: \quad ((\mathrm{A}, 2, \overline{3}, 4), (\mathrm{B}, 1, \overline{2}, \emptyset))$$

that makes explicit the reactant proteins – the first items of the triples –, the corresponding sites to be complexated – the second items – and the part of the interface which is tested, separating the sub-part which is permuted from the one which keeps unchanged – third and fourth item. For example, the rule $\mathtt{r}$ also applies to $\mathrm{A}(1+2+\overline{3}+4) \,,\, \mathrm{B}(1+\overline{2}+3)$ or $\mathrm{A}(\overline{1}+2+\overline{3}+4) \,,\, \mathrm{B}(1+\overline{2}+3^x)$ yielding solutions $\mathrm{A}(1+2^y+3+4) \,,\, \mathrm{B}(1^y+2+3)$ and $\mathrm{A}(\overline{1}+2^y+3+4) \,,\, \mathrm{B}(1^y+$

$2 + 3^x$), respectively. In general, the shape of a reaction schema is

$$\mathtt{r}: \quad ((\textsc{a}, i, \psi_1, \psi_2), (\textsc{b}, j, \phi_1, \phi_2))$$

that is a reaction name $\mathtt{r}$ and two tuple containing a protein name, a site and two v-h-maps. A generic *application* of the schema $\mathtt{r}$ may be written as

$$\begin{gathered} \textsc{a}(i + \psi_1 + \psi_2 + \psi') \mathbin{,} \textsc{b}(j + \phi_1 + \phi_2 + \phi') \\ \downarrow \\ \textsc{a}(i^x + \overline{\psi_1} + \psi_2 + \psi') \mathbin{,} \textsc{b}(j^x + \overline{\phi_1} + \phi_2 + \phi') \end{gathered}$$

where x is a fresh edge name and $i+\psi_1+\psi_2+\psi$ and $j+\phi_1+\phi_2+\phi$ are total on $[1 \mathinner{..} \mathfrak{s}(\textsc{a})]$ and $[1 \mathinner{..} \mathfrak{s}(\textsc{b})]$, respectively. It is worth to observe that the interfaces σ and σ' are not mentioned in the schema. This means that the interaction occurs *what ever* the states of the sites of these interfaces are.

Decomplexations are complexations in the other way round. For instance a decomplexation reaction is

$$\textsc{a}(1^x + 2^y + 3 + 4) \mathbin{,} \textsc{b}(1^y + 2 + \overline{3}) \longrightarrow \textsc{a}(1^x + 2 + \overline{3} + 4) \mathbin{,} \textsc{b}(1 + \overline{2} + \overline{3})$$

that removes the edge y. The schema describing decomplexations is similar to that of complexations:

$$\mathtt{r}': \quad ((\textsc{a}, i, \psi_1, \psi_2), (\textsc{b}, j, \phi_1, \phi_2))$$

The application of the decomplexation rule is different from complexation: in this case the two reactants must be connected by an edge between the site i of A and the site j of B. So, for example, a generic application of $\mathtt{r}'$ is

$$\begin{gathered} \textsc{a}(i^x + \psi_1 + \psi_2 + \psi') \mathbin{,} \textsc{b}(j^x + \phi_1 + \phi_2 + \phi') \\ \downarrow \\ \textsc{a}(i + \overline{\psi_1} + \psi_2 + \psi') \mathbin{,} \textsc{b}(j + \overline{\phi_1} + \phi_2 + \phi') \end{gathered}$$

In order to separate complexations from decomplexations we consider two functions, $\mathcal{C}$ for *complexations* and $\mathcal{D}$ for *decomplexations*, from rule names to tuples $((\textsc{a}, i, \phi_1, \phi_2), (\textsc{b}, j, \psi, \psi_2))$. These functions $\mathcal{C}$ and $\mathcal{D}$ are assumed with disjoint domains, therefore a rule name uniquely defines whether it is a complexation or a decomplexation. Let $\mathcal{R}$ range over $\mathcal{C}$ and $\mathcal{D}$ and let also write $(\textsc{a}, i, \phi_1, \phi_2) \in_{\text{left}} \mathcal{R}(\mathtt{r})$ if $\mathcal{R}(\mathtt{r}) = ((\textsc{a}, i, \phi_1, \phi_2), (\textsc{b}, j, \psi_1, \psi_2))$ and $(\textsc{a}, i, \phi_1, \phi_2) \in_{\text{right}} \mathcal{R}(\mathtt{r})$ if $\mathcal{R}(\mathtt{r}) = ((\textsc{b}, j, \psi_1, \psi_2), (\textsc{a}, i, \phi_1, \phi_2))$. Finally, let also $(\textsc{a}, i, \phi_1, \phi_2) \in \mathcal{R}(\mathtt{r})$ if either $(\textsc{a}, i, \phi_1, \phi_2) \in_{\text{left}} \mathcal{R}(\mathtt{r})$ or $(\textsc{a}, i, \phi_1, \phi_2) \in_{\text{right}} \mathcal{R}(\mathtt{r})$.

The operational semantics of core bioκ-calculus we are going to define use labels. A label is either a triple $(\mathrm{A}, x, \mathtt{r})$, also written $\mathrm{A}^x_{\mathtt{r}}$, where A is a protein name, x an edge name and $\mathtt{r}$ a rule name, or τ. We use μ to range over labels. Let $\mathtt{diff}(\mathsf{S}, \mathsf{S}')$ be the set $(\mathtt{en}(\mathsf{S}')\setminus\mathtt{en}(\mathsf{S})) \cup (\mathtt{en}(\mathsf{S})\setminus\mathtt{en}(\mathsf{S}'))$.

Definition 2 *The* transition relation $\mathsf{S} \xrightarrow{\mu} \mathsf{T}$ *is the least relation satisfying the following rules:*

$$\text{(COM)}\quad \frac{(\mathrm{A}, i, \phi_1, \phi_2) \in \mathcal{C}(\mathtt{r}) \quad \mathtt{x} \notin \mathtt{en}(\sigma)}{\mathrm{A}(i + \phi_1 + \phi_2 + \sigma) \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathrm{A}(i^x + \overline{\phi_1} + \phi_2 + \sigma)}$$

$$\text{(DEC)}\quad \frac{(\mathrm{A}, i, \phi_1, \phi_2) \in \mathcal{D}(\mathtt{r})}{\mathrm{A}(i^x + \phi_1 + \phi_2 + \sigma) \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathrm{A}(i + \overline{\phi_1} + \phi_2 + \sigma)}$$

$$\text{(SOL-L)}\quad \frac{\mathsf{S} \xrightarrow{\mu} \mathsf{S}' \quad \mathtt{diff}(\mathsf{S}, \mathsf{S}') \cap \mathtt{en}(\mathsf{T}) = \emptyset}{\mathsf{S}\ ,\ \mathsf{T} \xrightarrow{\mu} \mathsf{S}'\ ,\ \mathsf{T}} \qquad \text{(SOL-R)}\quad \frac{\mathsf{T} \xrightarrow{\mu} \mathsf{T}' \quad \mathtt{diff}(\mathsf{T}, \mathsf{T}') \cap \mathtt{en}(\mathsf{S}) = \emptyset}{\mathsf{S}\ ,\ \mathsf{T} \xrightarrow{\mu} \mathsf{S}\ ,\ \mathsf{T}'}$$

$$\text{(MEM)}\quad \frac{\mathsf{M} \xrightarrow{\mu} \mathsf{M}' \quad \mathtt{diff}(\mathsf{M}, \mathsf{M}') \cap \mathtt{en}(\mathsf{S}) = \emptyset}{\mathrm{M}\langle\!|\, \mathsf{M} \,|\!\rangle[\mathsf{S}] \xrightarrow{\mu} \mathrm{M}\langle\!|\, \mathsf{M}' \,|\!\rangle[\mathsf{S}]} \qquad \text{(CYTO)}\quad \frac{\mathsf{S} \xrightarrow{\tau} \mathsf{S}' \quad \mathtt{diff}(\mathsf{S}, \mathsf{S}') \cap \mathtt{en}(\mathsf{M}) = \emptyset}{\mathrm{M}\langle\!|\, \mathsf{M} \,|\!\rangle[\mathsf{S}] \xrightarrow{\tau} \mathrm{M}\langle\!|\, \mathsf{M} \,|\!\rangle[\mathsf{S}']}$$

$$\text{(REACT)}\quad \frac{\mathsf{S} \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathsf{S}' \quad \mathsf{T} \xrightarrow{\mathrm{B}^x_{\mathtt{r}}} \mathsf{T}' \quad \mathrm{A} \neq \mathrm{B}}{\mathsf{S}\ ,\ \mathsf{T} \xrightarrow{\tau} \mathsf{S}'\ ,\ \mathsf{T}'} \qquad \text{(MS-REACT)}\quad \frac{\mathsf{M} \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathsf{M}' \quad \mathsf{S} \xrightarrow{\mathrm{B}^x_{\mathtt{r}}} \mathsf{S}' \quad \mathrm{A} \neq \mathrm{B}}{\mathrm{M}\langle\!|\, \mathsf{M} \,|\!\rangle[\mathsf{S}] \xrightarrow{\tau} \mathrm{M}\langle\!|\, \mathsf{M}' \,|\!\rangle[\mathsf{S}']}$$

Rules (COM) and (DEC) respectively define complexations and decomplexations capabilities of proteins by lifting these information to labels of transitions and, at the same time, updating the proteins. Rules (SOL-L), (SOL-R) and (MEM) lift transitions to groups and membranes; it is crucial that edge names created or deleted do not occur elsewhere. Rule (CYTO) lift *internal* transitions of the cytoplasm to the whole cell; as before, edge names created or deleted must not occur elsewhere. We observe that (CYTO) bans complexation or decomplexation between cytoplasms and the solution external to the cells. Rule (REACT) and (MS-REACT) define reactions, both complexations and decomplexations, in groups and cells. In particular (REACT) also accounts for reactions between membranes of different cells. It is worth to notice that the constraint $\mathrm{A} \neq \mathrm{B}$ bans reactions between same proteins (*cf. self complexation* in [3]). This is for simplicity sake: in case reactants are proteins with a same name we need to carry more information on the labels to separate them.

It is worth to notice that the semantics of bioκ-calculus adhere to a style that is different than the one used for defining the transitions of mκ-calculus in [3]. The mκ-calculus has been equipped with an *unlabelled* reduction relation $\rightarrow$. This relation corresponds to the foregoing rules (COM), (DEC), (SOL-G), (SOL-D), and (REACT). We have preferred a labelled transition relation for simplicity sake: the number of rules for defining a relation $\rightarrow$ equivalent to $\xrightarrow{\mu}$ should have been larger than that of Definition 2 because of the presence of membranes.

The transition relation preserve well-formedness of solutions.

Proposition 1 *If* S *is well-formed and* $\mathsf{S} \xrightarrow{\mu} \mathsf{T}$ *then* T *is well-formed as well.*

It is worth to observe that membrane names do not play any role at this stage. They will be relevant in the complete system with complex membrane reactions presented in Section 4.

2.3 Modelisation of the signal transduction

The so-called RTK-MAPK pathway are intensely used and studied in many approaches modelling and simulating biological systems [2,3]. We therefore model the first steps of such a pathway in bioκ-calculus, thus providing a touchstone for our calculus.

