Scientific Annals of Computer Science vol. 24 (1), 2014, pp. 91–136
doi: 10.7561/SACS.2014.1.91

A Flat Process Calculus for Nested Membrane Interactions

Chiara BODEI[1], Linda BRODO[2], Roberto BRUNI[3], Davide CHIARUGI[4]

Abstract

The `link`-calculus has been recently proposed as a process calculus for representing interactions that are open (i.e. that the number of processes may vary), and multiparty (i.e. that may involve more than two processes). Here, we apply the `link`-calculus for expressing, possibly hierarchical and non dyadic, biological interactions. In particular, we provide a natural encoding of Cardelli's Brane calculus, a compartment-based calculus, introduced to model the behaviour of nested membranes. Notably, the `link`-calculus is flat, but we can model membranes just as special processes taking part in the biological reaction. Moreover, we give evidence that the `link`-calculus allows one to directly model biological phenomena at the more appropriate level of abstraction.

Keywords: Membrane Interactions, `link`-calculus, Brane Calculus

1 Introduction

In the last years, many process algebras have been adapted for modelling and analysing biological systems, by exploiting the similarity between distributed and concurrent systems and complex biological systems. Species can therefore

[1]Dipartimento di Informatica, Università di Pisa, Italy, Email: `chiara@di.unipi.it`
[2]Dipartimento di Scienze Politiche, Scienze della Comunicazione e Ingegneria dell'Informazione, Università di Sassari, Italy, Email: `brodo@uniss.it`
[3]Dipartimento di Informatica, Università di Pisa, Italy, Email: `bruni@di.unipi.it`
[4]Department of Theory and Bio Systems, Max Planck Institute of Colloids and Interfaces, Potsdam, Germany, Email: `davide.chiarugi@mpikg.mpi.de`

be abstractly seen as interacting processes and reactions as actions, thus helping the modeller to focus on the relationships among the species. As a dowry, process algebras provide a formal setting for describing biological phenomena and ready-to-use analysis methodologies and tools.

The symbolic abstraction offered by process algebras is crucial for managing the complexity of biological mechanisms, especially if it can be joined to a flexible modelling strategy able to switch between different levels of abstraction.

We focus here on process calculi that model biological compartments or membranes and their interactions, and in particular on Brane Calculi (presented in Section 5), a family of process calculi, introduced by Cardelli in [8], endowed with dynamically nested membranes, based on location mobility and awareness. Membrane interactions are described into terms of synchronisations between pairs of co-actions. Nevertheless, on a closer view, these apparently dyadic interactions are inherently multiparty. Each interaction not only involves the two processes triggering the interaction, by offering the required co-actions, but it also involves one or more membranes, as supporting actors. Each interaction leads indeed to a new membrane hierarchy.

To better capture this multiparty nature of interactions, we resort to a flat calculus, the `link`-calculus [4] (recalled in Section 3), whose primitives are designed for describing and coordinating interactions that are multiparty, i.e. that may involve more than two processes and are open, i.e. the number of involved processes is not fixed or known a priori by the other partners. The `link`-calculus is an extension of the π-calculus, where dyadic interactions are replaced by a more general synchronisation algebra.

The main difference, with respect to the π-calculus, is that in the `link`-calculus communication actions are given in terms of links, that record the source and the target ends of each hop of interaction. Links can be indeed combined in link chains that route information across processes from a source to a destination. Intuitively, an interaction can be imagined like the composition of a one-dimensional jigsaw puzzle: it requires different pieces to fit together, where the pieces are the links that, together