The *receptors tyrosine kinase* (RTKs) are receptors located at the surface of some cell membranes. After binding with some ligand (insulin, EGF, VEGF, etc ...), they are activated and lead to a specific cellular response (growth of the cell, proliferation, etc ...). There exists actually many different types of RTKs which are gathered into different families. They don't provide the same reaction and the cellular response depends on the type of RTK which is activated. After the binding with a ligand, the RTKs may *self-phosphorylate* (adding a phosphate group to a binding domain). This modification is detected by specific proteins which will bind to the RTK and recruit other proteins which will be activated or inhibited and will lead eventually to the cellular response. Another possibility for the RTKs to mediate their reaction is to phosphorylate other proteins (besides self-phosphorylation). These activated proteins lead to a cascade of signalisations too. This cascade results mainly in the activation or the inhibition of transcription factors which adjusts gene expressions. The generic schema of an RTK-MAPK cascade is then the following: the RTKs, after binding with a ligand (the signal), induce some changes into the cell (ex. gene expression) in order to mediate a biological effect (ex. entering the cell cycle).

The example we are developing here is that of the response induced by the

signal carried by an epidermic growth factor protein (EGF). The dimeric form (1) of EGF can bind to its associated receptor EGFR (2), a transmembrane protein with an extracellular ligand-binding domain located on the plasmic membrane of some cells. This binding activates EGFR by phosphorylating an internal domain of the protein (3 and 4). This activation leads to multiple interactions with cytoplasmic complexes of proteins by successive binding-phosphorylations, starting with the adapter protein SHC (5). The cascade of interactions ends with the activation of the extracellular signal-regulated kinase ERK. Once phosphorylated, this protein can then be transported into the nucleus by means of a translocation mechanism and will thereafter modify the gene expression, stimulating the cell to enter mitosis. This causes the cell to divide and proliferate.

After the biological description EGF, RTK, and SHC have respective arities 3, 4, and 2. We give here the formal rendering of the five first steps described above:

$$\begin{aligned}
\mathtt{r}_1 &: ((\text{EGF}, 1, \overline{2}, \emptyset), (\text{EGF}, 1, \overline{2}, \emptyset)) \in \mathcal{C} \\
\mathtt{r}_2 &: ((\text{EGF}, 2, \emptyset, \emptyset), (\text{EGFR}, 1, \overline{4}, \emptyset)) \in \mathcal{C} \\
\mathtt{r}_3 &: ((\text{EGFR}, 2, \overline{3}+4, \emptyset), (\text{EGFR}, 2, \overline{3}+4, \emptyset)) \in \mathcal{C} \\
\mathtt{r}_4 &: ((\text{EGFR}, 2, \emptyset, \emptyset), (\text{EGFR}, 2, \emptyset, \emptyset)) \in \mathcal{D} \\
\mathtt{r}_5 &: ((\text{EGFR}, 3, \emptyset, \emptyset), (\text{SHC}, 1, \overline{2}, \emptyset)) \in \mathcal{C}
\end{aligned}$$

A possible run of those rules is presented in Figure 3, showing that our language is expressive enough to define the inherent causality involved in the transduction in a precise yet natural way.

A couple of problems of our notation deserve to be discussed though. Consider the initial solution:

$$\begin{aligned}
&\text{EGF}(1+\overline{2})\ ,\ \text{EGF}(1+\overline{2})\ ,\ \text{EGF}(1+\overline{2})\ ,\ \text{EGF}(1+\overline{2})\ , \\
&\qquad \text{M}(\!|\ \text{EGFR}(1+2^x+\overline{3}+\overline{4})\ ,\ \text{EGFR}(1+2^x+\overline{3}+\overline{4})\ ,\ \mathsf{M}\ |\!)[\text{SHC}(1+\overline{2})\ ,\ \mathsf{S}]
\end{aligned}$$

After two applications of rule $\mathtt{r}_1$, we obtain the solution

$$\begin{aligned}
&\text{EGF}(1^x+2)\ ,\ \text{EGF}(1^x+2)\ ,\ \text{EGF}(1^y+2)\ ,\ \text{EGF}(1^y+2)\ , \\
&\qquad \text{M}(\!|\ \text{EGFR}(1+2^x+\overline{3}+\overline{4})\ ,\ \text{EGFR}(1+2^x+\overline{3}+\overline{4})\ ,\ \mathsf{M}\ |\!)[\text{SHC}(1+\overline{2})\ ,\ \mathsf{S}]
\end{aligned}$$

that reduces, after two application of rule $\mathtt{r}_2$ to the wrong solution

$$\begin{aligned}
&\text{EGF}(1^x+2)\ ,\ \text{EGF}(1^x+2^u)\ ,\ \text{EGF}(1^y+2^v)\ ,\ \text{EGF}(1^y+2)\ , \\
&\qquad \text{M}(\!|\ \text{EGFR}(1^u+2^x+\overline{3}+4)\ ,\ \text{EGFR}(1^v+2^x+\overline{3}+4)\ ,\ \mathsf{M}\ |\!)[\text{SHC}(1+\overline{2})\ ,\ \mathsf{S}]
\end{aligned}$$

$$
\begin{aligned}
&\textsc{egf}(1+\overline{2})\ ,\ \textsc{egf}(1+\overline{2})\ ,\\
&\qquad \textsc{m}\big(\!\big|\ \textsc{egfr}(1+2^{+}\overline{3}+\overline{4})\ ,\ \textsc{egfr}(1+2^{+}\overline{3}+\overline{4})\ ,\ \mathsf{M}\ \big|\!\big)[\textsc{shc}(1+\overline{2})\ ,\ \mathsf{S}]\\
&\xrightarrow{\tau}\ \textsc{egf}(1^{z}+2)\ ,\ \textsc{egf}(1^{z}+2)\ ,\\
&\qquad \textsc{m}\big(\!\big|\ \textsc{egfr}(1+2+\overline{3}+\overline{4})\ ,\ \textsc{egfr}(1+2+\overline{3}+\overline{4})\ ,\ \mathsf{M}\ \big|\!\big)[\textsc{shc}(1+\overline{2})\ ,\ \mathsf{S}] \qquad (\mathtt{r}_1)\\
&\xrightarrow{\tau}\ \textsc{egf}(1^{z}+2^{y})\ ,\ \textsc{egf}(1^{z}+2)\ ,\\
&\qquad \textsc{m}\big(\!\big|\ \textsc{egfr}(1^{y}+2+\overline{3}+4)\ ,\ \textsc{egfr}(1+2+\overline{3}+\overline{4})\ ,\ \mathsf{M}\ \big|\!\big)[\textsc{shc}(1+\overline{2})\ ,\ \mathsf{S}] \qquad (\mathtt{r}_2)\\
&\xrightarrow{\tau}\ \textsc{egf}(1^{z}+2^{y})\ ,\ \textsc{egf}(1^{z}+2^{u})\ ,\\
&\qquad \textsc{m}\big(\!\big|\ \textsc{egfr}(1^{y}+2+\overline{3}+4)\ ,\ \textsc{egfr}(1^{u}+2+\overline{3}+4)\ ,\ \mathsf{M}\ \big|\!\big)[\textsc{shc}(1+\overline{2})\ ,\ \mathsf{S}] \qquad (\mathtt{r}_2)\\
&\xrightarrow{\tau}\ \textsc{egf}(1^{z}+2^{y})\ ,\ \textsc{egf}(1^{z}+2^{u})\ ,\\
&\qquad \textsc{m}\big(\!\big|\ \textsc{egfr}(1^{y}+2^{x}+3+\overline{4})\ ,\ \textsc{egfr}(1^{u}+2^{x}+3+\overline{4})\ ,\ \mathsf{M}\ \big|\!\big)[\textsc{shc}(1+\overline{2})\ ,\ \mathsf{S}] \qquad (\mathtt{r}_3)\\
&\xrightarrow{\tau}\ \textsc{egf}(1^{z}+2^{y})\ ,\ \textsc{egf}(1^{z}+2^{u})\ ,\\
&\qquad \textsc{m}\big(\!\big|\ \textsc{egfr}(1^{y}+2+3+\overline{4})\ ,\ \textsc{egfr}(1^{u}+2+3+\overline{4})\ ,\ \mathsf{M}\ \big|\!\big)[\textsc{shc}(1+\overline{2})\ ,\ \mathsf{S}] \qquad (\mathtt{r}_4)\\
&\xrightarrow{\tau}\ \textsc{egf}(1^{z}+2^{y})\ ,\ \textsc{egf}(1^{z}+2^{u})\ ,\\
&\qquad \textsc{m}\big(\!\big|\ \textsc{egfr}(1^{y}+2+3^{t}+\overline{4})\ ,\ \textsc{egfr}(1^{u}+2+3+\overline{4})\ ,\ \mathsf{M}\ \big|\!\big)[\textsc{shc}(1^{t}+2)\ ,\ \mathsf{S}] \qquad (\mathtt{r}_5)
\end{aligned}
$$

Fig. 3. The first steps of the RTK-MAPK cascade.

where two different dimeric forms of EGF connect to a same pair of EGFR receptors. Our notation is too simple to rule out such configurations. In $\mathtt{m}\kappa$-calculus, this expressiveness issue is solved by the use of reaction IDs and pattern matching over them. Actually, this issue is related to a more general question named self-assembly problem and worth to be studied independently [9].

The second problem is manifested at the end of the RTK-MAPK pathway. The pathway causes a phosphorylation of a particular protein (ERK) that enters in the nucleus, which is represented as cell, as well. At this stage we have no mechanism that make entities enter in the cell. Such mechanisms will be discussed in detail in Section 5.

3 Extensional semantics: weak bisimulation

The transition relation of Definition 2 associates solutions to graphs where nodes are solutions and μ-labelled edges model the transitions $\mathsf{S} \xrightarrow{\mu} \mathsf{T}$. The induced equivalence on $\mathtt{bio}\kappa$-calculus is graph isomorphism: two terms are equivalent provided their associated graphs are isomorphic. Graph isomorphism is however too strong as a biological semantics because it dis-

tinguishes solutions that differ for τ-transitions. For instance, let $\mathcal{C}(\mathtt{r}) = ((\text{A}, 1, \emptyset), (\text{B}, 1, \emptyset)) = \mathcal{D}(\mathtt{r}')$, that is $\mathtt{r}$ and $\mathtt{r}'$ are *reversible* reactions. Then the solutions $\text{A}(1^y + \sigma)$, $\text{B}(1^y + \sigma')$ and $\text{A}(1 + \sigma)$, $\text{B}(1 + \sigma')$ have underlying graphs that are not isomorphic. The failure of the isomorphism is due to τ-transitions: every solution may be reduced to the other by means of a τ-transition.

The following equivalence corrects the above criticism. It adapts *weak bisimulation* [1] to our setting. We write $\mathsf{S} \stackrel{\tau}{\Longrightarrow} \mathsf{S}'$ if $\mathsf{S} \stackrel{\tau}{\longrightarrow}^* \mathsf{S}'$ and $\mathsf{S} \stackrel{\mu}{\Longrightarrow} \mathsf{S}'$, with $\mu \neq \tau$, if $\mathsf{S} \stackrel{\tau}{\longrightarrow}^* \stackrel{\mu}{\longrightarrow} \stackrel{\tau}{\longrightarrow}^* \mathsf{S}'$.

Definition 3 *A* (weak) bisimulation *is a symmetric binary relation* $\mathfrak{R}$ *between solutions such that* $\mathsf{S}\ \mathfrak{R}\ \mathsf{T}$ *implies:*

(1) if $\mathsf{S} \stackrel{\tau}{\longrightarrow} \mathsf{S}'$ *then* $\mathsf{T} \stackrel{\tau}{\Longrightarrow} \mathsf{T}'$ *and* $\mathsf{S}'\ \mathfrak{R}\ \mathsf{T}'$*;*
(2) if $\mathsf{S} \stackrel{\text{A}^x_{\mathtt{r}}}{\longrightarrow} \mathsf{S}'$ *then* $\mathsf{T} \stackrel{\text{A}^x_{\mathtt{r}}}{\Longrightarrow} \mathsf{T}'$ *and* $\mathsf{S}'\ \mathfrak{R}\ \mathsf{T}'$*.*

S *is bisimilar to* T*, written* $\mathsf{S} \approx \mathsf{T}$*, if* $\mathsf{S}\ \mathfrak{R}\ \mathsf{T}$ *for some bisimulation* $\mathfrak{R}$*.*

With this notion of equivalence, it is possible to show the following properties for the core-$\mathtt{bio}\kappa$-calculus:

Proposition 2 *(1) " , " is an abelian monoidal operator with identity* $\mathbf{0}$*. Namely* $\mathsf{S} , \mathsf{T} \approx \mathsf{T} , \mathsf{S}$ *and* $(\mathsf{S} , \mathsf{T}) , \mathsf{R} \approx \mathsf{S} , (\mathsf{T} , \mathsf{R})$ *and* $\mathsf{S} , \mathbf{0} \approx \mathsf{S}$*.*
(2) $\approx$ *is preserved by injective renamings that are identities on dangling edge names. Namely, let* ι *be an injective renaming on* $\mathtt{en}(\mathsf{S})$ *such that* ι *is the identity on* $\mathtt{de}(\mathsf{S})$*, then* $\mathsf{S} \approx \iota(\mathsf{S})$*.*
(3) $\approx$ *is preserved by reversible rules. Namely, let* $\mathcal{C}(\mathtt{r}) = ((\text{A}, i, \psi), (\text{B}, j, \phi))$ *and* $\mathcal{D}(\mathtt{r}') = ((\text{A}, i, \overline{\psi}), (\text{B}, j, \overline{\phi}))$ *then* $\text{A}(i + \psi + \sigma)$, $\text{B}(j + \phi + \sigma') \approx \text{A}(i^x + \overline{\psi} + \sigma)$, $\text{B}(j^x + \overline{\phi} + \sigma')$*.*

Another relevant property of $\approx$ is that every two bisimilar systems behave in the same way when plugged in a same context.

Theorem 1 $\approx$ *is a congruence.*

Proof: *We must prove that, if* $\mathsf{S} \approx \mathsf{T}$ *then*

(1) $\mathsf{S} , \mathsf{R} \approx \mathsf{T} , \mathsf{R}$*,*
(2) $\text{M}(\!|\, \mathsf{S} \,|\!)[\mathsf{R}] \approx \text{M}(\!|\, \mathsf{T} \,|\!)[\mathsf{R}]$
(3) $\text{M}(\!|\, \mathsf{M} \,|\!)[\mathsf{S}] \approx \text{M}(\!|\, \mathsf{M} \,|\!)[\mathsf{T}]$

when these solutions are well-formed. We demonstrate (1), the other cases are similar. Let $\mathfrak{R}$ *be the bisimulation containing the pair* (S, T) *and let* $(\mathsf{S}' , \mathsf{R})\ \mathfrak{R}'\ (\mathsf{T}' , \mathsf{R})$ *if*

(1) $\mathsf{S}' \mathfrak{R} \mathsf{T}'$,

(2) and $\mathsf{S}' \mathbin{,} \mathsf{R}$ *and* $\mathsf{T}' \mathbin{,} \mathsf{R}$ *are well-formed.*

To demonstrate that $\mathfrak{R}'$ *is a bisimulation one has to prove that if* $\mathsf{S}' \mathbin{,} \mathsf{R} \xrightarrow{\mu} \mathsf{U}$ *then there exists* U' *such that* $\mathsf{T}' \mathbin{,} \mathsf{R} \stackrel{\mu}{\Longrightarrow} \mathsf{U}'$ *and* $\mathsf{U} \mathfrak{R}' \mathsf{U}'$. *There are two cases:*

$\mu = \mathrm{A}^x_{\mathtt{r}}$. *Then either* $\mathsf{U} = \mathsf{S}' \mathbin{,} \mathsf{R}'$ *with* $\mathsf{R} \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathsf{R}'$ *or* $\mathsf{U} = \mathsf{S}'' \mathbin{,} \mathsf{R}$ *with* $\mathsf{S}' \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathsf{S}''$. *The first case is immediate. In the second case,* $\mathsf{T}' \stackrel{\mathrm{A}^x_{\mathtt{r}}}{\Longrightarrow} \mathsf{T}''$ *and* $\mathsf{S}'' \mathfrak{R} \mathsf{T}''$. *By definition:* $(\mathsf{S}'' \mathbin{,} \mathsf{R}) \mathfrak{R}' (\mathsf{T}'' \mathbin{,} \mathsf{R})$.

$\mu = \tau$. *The interesting case is when* S' *and* R *interact (the other cases are dealt as above) by the rule (*REACT*). Suppose* $\mathsf{U} = \mathsf{S}'' \mathbin{,} \mathsf{R}'$ *with* $\mathsf{S}' \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathsf{S}''$ *and* $\mathsf{R} \xrightarrow{\mathrm{B}^x_{\mathtt{r}}} \mathsf{R}'$. *Since* $\mathsf{S}' \mathfrak{R} \mathsf{T}'$, *there exists* T'' *such that* $\mathsf{T}' \stackrel{\mathrm{A}^x_{\mathtt{r}}}{\Longrightarrow} \mathsf{T}''$ *and* $\mathsf{S}'' \mathfrak{R} \mathsf{T}''$. *Then, by (*REACT*),* $\mathsf{T}' \mathbin{,} \mathsf{R} \stackrel{\tau}{\Longrightarrow} \mathsf{T}'' \mathbin{,} \mathsf{R}'$ *and* $(\mathsf{S}'' \mathbin{,} \mathsf{R}') \mathfrak{R}' (\mathsf{T}'' \mathbin{,} \mathsf{R}')$ *by definition of* $\mathfrak{R}'$.

The substitution property of Proposition 2 owned by $\approx$ might be too strong in biology, thus making this equivalence not realistic. In this context, one usually wants to prove that two parts behave in the same way when plugged *under a certain number of contexts*, rather than every possible one. Therefore, a semantics owning a parametric congruence property might fit better with biology. However what these parameters are and what are the properties owned by a "good" extensional semantics for biology remains unclear to us and is left as an open issue.

We finish this section by a series of remarks concerning $\approx$. It is worth noticing for example that $\mathsf{S} \approx \mathsf{T}$ does not imply $\mathtt{de}(\mathsf{S}) = \mathtt{de}(\mathsf{T})$. For two reasons: First of all, by bisimulation, $\mathsf{S} \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathsf{S}'$ may be matched by $\mathsf{T} \xrightarrow{\mathrm{A}^y_{\mathtt{r}}} \mathsf{T}'$ with $x \neq y$. Second, taking empty biological relations – i.e. $\mathcal{C} = \emptyset$ and $\mathcal{D} = \emptyset$ –, then $\mathrm{A}(1^x) \approx \mathbf{0}$ but their dangling edges are different. Nevertheless, one might establish a relation between subsets of dandling edges of two bisimilar solutions. Let $\mathtt{oe}(\mathsf{S})$, called the *observable edges*, be the set

$$\mathtt{oe} = \{x \mid \mathsf{S} \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathsf{S}' \text{ and } \mathtt{r} \text{ in domain of } \mathcal{D}\}$$

It is easy to prove that if $\mathsf{S} \approx \mathsf{T}$ then there is an injective renaming ι such that $\mathtt{oe}(\mathsf{S}) = \iota(\mathtt{oe}(\mathsf{T}))$.

Finally, as an illustration of Definition 3, one may note that a cell whose membrane cannot interact with elements outside is equivalent to the empty solution. Let M be a group of proteins. M is said to be *inert* if $\mathsf{M} \not\stackrel{\mathrm{A}^x_{\mathtt{r}}}{\Longrightarrow}$ for every $\mathrm{A}^x_{\mathtt{r}}$. It is possible to verify that, if M is inert, then $\mathrm{M}(\!|\, \mathsf{M} \,|\!)[\mathsf{S}] \approx \mathbf{0}$.

4 Cell interactions

The calculus of Section 2 is not very different from $\mathtt{m}\kappa$-calculus. Cells, in particulars, do not play any relevant role since their structure is preserved by the transition. In this section we explore an extension with brane primitives, thus being able to model merging and splitting of cells such as the following *endosomes fusion*:

$$\textsc{esm}(\!|\,\mathsf{M}\,|\!)[\mathsf{S}]\ ,\ \textsc{esm}(\!|\,\mathsf{N}\,|\!)[\mathsf{T}] \ \xrightarrow{\tau}\ \textsc{esm}(\!|\,\mathsf{M}\ ,\ \mathsf{N}\,|\!)[\mathsf{S}\ ,\ \mathsf{T}]$$

The extension of core $\mathtt{bio}\kappa$-calculus we are going to design retains higher-order mechanisms, following *higher-order π-calculus*, a similar extension already studied for π-calculus [10].

4.1 Mreagents

We begin by augmenting the syntax with membrane reagents:

Definition 4 *The syntax of the $\mathtt{bio}\kappa$-calculus defines the solutions S by the grammar:*

$$\begin{array}{lccccccc} \mathsf{M},\mathsf{S},\mathsf{T} ::= & \mathbf{0} & | & \textsc{a}(\sigma) & | & \textsc{m}(\!|\,\mathsf{M}\,|\!)[\mathsf{S}] & | & \mathsf{S}\ ,\ \mathsf{T} \quad | \wr \mathsf{M}\ ;\ \mathsf{S} \wr \cdot \mathsf{T} \\ & \textit{(empty)} & & \textit{(protein)} & & \textit{(cell)} & & \textit{(group)} \quad \textit{(mreagent)} \end{array}$$

An mreagent $\wr \mathsf{M}\ ;\ \mathsf{S} \wr \cdot \mathsf{T}$ is an intermediate (unstable) solution that is used for manifesting the capability of a membrane M, isolating a solution S from the environment T, to fuse with another membrane. This remark leads to consider the elements M and S as the components of a cell $\textsc{m}(\!|\,\mathsf{M}\,|\!)[\mathsf{S}]$ and, as such, the following well-formedness constraints are put on their structure:

- M is a multiset of proteins;
- $\mathtt{de}(\mathsf{S}) \subseteq \mathtt{de}(\mathsf{M})$, namely the dandling edges of S are connected to proteins located in the membrane M;
- mreagents do not contain other mreagents.

Two operations involve membranes: *fusions*, which put together in a unique cell two cytoplasms separated by two membranes, and *activations* which change the type of a membrane as a consequence of a complexation or a decomplexation. Fusions are defined by a function $\mathcal{F}$ from rule names to triples $((\textsc{m},\textsc{m}'),\textsc{n})$. We write $(\textsc{m} \otimes \textsc{m}',\textsc{n}) = \mathcal{F}(\mathtt{r})$ if either $\mathcal{F}(\mathtt{r}) = ((\textsc{m},\textsc{m}'),\textsc{n})$ or $\mathcal{F}(\mathtt{r}) = ((\textsc{m}',\textsc{m}),\textsc{n})$. We also write $\textsc{m} \in \mathcal{F}(\mathtt{r})$ if $(\textsc{m} \otimes \textsc{m}',\textsc{n}) = \mathcal{F}(\mathtt{r})$, for some $\textsc{m}'$ and $\textsc{n}$. We assume that the domains of $\mathcal{F}$, $\mathcal{C}$ and $\mathcal{D}$ are disjoint. Activations

are defined by a function $\mathcal{A}$ that takes pairs $(\mathrm{A}_{\mathtt{r}}, \mathrm{M})$ and returns membrane names. The intuition is that a reaction involving a transmembrane protein may activate the membrane it belongs to, thus preparing the cell to a possible fusion.

With abuse of notation, we use μ to also range over labels of the form $\mathrm{M}_{\mathtt{r}}$.

Definition 5 *The* transition relation $\xrightarrow{\mu}$ *is the least one that includes the rules in Definition 2 where (*MEM*) and (*MS-REACT*) have also the premise "*$(\mathrm{A}_{\mathtt{r}}, \mathrm{M})$ *not in the domain of* $\mathcal{A}$*" and the following reductions:*

(OPEN)
$$\frac{\mathrm{M} \in \mathcal{F}(\mathtt{r})}{\mathrm{M}(\!|\, \mathsf{M} \,|\!)[\mathsf{S}] \xrightarrow{\mathrm{M}_{\mathtt{r}}} \wr \mathsf{M} \,;\, \mathsf{S} \int \cdot \mathbf{0}}$$

(GRASP)
$$\frac{\mathsf{S} \xrightarrow{\mu} \wr \mathsf{M} \,;\, \mathsf{S}'' \int \cdot \mathsf{S}'}{\mathsf{S} \,,\, \mathsf{T} \xrightarrow{\mu} \wr \mathsf{M} \,;\, \mathsf{S}'' \int \cdot (\mathsf{S}' \,,\, \mathsf{T})}$$

(FUSE-H)
$$\frac{\mathsf{S} \xrightarrow{\mathrm{M}_{\mathtt{r}}} \wr \mathsf{M} \,;\, \mathsf{S}'' \int \cdot \mathsf{S}' \quad \mathsf{T} \xrightarrow{\mathrm{N}_{\mathtt{r}}} \wr \mathsf{N} \,;\, \mathsf{T}'' \int \cdot \mathsf{T}' \qquad \mathcal{F}(\mathtt{r}) = (\mathrm{M} \otimes \mathrm{N}, \mathrm{M}')}{\mathsf{S} \,,\, \mathsf{T} \xrightarrow{\tau} \mathsf{S}' \,,\, \mathsf{T}' \,,\, \mathrm{M}'(\!|\, \mathsf{M} \,,\, \mathsf{N} \,|\!)[\mathsf{S}'' \,,\, \mathsf{T}'']}$$

(FUSE-V)
$$\frac{\mathsf{S} \xrightarrow{\mathrm{N}_{\mathtt{r}}} \wr \mathsf{N} \,;\, \mathsf{T} \int \cdot \mathsf{S}' \qquad \mathcal{F}(\mathtt{r}) = (\mathrm{M} \otimes \mathrm{N}, \mathrm{M}')}{\mathrm{M}(\!|\, \mathsf{M} \,|\!)[\mathsf{S}] \xrightarrow{\tau} \mathrm{M}'(\!|\, \mathsf{M} \,,\, \mathsf{N} \,|\!)[\mathsf{S}'] \,,\, \mathsf{T}}$$

(MEM-A)
$$\frac{\mathsf{M} \xrightarrow{\mathrm{A}^{x}_{\mathtt{r}}} \mathsf{M}' \qquad \mathcal{A}(\mathrm{A}_{\mathtt{r}}, \mathrm{M}) = \mathrm{N} \quad \mathtt{diff}(\mathsf{M}, \mathsf{M}') \cap \mathtt{en}(\mathsf{S}) = \emptyset}{\mathrm{M}(\!|\, \mathsf{M} \,|\!)[\mathsf{S}] \xrightarrow{\mathrm{A}^{x}_{\mathtt{r}}} \mathrm{N}(\!|\, \mathsf{M}' \,|\!)[\mathsf{S}]}$$

(MS-AREACT)
$$\frac{\mathsf{M} \xrightarrow{\mathrm{A}^{x}_{\mathtt{r}}} \mathsf{M}' \quad \mathsf{S} \xrightarrow{\mathrm{B}^{x}_{\mathtt{r}}} \mathsf{S}' \qquad \mathrm{A} \neq \mathrm{B} \quad \mathcal{A}(\mathrm{A}_{\mathtt{r}}, \mathrm{M}) = \mathrm{N}}{\mathrm{M}(\!|\, \mathsf{M} \,|\!)[\mathsf{S}] \xrightarrow{\tau} \mathrm{N}(\!|\, \mathsf{M}' \,|\!)[\mathsf{S}']}$$

Rule (OPEN) prepares a cell to be fused with an enclosing cell or with a peer cell; the precondition guarantees that the cell may participate to a fusion. Rule (GRASP) lifts the fusion capability to groups by freezing them in a mreagent. Rules (FUSE-H) and (FUSE-V) define fusions between peer and nested cells, respectively. The new cell is created with the membrane name returned by the function $\mathcal{F}$. Rule (MEM-A) is a refinement of (MEM). It models possible side-effects on the membrane name due to interactions between membrane proteins and proteins outside the cell. Such interactions may activate membranes by changing their fusion capability, which is encoded in our formalism by membrane names. In a similar way, rule (MS-AREACT) refines (MS-REACT).

4.2 Viral infection

A virus is an intracellular parasite that uses the infected cell replication machinery in order to duplicate its own genetic material. Usually, a virus consists of a genetic material (DNA or RNA), a proteic structure called *capsid* providing protection for the genetic material – we use the term of nucleocapsid to denote both the capsid and the genetic material –, and a possible envelope (usually extracted from a former infected cell and used later on to infect other cells).

Below we encode in $\texttt{bio}\kappa$-calculus an influenza-like virus relying on similar descriptions in [6,11]. We focus on the infection part, as we cannot express any creation of new material. This part consists of the following steps:

(1) a protein-protein interaction between a virus membrane protein – the hemagglutinin HA – and a receptor – a glycoprotein GLY – on the cell's membrane; this activates GLY and prepares the cell to the phagocytosis;
(2) the phagocytosis of the virus occurs thus creating a new vesicle VES engulfing the virus;
(3) a fusion occurs between the new vesicle and an endosomal membrane (EDSM) in the cytoplasm; this makes the virus enter the endosome;
(4) a further fusion occurs between the endosome and the virus that is now part of the cytoplasm; this leads to an exocytosis that eventually releases the virus nucleocapsid into the cytoplasm.

We consider the following rules:

$$\mathtt{r}_3 \ : ((\text{EDSM}, \text{VES}), \text{EDSM}) \ \in \mathcal{F}$$
$$\mathtt{r}_4 \ : \ ((\text{EDSM}, \text{VS}), \text{EDSM}) \ \in \mathcal{F}$$

The initial solution is Virus , Cell where the components are as follows:

$$\begin{aligned}
\mathsf{Virus} &:= \text{VS}(\!| \, \text{HA}(1) \, |\!)[\mathsf{Nucaps}] \\
\mathsf{Cell} &:= \text{CLL}(\!| \, \text{GLY}(1) \ , \ \mathsf{M}_c \, |\!)[\mathsf{Endosome} \ , \ \mathsf{Cytosol}] \\
\mathsf{Endosome} &:= \text{EDSM}(\!| \, \mathsf{M}_e \, |\!)[\mathsf{E}_s]
\end{aligned}$$

We describe the last part of the infection pathway, assuming that the virus has already been engulfed in the host cell. We skip the first steps because we cannot express the phagocytosis of the virus. In Section 5 we will analyse the missing part. Therefore, let

$$\text{CLL}(\!| \, \mathsf{M}_c \, |\!)[\mathsf{Endosome} \ , \ \text{VES}(\!| \, \text{GLY}(1) \, |\!)[\mathsf{Virus}] \ , \ \mathsf{Cytosol}]$$

be the initial solution. A possible run is:

$$
\begin{array}{ll}
& \textsc{cll}(\!|\,\mathsf{M}_c\,|\!)[\textsc{edsm}(\!|\,\mathsf{M}_e\,|\!)[\mathsf{E}_s]\ ,\ \textsc{ves}(\!|\,\textsc{gly}(1)\,|\!)[\mathsf{Virus}]\ ,\ \mathsf{Cytosol}] \\
\xrightarrow{\tau} & \textsc{cll}(\!|\,\mathsf{M}_c\,|\!)[\textsc{edsm}(\!|\,\mathsf{M}_e\ ,\ \textsc{gly}(1)\,|\!)[\mathsf{Virus}\ ,\ \mathsf{E_s}]\ ,\ \mathsf{Cytosol}] & (\mathtt{r}_3) \\
\equiv & \textsc{cll}(\!|\,\mathsf{M}_c\,|\!)[\textsc{edsm}(\!|\,\mathsf{M}_e\ ,\ \textsc{gly}(1)\,|\!)[\textsc{vs}(\!|\,\textsc{ha}(1)\,|\!)[\mathsf{Nucaps}]\ ,\ \mathsf{E}_s]\ ,\ \mathsf{Cytosol}] \\
\xrightarrow{\tau} & \textsc{cll}(\!|\,\mathsf{M}_c\,|\!)[\textsc{edsm}(\!|\,\mathsf{M}_e\ ,\ \textsc{gly}(1)\ ,\ \textsc{ha}(1)\,|\!)[\mathsf{E}_s]\ ,\ \mathsf{Nucaps}\ ,\ \mathsf{Cytosol}] & (\mathtt{r}_4)
\end{array}
$$

4.3 An extensional semantics for cells

The extensional semantics of Definition 3 must be refined in order to account with new transitions and mreagents. This refinement should for instance equate solutions such as $\textsc{a}(1^x)\ ,\ \textsc{m}(\!|\,\textsc{b}(1^x)\,|\!)[\mathsf{S}]$ and $\textsc{a}(1)\ ,\ \textsc{n}(\!|\,\textsc{b}(1)\,|\!)[\mathsf{S}]$ when $\mathcal{C}(\mathtt{r}) = ((\textsc{a},1,\emptyset),(\textsc{b},1,\emptyset))$ $\mathcal{D}(\mathtt{r}') = ((\textsc{a},1,\emptyset),(\textsc{b},1,\emptyset))$ and $\mathcal{A}(\textsc{b}_{\mathtt{r}},\textsc{n}) = \textsc{m}'$ and $\mathcal{A}(\textsc{b}_{\mathtt{r}'},\textsc{m}') = \textsc{n}$. This case is very similar to that of reversible reactions discussed in the previous section.

The notations $\mathsf{S} \stackrel{\tau}{\Longrightarrow} \mathsf{T}$ and $\mathsf{S} \stackrel{\mu}{\Longrightarrow} \mathsf{T}$, $\mu \neq \tau$, are defined in the same way as in Definition 2.

Definition 6 *A* context bisimulation *is a symmetric binary relation* $\mathfrak{R}$ *between solutions such that it is a bisimulation and if* $\mathsf{S}\ \mathfrak{R}\ \mathsf{T}$ *and if* $\mathsf{S} \xrightarrow{\mathrm{M}_{\mathtt{r}}} (\!|\,\mathsf{M}\ ;\ \mathsf{S}''\,|\!) \cdot \mathsf{S}'$ *then* $\mathsf{T} \stackrel{\mathrm{M}_{\mathtt{r}}}{\Longrightarrow} (\!|\,\mathsf{M}'\ ;\ \mathsf{T}''\,|\!) \cdot \mathsf{T}'$ *and, for every* N, R, *and* $\textsc{n}$ *such that* $\mathcal{F}(\mathtt{r}) = (\textsc{m} \otimes \textsc{n}, \textsc{n}')$, *we have both:*

- $\big(\mathsf{S}''\ ,\ \textsc{n}'(\!|\,\mathsf{M}\ ,\ \mathsf{N}\,|\!)[\mathsf{S}']\big)\ \mathfrak{R}\ \big(\mathsf{T}''\ ,\ \textsc{n}'(\!|\,\mathsf{M}'\ ,\ \mathsf{N}\,|\!)[\mathsf{T}'']\big)$
- $\big(\mathsf{S}'\ ,\ \textsc{n}'(\!|\,\mathsf{M}\ ,\ \mathsf{N}\,|\!)[\mathsf{S}''\ ,\ \mathsf{R}]\big)\ \mathfrak{R}\ \big(\mathsf{T}'\ ,\ \textsc{n}'(\!|\,\mathsf{M}'\ ,\ \mathsf{N}\,|\!)[\mathsf{T}''\ ,\ \mathsf{R}]\big)$.

S *is context bisimilar to* T, *written* $\mathsf{S} \approx_c \mathsf{T}$, *if* $\mathsf{S}\ \mathfrak{R}\ \mathsf{T}$ *for some context bisimulation* $\mathfrak{R}$.

Context bisimulation retains the same substitutivity property of $\approx$ stated in Proposition 2. Besides, we have the following theorem:

Theorem 1 $\approx_c$ *is a congruence.*

Proof: *The proof is similar to that of Theorem 1. We only discuss the case of contexts* $[\cdot]\ ,\ \mathsf{R}$. *Let* $\mathfrak{R}$ *be a context bisimulation such that* $\mathsf{S}\ \mathfrak{R}\ \mathsf{T}$ *and let* $(\mathsf{S}'\ ,\ \mathsf{R})\ \mathfrak{R}'\ (\mathsf{T}'\ ,\ \mathsf{R})$ *if*

(1) $\mathsf{S}'\ \mathfrak{R}\ \mathsf{T}'$,

(2) and $\mathsf{S}'\ ,\ \mathsf{R}$ *and* $\mathsf{T}'\ ,\ \mathsf{R}$ *are well-formed.*

To demonstrate that $\mathfrak{R}'$ *is a bisimulation one has to prove that if* $\mathsf{S}', \mathsf{R} \xrightarrow{\mu} \mathsf{U}$ *then* $\mathsf{T}', \mathsf{R} \stackrel{\mu}{\Longrightarrow} \mathsf{U}'$ *and* $\mathsf{U}\,\mathfrak{R}'\,\mathsf{U}'$*. Among the possible cases we discuss the case when* $\mathsf{S}', \mathsf{R} \xrightarrow{\tau} \mathsf{U}$ *is due to a rule (*FUSE*) between an mreagent of* S' *and another of* R*.*

Let $\mathsf{S}' \xrightarrow{\mathrm{M}_{\mathtt{r}}} \{\!|\mathsf{M}\,;\mathsf{S}'''|\!\} \cdot \mathsf{S}''$*,* $\mathsf{R} \xrightarrow{\mathrm{N}_{\mathtt{r}}} \{\!|\mathsf{N}\,;\mathsf{R}''|\!\} \cdot \mathsf{R}'$*, and* $\mathcal{F}(\mathtt{r}) = (\mathrm{M} \otimes \mathrm{N}, \mathrm{M}')$*. Then* $\mathsf{U} = \mathsf{S}'', \mathrm{M}'(\!|\mathsf{M}, \mathsf{N}|\!)[\mathsf{S}''', \mathsf{R}''], \mathsf{R}'$*. Since* $\mathsf{S}'\,\mathfrak{R}\,\mathsf{T}'$ *then* $\mathsf{T}' \stackrel{\mathrm{M}_{\mathtt{r}}}{\Longrightarrow} \{\!|\mathsf{M}'\,;\mathsf{T}'''|\!\} \cdot \mathsf{T}''$ *and* $\mathsf{U}' = \mathsf{T}'', \mathrm{M}'(\!|\mathsf{M}', \mathsf{N}|\!)[\mathsf{T}''', \mathsf{R}''], \mathsf{R}'$*. By definition of* $\mathfrak{R}$*:*

$$(\mathsf{S}'', \mathrm{M}'(\!|\mathsf{M}, \mathsf{N}|\!)[\mathsf{S}''', \mathsf{R}'']) \,\mathfrak{R}\, (\mathsf{T}'', \mathrm{M}'(\!|\mathsf{M}', \mathsf{N}|\!)[\mathsf{T}''', \mathsf{R}''])$$

Therefore $\mathsf{U}\,\mathfrak{R}'\,\mathsf{U}'$ *follows because of the* $\mathfrak{R}'$*-closure with respect to contexts* $[\cdot], \mathsf{R}'$*.*

Context bisimilarity retains a universal quantification that is hard to check in practice. One might wonder whether it is possible to simplify the definition. For example, instead of quantifying on cells, one may simply require the bisimilarity of components of mreactants, namely $\mathsf{M} \approx_c \mathsf{M}'$, $\mathsf{S}'' \approx_c \mathsf{T}''$, and $\mathsf{S}' \approx_c \mathsf{T}'$. It is easy to demonstrate that the induced equivalence, which we note $\approx_c^+$, is a congruence and $\approx_c^+ \subseteq \approx_c$. At the time we write this note it is not clear to us whether this containment is strict or not. This issue actually requires further investigations.

5 Translocation and phagocytosis

The bioκ-calculus presented in Sections 2 and 4 has a limited expressive power: mechanisms such as translocation, where a single protein may enter a cell, or phagocytosis, where a cell may enter another cell cannot be described. For this reason we overlooked the first steps of the virus infection in example of Section 4.2. The integration of translocation and phagocytosis in bioκ-calculus is not simple and admits several design choices. We discuss few possible formalisations below.

5.1 Translocation

Translocation is a mechanism enabling the transport of proteins through a membrane. This mechanism is very specific and controlled by particular membrane proteins that are different for each type of membrane.

Translocations usually do not transport the full proteins in one step because they are too big for traversing the membrane. This problem is solved in two ways. One way is that the mRNA containing the genetic code of the protein interacts with a ribosome – big proteic complex in charge of translating the

code. This interaction translates part of the code in the cell and, *at the same time,* creates the encoded protein. Alternatively, ad-hoc proteins, called the *chaperons*, are used to unfold the entering protein during the process (this is actually what happen at the end of the RTK-MAPK cascade described in Section 2).

We abstract from this low level mechanisms and assume that proteins may safely traverse the membrane. A first approximative definition of translocation might only check that the entering protein be disconnected and retains a suitable interface:

$$\mathrm{A}(\phi+\psi)\ ,\ \mathrm{M}(\!|\ \mathsf{M}\ |\!)[\mathsf{S}] \xrightarrow{\tau} \mathrm{M}(\!|\ \mathsf{M}\ |\!)[\mathrm{A}(\overline{\phi}+\psi)\ ,\ \mathsf{S}']$$

According to our notation, both ϕ and ψ are v-h-maps, therefore $\mathtt{en}(\phi+\psi)=\emptyset$. This description is not satisfactory for it makes the membrane play a passive role while this process is on the contrary highly specific. As such, it is impossible to represent the effects provided by the chaperon proteins.

A better way is to model translocation as a decomplexation rule between an external protein already connected to the membrane (and nowhere else) and a membrane protein. To avoid the confusion of two different phenomena – simple decomplexations and decomplexations with translocations – we use a further function $\mathcal{T}$ from rule names to tuples $((\mathrm{A}, i, \phi, \phi'), (\mathrm{B}, j, \psi, \psi'), \mathrm{M}, \mathrm{N})$. As usual we assume that there is no clash between rule names in the domain of $\mathcal{T}$ and the other functions that have been used in the paper. The two rules controlling translocations are:

$$\text{(TRS-P)}\quad \frac{\mathcal{T}(\mathtt{r}) = ((\mathrm{A}, \mathtt{i}, \phi, \phi'), (\mathrm{B}, \mathtt{j}, \psi, \psi'), \mathrm{M}, \mathrm{N})}{\mathrm{A}(i^x+\phi+\phi') \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathbf{0}}$$

$$\text{(TRS-M)}\quad \frac{\mathsf{M} \xrightarrow{\mathrm{B}^x_{\mathtt{r}}} \mathsf{M}' \qquad \mathcal{T}(\mathtt{r}) = ((\mathrm{A}, \mathtt{i}, \phi, \phi'), (\mathrm{B}, \mathtt{j}, \psi, \psi'), \mathrm{M}, \mathrm{N})}{\mathrm{M}(\!|\ \mathsf{M}\ |\!)[\mathsf{S}] \xrightarrow{\mathrm{B}^x_{\mathtt{r}}} \mathrm{N}(\!|\ \mathsf{M}'\ |\!)[\mathrm{A}(i+\overline{\phi}+\phi')\ ,\ \mathsf{S}']}$$

Fig. 4. Reduction rules for translocation

In rule (TRS-P) the interface of A has exactly one site bound because the interfaces ψ and ψ' are v-h-maps according our notation. The interface ψ is being turned into $\overline{\psi}$ during the translocation, ψ' is unchanged. The protein A disappears during (TRS-P). Dually, the protein A appears during (TRS-M). The translocation will be the effect of the synchronisation (REACT).

5.2 Phagocytosis.

As stated in the introduction, endocytosis is a mechanism allowing a cell to engulf new material. There exists of course many different types of endocytosis. We focus in particular on the *active endocytosis* which is a phenomenon arising as a response to an activation or, more generally, to an external signal. In order to distinguish it from the general endocytosis and since we intend to use this phenomenon to engulf complex structures (such as a virus), we use the term of *phagocytosis*, after the name of the associated brane primitive. The phagocytosis of $\mathrm{M}(\!|\,\mathsf{M}\,|\!)[\mathsf{S}]$ – usually a small cell – by $\mathrm{N}(\!|\,\mathsf{N}\,|\!)[\mathsf{T}]$ – usually a big cell – transports the former into the cytoplasm of the latter *by surrounding* $\mathrm{M}(\!|\,\mathsf{M}\,|\!)[\mathsf{S}]$ *with a new membrane that is part of* N. This surrounding mechanism is the problematic one because it amounts to split the host cell membrane in some "not local" way. For example the transition

$$\mathrm{M}(\!|\,\mathsf{M}\,|\!)[\mathsf{S}]\ ,\mathrm{N}(\!|\,\mathsf{N}\,|\!)[\mathsf{T}] \xrightarrow{\tau} \mathrm{N}(\!|\,\mathsf{N}\,|\!)[\mathrm{N}'(\!|\,\mathsf{0}\,|\!)[\mathrm{M}(\!|\,\mathsf{M}\,|\!)[\mathsf{S}]]\ ,\mathsf{T}]$$

is not very appropriate because the new membrane N' is empty. As we don't have any mechanism for feeding the membrane yet, the solution ban complexations of the new membrane with proteins.

Actually, phagocytosis should be possible provided the membrane of the host cell had enough material for a new membrane. We therefore model phagocytosis as a decomplexation of two proteins in the membranes of the reactant cells with the side effect of splitting the host cell according to some predefined pattern. As for translocations, we use a new functions from rule names to tuples $((\mathrm{A}, i, \phi, \phi'), (\mathrm{B}, j, \psi, \psi'), \mathrm{M}, \mathrm{N}, \mathrm{N}', \mathrm{N}'', \mathsf{N}')$, where $\mathtt{de}(\mathsf{N}') = \emptyset$. The meaning of this tuple is the following: $(\mathrm{A}, i, \phi, \phi')$ and $(\mathrm{B}, j, \psi, \psi')$ are the two proteins that decomplexate and are located in two membranes M and N, respectively. The name N' is the one given at the new membrane surrounding the phagocytosed cell, N' is the membrane material of the new cell.

In the following rules, transition labels are extended with $\mathrm{M}^x_{\mathtt{r}}$ and still ranged over by μ. Let $\uplus$ denote disjoint union of sets. The rules defining phagocytosis are the one of Figure 5.

Rule (OPEN-P) defines the transition of the phogocytosed cell. The label $\mathrm{A}^x_{\mathtt{r}}$ of the membrane transition becomes $\mathrm{M}^x_{\mathtt{r}}$ in the cellular transition. This exposes the phagocytosis to the label. Similarly for the rule (OPEN-C). In (OPEN-C) the material needed for the new membrane surrounding the phagocytosed cell is removed from the host cell membrane. This material is restored in the rule (PHAGO).

It is not clear to us how close the above rules are to the biological phagocytosis. It is worth to observe that (OPEN-C) is computationally expensive, at least if

$$\text{(OPEN-P)}\quad \frac{\mathsf{M} \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathsf{M}' \quad \mathcal{P}(\mathtt{r}) = ((\mathrm{A},\mathtt{i},\phi,\phi'),(\mathrm{B},\mathtt{j},\psi,\psi'),\mathrm{M},\mathrm{N},\mathrm{N}',\mathrm{N}'',\mathsf{N}')}{{}_{\mathrm{M}}\langle\!|\,\mathsf{M}\,|\!\rangle[\mathsf{S}] \xrightarrow{\mathrm{M}^x_{\mathtt{r}}} \wr\mathsf{M}'\,;\,\mathsf{S}\int\cdot\mathbf{0}}$$

$$\text{(OPEN-C)}\quad \frac{\mathsf{M} \xrightarrow{\mathrm{B}^x_{\mathtt{r}}} \prod_{k\in I\uplus J}\mathrm{B}_k(\sigma_k) \quad \mathcal{P}(\mathtt{r}) = ((\mathrm{A},\mathtt{i},\phi,\phi'),(\mathrm{B},\mathtt{j},\psi,\psi'),\mathrm{M},\mathrm{N},\mathrm{N}',\mathrm{N}'',\prod_{\mathtt{j}\in\mathrm{J}}\mathrm{B}_j(\sigma_{\mathtt{j}}))}{{}_{\mathrm{N}}\langle\!|\,\mathsf{M}\,|\!\rangle[\mathsf{S}] \xrightarrow{\mathrm{N}^x_{\mathtt{r}}} \wr\prod_{i\in I}\mathrm{B}_i(\sigma_i)\,;\,\mathsf{S}\int\cdot\mathbf{0}}$$

$$\text{(PHAGO)}\quad \frac{\begin{array}{c}\mathsf{S} \xrightarrow{\mathrm{M}^x_{\mathtt{r}}} \wr\mathsf{M}\,;\,\mathsf{S}''\int\cdot\mathsf{S}' \qquad \mathsf{T} \xrightarrow{\mathrm{N}^x_{\mathtt{r}}} \wr\mathsf{N}\,;\,\mathsf{T}''\int\cdot\mathsf{T}' \\ \mathcal{P}(\mathtt{r}) = ((\mathrm{A},\mathtt{i},\phi,\phi'),(\mathrm{B},\mathtt{j},\psi,\psi'),\mathrm{M},\mathrm{N},\mathrm{N}',\mathrm{N}'',\mathsf{N}')\end{array}}{\mathsf{S}\,,\,\mathsf{T} \xrightarrow{\tau} \mathsf{S}'\,,\,\mathsf{T}'\,,\,{}_{\mathrm{N}'}\langle\!|\,\mathsf{N}\,|\!\rangle[{}_{\mathrm{N}''}\langle\!|\,\mathsf{N}'\,|\!\rangle[{}_{\mathrm{M}}\langle\!|\,\mathsf{M}\,|\!\rangle[\mathsf{S}'']]\,,\,\mathsf{T}'']}$$

Fig. 5. Reduction rules for phagocytosis

compared to the other operations described in the paper. According to (OPEN-C), extracting a pattern of proteins out of a membrane amounts to a long sequel of checks that lock the membrane, thus inhibiting other interactions. It is an open question whether it is possible or not to design simpler and more basic mechanisms for phagocytosis.

Then we could finish the modelling of the example of Section 4.2. We present Figure 6 the complete set of rules used to make evolve the solution $\mathsf{Virus}\,,\,\mathsf{Cell}$ to the state in which the nucleocapsid is released into the cytoplasms of the infected cell.

$$\begin{aligned}
\mathtt{r}_1 &: && ((\mathrm{HA},1,\emptyset),(\mathrm{GLY},1,\emptyset)) && \in \mathcal{C}\\
\mathtt{r}'_1 &: && (\mathrm{GLY}_{\mathtt{r}_1},\mathrm{CLL}) \mapsto \mathrm{ACLL} && \in \mathcal{A}\\
\mathtt{r}_2 &: ((\mathrm{HA},1,\emptyset),(\mathrm{GLY},1,\emptyset)),\mathrm{VS},\mathrm{ACLL},\mathrm{CLL},\mathrm{VES},(\mathrm{GLY}(1)) && && \in \mathcal{P}\\
\mathtt{r}_3 &: && ((\mathrm{EDSM},\mathrm{VES}),\mathrm{EDSM}) && \in \mathcal{F}\\
\mathtt{r}_4 &: && ((\mathrm{EDSM},\mathrm{VS}),\mathrm{EDSM}) && \in \mathcal{F}
\end{aligned}$$

Fig. 6. Set of interactions rules modelling the virus infection

The formal rendering of the example of Section 4.2 can now be stated from the beginning, namely before that the phagocytosis occurs. This is what is presented in Figure 7.

$$
\begin{aligned}
& \mathrm{VS}(\!|\,\mathrm{HA}(1)\,|\!)[\mathsf{Nucaps}]\ ,\ \mathrm{CLL}(\!|\,\mathrm{GLY}(1)\ ,\ \mathsf{M_c}\,|\!)[\mathsf{Endosome}\ ,\ \mathsf{Cytosol}] \\
\xrightarrow{\tau}\ & \mathrm{VS}(\!|\,\mathrm{HA}(1^x)\,|\!)[\mathsf{Nucaps}]\ ,\ \mathrm{ACLL}(\!|\,\mathrm{GLY}(1^{\mathsf{x}})\ ,\ \mathsf{M_c}\,|\!)[\mathsf{Endosome}\ ,\ \mathsf{Cytosol}] && (\mathtt{r}_1 - \mathtt{r}_1') \\
\xrightarrow{\tau}\ & \mathrm{CLL}(\!|\,\mathsf{M}_c\,|\!)[\mathrm{VES}(\!|\,\mathrm{GLY}(1)\,|\!)[\mathrm{VS}(\!|\,\mathrm{HA}(1)\,|\!)[\mathsf{Nucaps}]]\ ,\ \mathsf{Endosome}\ ,\ \mathsf{Cytosol}] && (\mathtt{r}_2) \\
\xrightarrow{\tau}\ & \mathrm{CLL}(\!|\,\mathsf{M}_c\,|\!)[\mathrm{EDSM}(\!|\,\mathsf{M}_e\ ,\ \mathrm{GLY}(1)\,|\!)[\mathsf{Virus}\ ,\ \mathsf{E_s}]\ ,\ \mathsf{Cytosol}] && (\mathtt{r}_3) \\
\equiv\ & \mathrm{CLL}(\!|\,\mathsf{M}_c\,|\!)[\mathrm{EDSM}(\!|\,\mathsf{M}_e\ ,\ \mathrm{GLY}(1)\,|\!)[\mathrm{VS}(\!|\,\mathrm{HA}(1)\,|\!)[\mathsf{Nucaps}]\ ,\ \mathsf{E}_s]\ ,\ \mathsf{Cytosol}] \\
\xrightarrow{\tau}\ & \mathrm{CLL}(\!|\,\mathsf{M}_c\,|\!)[\mathrm{EDSM}(\!|\,\mathsf{M}_e\ ,\ \mathrm{GLY}(1)\ ,\ \mathrm{HA}(1)\,|\!)[\mathsf{E}_s]\ ,\ \mathsf{Nucaps}\ ,\ \mathsf{Cytosol}] && (\mathtt{r}_4)
\end{aligned}
$$

Fig. 7. Run of the first steps of the virus infection

6 Implementing the core bioκ-calculus

In this section we discuss an implementation of core bioκ-calculus in π-calculus [12]. The encoding below is simpler than the one in [3], because rules in core bioκ-calculus have always two interacting proteins. In particular, the implementation below do not change the granularity of reactions: every biological reaction is translated into *single* π-calculus transitions.

We begin with a brief introduction to π-calculus, and then detail the compilation of bioκ-calculus.

6.1 The π-calculus

The π-calculus uses a countable set of *names* $\mathcal{N}$, ranged over by x, y, z, ..., and *agent names*, ranged over by $\mathtt{A}$, $\mathtt{B}$, ... Tuples of names are noted $\widetilde{x}$. Processes P are defined by the grammar:

$$
\begin{aligned}
P := \quad & \mathbf{0} \ \mid\ \overline{x}\,\widetilde{z}.P \ \mid\ x\,(\widetilde{z}).P \ \mid\ (x)P \ \mid\ P+P \ \mid\ P\,|\,P \\
& \mid\ [x=y]\,P \ \mid\ \mathtt{A}(\widetilde{x})
\end{aligned}
$$

where process names are defined by a set of equations $\mathtt{A}(\widetilde{z}) := P$.

A process can be the inert process $\mathbf{0}$, an output $\overline{x}\,\widetilde{z}.P$ that produces a message $\overline{x}\,\widetilde{z}$ and behaves like P; an input $x\,(\widetilde{z}).P$ that consumes a message $\overline{x}\,\widetilde{u}$ and behaves like $P\{\widetilde{u}/\widetilde{z}\}$; a restriction $(x)P$ that behaves as P except that messages on x are prohibited; a choice $P+Q$ that behaves either as P or as Q; a parallel composition $P\,|\,Q$ of two processes; a match $[x=y]P$ that behaves as P provided x and y are equal; an agent invocation $\mathtt{A}(\widetilde{x})$. The input $x\,(\widetilde{z}).P$, restriction $(x)P$, and agent definition $\mathtt{A}(\widetilde{z}) := P$ are binders of names $\widetilde{z}$, x, and $\widetilde{z}$, respectively. The scope of these binders are the processes P. We use the

(INP) $x\,(\widetilde{z}).P \xrightarrow{x\,(\widetilde{z})} P$

(OUT) $\overline{x}\,\widetilde{z}.P \xrightarrow{\overline{x}\widetilde{z}} P$

(NEW) $$\frac{P \xrightarrow{\mu} Q \quad x \notin \mathtt{fn}(\mu)}{(x)P \xrightarrow{\mu} (x)Q}$$

(OPEN) $$\frac{P \xrightarrow{(\widetilde{y})\overline{x}\widetilde{z}} Q \quad z' \neq x \quad z' \in \widetilde{z} \setminus \widetilde{y}}{(z')P \xrightarrow{(z'\widetilde{y})\overline{x}\widetilde{z}} Q}$$

(SUM) $$\frac{P \xrightarrow{\mu} P'}{P + Q \xrightarrow{\mu} P'}$$

(PAR) $$\frac{P \xrightarrow{\mu} P' \quad \mathtt{bn}(\mu) \cap \mathtt{fn}(Q) = \emptyset}{P|Q \xrightarrow{\mu} P'|Q}$$

(MATCH) $$\frac{P \xrightarrow{\mu} Q}{[u = u]\,P \xrightarrow{\mu} Q}$$

(APP) $$\frac{\mathtt{A}(\widetilde{x}) := P \quad P\{\widetilde{z}/\widetilde{x}\} \xrightarrow{\mu} Q}{\mathtt{A}(\widetilde{z}) \xrightarrow{\mu} Q}$$

(COM) $$\frac{P \xrightarrow{(\widetilde{y})\overline{x}\widetilde{z}} P' \quad Q \xrightarrow{x\,(\widetilde{w})} Q' \quad \widetilde{y} \cap \mathtt{fn}(Q) = \emptyset}{P|Q \xrightarrow{\tau} (\widetilde{y})(P'|Q'\{\widetilde{z}/\widetilde{w}\})}$$

Fig. 8. Operational semantics of the π-calculus.

standard notion of α-equivalence, *free* and *bound names* of processes, noted $\mathtt{fn}(P)$ and $\mathtt{bn}(P)$, respectively. In the following we let $\sum_{i\in I} P_i$ be the choice between the processes P_i. We also let $\prod_{i\in I}[x_i = y_i]$ be the sequence of matches $[x_i = y_i]$.

Figure 8 collects the semantics of π-calculus, except for the symmetric forms of rules (SUM) and (PAR) that are omitted. The semantics is described as a transition system on syntactic processes with transitions labelled by *actions*. These actions, written μ, are of three types: *internal* actions τ, *inputs* $x\,(\widetilde{u})$, and *outputs* $(\widetilde{y})\overline{x}\,\widetilde{u}$. Outputs $(\widetilde{y})\overline{x}\,\widetilde{u}$ are *bounded* when $\widetilde{y}$ is not empty. Bounded outputs, $(\widetilde{y})\overline{x}\,\widetilde{u}$, generated by the transitions above all satisfy: 1) $\widetilde{y} \subseteq \widetilde{u}$ and 2) $x \notin \widetilde{y}$.

One defines:

$$\mathtt{fn}(\tau) = \emptyset \qquad \mathtt{bn}(\tau) = \emptyset$$
$$\mathtt{fn}(x\,(\widetilde{u})) = \{x\} \qquad \mathtt{bn}(x\,(\widetilde{u})) = \widetilde{u}$$
$$\mathtt{fn}((\widetilde{y})\overline{x}\,\widetilde{u}) = \{x\} \cup (\widetilde{u} \setminus \widetilde{y}) \qquad \mathtt{bn}((\widetilde{y})\overline{x}\,\widetilde{u}) = \widetilde{y}$$

Rules (PAR), (COM), (NEW) and (OPEN) have side-conditions controlling bounded output and involving $\mathtt{fn}(\mu)$ and $\mathtt{bn}(\mu)$. Specifically, these conditions ensure that 1) the exported bounded names do not capture any variables when they

finally appear in the right hand side of the conclusion of rule (COM), 2) it is not possible to input or output on a restricted name.

6.2 The compilation

We distinguish three names, v, h, and b, in the set $\mathcal{N}$. These names are considered constants. Our compilation is "protein-centric", that is to say proteins are translated as processes whose behaviour is obtained from all the interactions they participate to. In particular, the behaviour of a protein A is specified by a (unique) process definition

$$\mathtt{A}(\widetilde{u};\widetilde{x};\widetilde{r}) := P \ .$$

where the arguments of $\mathtt{A}$ have been partitioned using semicolons because they play different roles. The tuples $\widetilde{u}$ and $\widetilde{x}$ have length $\mathfrak{s}(\mathrm{A})$. Names $\widetilde{u}$ are always instantiated by v, h, or b, according to the corresponding site is visible, hidden, or bound, respectively. The tuple $\widetilde{x}$ is instantiated by names representing bindings between proteins. These names are meaningful provided the corresponding name in the tuple $\widetilde{u}$ is b. When names are meaningless, they are set to h. The tuple $\widetilde{r}$ has length twice the number of complexation rules where A is a reactant ($\widetilde{r}$ may be seen as a sequence of pairs). Let $\mathtt{r}$ be a complexation with a reactant A and let r_i, r_e be the pair in $\widetilde{r}$ corresponding to such a rule. The two names r_i and r_e are used for interacting with proteins outside the cell and with proteins in the cytoplasm by means of $\mathtt{r}$. This distinction is significant for proteins in membranes; the two names are identical for the other proteins. In the following we assume that complexation rules are enumerated. Therefore, there is a first complexation rule, a second one, etc.

The $\mathtt{A}(\widetilde{u};\widetilde{x};\widetilde{r}) := P$ requires the following notational conventions:

$\mathsf{prj}_i(\widetilde{x})$ is the i-th element of $\widetilde{x}$.
$\sharp_c(\mathtt{r})$ gives the ordinal of the complexation rule $\mathtt{r}$. It is undefined if $\mathtt{r}$ is a decomplexation.
$\sharp_{d,\mathrm{A}}(\mathtt{r})$ gives the bound site of the protein A in the decomplexation rule $\mathtt{r}$. It is undefined if $\mathtt{r}$ is a complexation or A is not a reactant of $\mathtt{r}$.
$\mathrm{TEST}(\widetilde{u},\phi)$ gives the sequence of matches $\prod_{j\in\mathrm{dom}(\phi)}[\mathsf{prj}_j(\widetilde{u}) = \phi(j)]$.

$\textsc{set}(\widetilde{u}, i, \phi)$, where $i \notin \mathrm{dom}(\phi)$, is defined elementwise as follows:

$$\mathsf{prj}_j(\textsc{set}(\widetilde{u}, i, \phi)) := \begin{cases} \mathsf{prj}_j(\widetilde{u}) & \text{if } j \notin \{i\} \cup \mathrm{dom}(\phi) \\ b & \text{if } i = j \text{ and } \mathsf{prj}_i(\widetilde{u}) = v \\ v & \text{if } i = j \text{ and } \mathsf{prj}_i(\widetilde{u}) = b \\ h & \text{if } \phi(j) = v \\ v & \text{if } \phi(j) = h \end{cases}$$

$\textsc{set}_x(\widetilde{u}, i)$ is defined elementwise as follows:

$$\mathsf{prj}_j(\textsc{set}_x(\widetilde{u}, i)) := \begin{cases} x & \text{if } i = j \\ \mathsf{prj}_j(\widetilde{u}) & \text{otherwise} \end{cases}$$

Every preliminary notion is set for the definition of $\mathtt{A}(\widetilde{u}; \widetilde{x}; \widetilde{r})$.

$$\begin{aligned} &\mathtt{A}(\widetilde{u}; \widetilde{x}; \widetilde{r}) := \\ &\quad \textstyle\sum_{\mathrm{A}(i+\phi+\phi') \in_{\mathrm{left}} \mathcal{C}(\mathtt{r})} \quad \textsc{test}(\widetilde{u}, i + \phi + \phi')\,(z)\Big(\\ &\qquad\qquad \overline{\mathsf{prj}_{2\sharp_c(\mathtt{r})}(\widetilde{r})}\, z.\ \mathtt{A}(\textsc{set}(\widetilde{u}, i, \phi); \textsc{set}_z(\widetilde{x}, i); \widetilde{r}) \\ &\qquad\qquad + \quad \overline{\mathsf{prj}_{2\sharp_c(\mathtt{r})+1}(\widetilde{r})}\, z.\ \mathtt{A}(\textsc{set}(\widetilde{u}, i, \phi); \textsc{set}_z(\widetilde{x}, i); \widetilde{r})\Big) \\ &\quad + \textstyle\sum_{\mathrm{A}(i+\phi+\phi') \in_{\mathrm{right}} \mathcal{C}(\mathtt{r})} \quad \textsc{test}(\widetilde{u}, i + \phi + \phi')\Big(\\ &\qquad\qquad \mathsf{prj}_{2\sharp_c(\mathtt{r})}(\widetilde{r})\,(z).\ \mathtt{A}(\textsc{set}(\widetilde{u}, i, \phi); \textsc{set}_z(\widetilde{x}, i); \widetilde{r}) \\ &\qquad\qquad + \quad \mathsf{prj}_{2\sharp_c(\mathtt{r})+1}(\widetilde{r})\,(z).\ \mathtt{A}(\textsc{set}(\widetilde{u}, i, \phi); \textsc{set}_z(\widetilde{x}, i); \widetilde{r})\Big) \\ &\quad + \textstyle\sum_{\mathrm{A}(i+\phi+\phi') \in_{\mathrm{left}} \mathcal{D}(\mathtt{r})} \quad [\mathsf{prj}_i(\widetilde{u}) = b]\textsc{test}(\widetilde{u}, \phi + \phi') \\ &\qquad\qquad \overline{\mathsf{prj}_{\sharp_{d,\mathrm{A}}(\mathtt{r})}(\widetilde{x})}\ .\ \mathtt{A}(\textsc{set}(\widetilde{u}, i, \phi); \textsc{set}_h(\widetilde{x}, i); \widetilde{r}) \\ &\quad + \textstyle\sum_{\mathrm{A}(i+\phi+\phi') \in_{\mathrm{right}} \mathcal{D}(\mathtt{r})} \quad [\mathsf{prj}_i(\widetilde{u}) = b]\textsc{test}(\widetilde{u}, \phi + \phi') \\ &\qquad\qquad \mathsf{prj}_{\sharp_{d,\mathrm{A}}(\mathtt{r})}(\widetilde{x})\,(\,).\ \mathtt{A}(\textsc{set}(\widetilde{u}, i, \phi); \textsc{set}_h(\widetilde{x}, i); \widetilde{r}) \end{aligned}$$

Every branch of the choices in $A(\widetilde{u}; \widetilde{x}; \widetilde{r})$ verifies whether the interface fits with a left-hand side of some rule having $\mathtt{A}$ as a reactant or not. The test is implemented by the operation $\textsc{test}(\widetilde{u}, \phi)$, following our encoding of sites in the arguments $\widetilde{u}$ of $\mathtt{A}$. We notice that, the branches of $A(\widetilde{u}; \widetilde{x}; \widetilde{r})$ corresponding to decomplexation rules have an additional test $[\mathsf{prj}_i(\widetilde{u}) = b]$ verifying that the i-th site is actually bound.

Once a branch has been chosen, a biological reaction is compiled into a single π-calculus interaction, where the left reactant plays the role of sender and the

right reactant plays the role of receiver. The sender of a complexation has to create a new name representing the biological edge created by the rule. This name is communicated during the interaction to the other reactant and will be used to decomplexate the two proteins. The new state of the protein A is obtained by updating the sites as prescribed by the rule – see the definition of SET.

Few other notational conventions are useful for the definition of the compilation. Let $[\![\sigma]\!]_{\mathrm{A}}$ be function yielding a tuple of length $2 \times \mathfrak{s}(\mathrm{A})$ defined elementwise as follows:

$$\mathsf{prj}_i([\![\sigma]\!]_{\mathrm{A}}) = \begin{cases} v & \text{if } \sigma(i) = v \\ h & \text{if } \sigma(i) = h \\ b & \text{otherwise} \end{cases}$$

$$\mathsf{prj}_{2i}([\![\sigma]\!]_{\mathrm{A}}) = \begin{cases} x & \text{if } \sigma(i) = x \text{ and } x \notin \{v, h\} \\ h & \text{otherwise} \end{cases}$$

For example $\left[\!\left[1^x + 2 + \overline{3}\right]\!\right]_{\mathrm{A}} = (b, v, h; x, h, h)$. We notice that $[\![\sigma]\!]_{\mathrm{A}}$ is undefined if σ is not total on $1..\mathfrak{s}(\mathrm{A})$. Let also $(x_1, y_1) \cdots (x_n, y_n)\,|_{\mathrm{A}}$ be the subsequence containing (x_i, y_i) provided A is a reactant of the i-th complexation rule. Finally, let n be the number of complexation rules of the biological system. The compilation of a solution S is given by $[\![\mathsf{S}]\!]_{(r_1,r_1),\cdots,(r_n,r_n)}$, where $r_1, \cdots, r_n$ are pairwise different names:

$$\begin{aligned} [\![\mathbf{0}]\!]_\ell &:= \mathbf{0} \\ [\![\mathrm{A}(\sigma)]\!]_\ell &:= \mathtt{A}([\![\sigma]\!]_{\mathrm{A}}\,; \ell|_{\mathrm{A}}) \\ [\![\mathsf{S}\ ,\ \mathsf{T}]\!]_\ell &:= [\![\mathsf{S}]\!]_\ell \mid [\![\mathsf{T}]\!]_\ell \\ \left[\!\left[\mathrm{M}(\!|\,\mathsf{M}\,|\!)[\mathsf{S}]\right]\!\right]_{(r_1,r_1')\cdots(r_n,r_n')} &:= (r_1'') \cdots (r_n'')\big(\ [\![\mathsf{M}]\!]_{(r_1',r_1'')\cdots(r_n',r_n'')} \mid [\![\mathsf{S}]\!]_{(r_1'',r_1'')\cdots(r_n'',r_n'')}\big) \end{aligned}$$

The compilation schemas are simple except that of $\mathrm{M}(\!|\,\mathsf{M}\,|\!)[\mathsf{S}]$, which is discussed below. As regards cells, one has to ban reactions between proteins in the cytoplasm and those that are external to the cell. This may be enforced in π-calculus by picking a fresh set of names for complexation rules. For this reason the compilation creates a fresh set of complexation names $r_1'', \cdots, r_n''$ and allows them to occur in proteins of S and M. On the contrary, proteins in the membrane M may interact both with the external environment and with the cytoplasm. The compilation supports this feature by encoding proteins in M with the set of complexation names of the external environment and the fresh set of complexation names of the cytoplasm. It is worth to notice that

decomplexations are not an issue because their correctness follows from the well-formedness of the solution.

The following proposition states the correctness of the compilation. The proof is a standard induction on the structure of S.

Proposition 2 *Let $\mathcal{C} = \{\mathtt{r}_1, \cdots, \mathtt{r}_n\}$, and let $r_1, \cdots, r_n$ and $r'_1, \cdots, r'_n$ be two tuples of pairwise different names. Let $\ell = (r_1, r'_1) \cdots (r_n, r'_n)$. The following reductions are in core* bioκ*-calculus.*

(1) $\mathsf{S} \xrightarrow{\mathrm{A}^x_{\mathtt{r}_k}} \mathsf{T}$*, where* A *is the left reactant (resp. right reactant), if and only if either* $[\![\mathsf{S}]\!]_\ell \xrightarrow{(z)\overline{r_k}\,z} [\![\mathsf{T}]\!]_\ell$ *or* $[\![\mathsf{S}]\!]_\ell \xrightarrow{(z)\overline{r'_k}\,z} [\![\mathsf{T}]\!]_\ell$ *(resp. either* $[\![\mathsf{S}]\!]_\ell \xrightarrow{r_k\,(z)} [\![\mathsf{T}]\!]_\ell$ *or* $[\![\mathsf{S}]\!]_\ell \xrightarrow{r'_k\,(z)} [\![\mathsf{T}]\!]_\ell$*);*

(2) $\mathsf{S} \xrightarrow{\mathrm{A}^x_{\mathtt{r}}} \mathsf{T}$*, where* r *is a decomplexation rule and* A *is the left reactant (resp. the right reactant), if and only if* $[\![\mathsf{S}]\!]_\ell \xrightarrow{\overline{z}} [\![\mathsf{T}]\!]_\ell$ *(resp.* $[\![\mathsf{S}]\!]_\ell \xrightarrow{z\,()} [\![\mathsf{T}]\!]_\ell$*);*

(3) $\mathsf{S} \xrightarrow{\tau} \mathsf{T}$ *if and only if* $[\![\mathsf{S}]\!]_\ell \xrightarrow{\tau} [\![\mathsf{T}]\!]_\ell$*.*

We conclude by commenting on the extension of the encoding to the full bioκ-calculus. The problematic rules of bioκ-calculus are fusions because they allow interactions between proteins that were banned. For instance, the fusion between two cells in a same solution allows interactions between the proteins in the two cytoplasms. So, a local interaction has global effects in a given environment. This situation is usually hard to encode into π-calculus. A way out is to use formalisms permitting simple modellings of such global effects, such as the *fusion calculus* [13]. In this calculus, a communication equates names, thus causing communications on names that were not possible before. In the foregoing compilation, a fusion should equate tuples of names $r_1, \cdots, r_n$, thus allowing complexations between proteins using such names. We are currently investigating the soundness of this idea.

7 Conclusions

We have presented a unique framework for modelling proteins and cells interactions – the bioκ-calculus. Protein interactions in bioκ-calculus are of two types: complexations and decomplexations; cell interactions in bioκ-calculus describe fusions. All interactions are "local" in the sense that they always involve two proteins. Fusions have been modelled by using an higher order semantics in the style of [10]. We have studied the operational semantics of bioκ-calculus and an extensional semantics of its – the bisimulation. The expressiveness has been analysed by modelling two significant systems and

comparing them with similar ones that have been proposed in other algebraic approaches.

Some extensions of bioκ-calculus rules may be done without difficulties. In this paper we have discussed rules modelling translocations and phagocytosis, even if the latter ones are not very satisfactory. Other biological reactions have not yet been considered and are left to future work, such as those in [6] or in [11].

Extensional semantics of bioκ-calculus are an intriguing issue we plan to investigate in the future. In particular we are interested in mathematical tools and techniques that help in assessing properties of biological solutions. Such tools might include stochastic measures in the style of [14] and might be extensively used to predict outputs of experiments in vitro.

Acknowledgements. We thank Gianluigi Zavattaro for the discussions and the suggestions on the encodings of the bioκ-calculus into π-calculus.

References

[1] R. Milner, Communicating and mobile systems: the π-calculus, Cambridge University Press, Cambridge, 1999.

[2] A. Regev, W. Silverman, E. Shapiro, Representation and simulation of biochemical processes using the π-calculus process algebra, in: R. B. Altman, A. K. Dunker, L. Hunter, T. E. Klein (Eds.), Pacific Symposium on Biocomputing, Vol. 6, World Scientific Press, Singapore, 2001, pp. 459–470. URL http://www.smi.stanford.edu/projects/helix/psb01/ regev.pdf

[3] V. Danos, C. Laneve, Formal molecular biology, Theoritical Computer Science 325 (1) (2004) 69–110.

[4] L. Cardelli, A. D. Gordon, Mobile ambients., Theoritical Computer Science 240 (1) (2000) 177–213.

[5] G. Paun, Membrane computing. an introduction., Springer-Verlag, Berlin, 2002.

[6] L. Cardelli, Brane calculi., in: CMSB, 2004, pp. 257–278.

[7] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, E. Shapiro, Bioambients: an abstraction for biological compartments, Theoretical Computer Science 325 (1) (2004) 141–167.

[8] R. Milner, Communication and Concurrency, International Series on Computer Science, Prentice Hall, 1989.

[9] V. Danos, F. Tarissan, Self-assembling graphs, in: J. Mira, J. R. Álvarez (Eds.), Mechanisms, Symbols, and Models Underlying Cognition, Vol. 3561 of Lecture Notes in Computer Science, Springer, 2005, pp. 498–507.

[10] D. Sangiorgi, From π-calculus to Higher-Order π-calculus — and back, in: M.-C. Gaudel, J.-P. Jouannaud (Eds.), Proc. TAPSOFT'93, Vol. 668 of Lecture Notes in Computer Science, 1993, pp. 151–166.

[11] V. Danos, S. Pradalier, Projective brane calculus., in: CMSB, 2004, pp. 134–148.

[12] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, Journal of Information and Computation 100 (1992) 1–77.

[13] J. Parrow, B. Victor, The fusion calculus: Expressiveness and symmetry in mobile processes, in: Proceedings of LICS '98, IEEE, Computer Society Press, 1998, pp. 176–185.

[14] A. Credi, M. Garavelli, C. Laneve, S. Pradalier, S. Silvi, G. Zavattaro, Modelization and simulation of nano devices in nanoκ calculus, in: CMSB, Vol. 4695 of Lecture Notes in Computer Science, Springer, 2007, pp. 168–183.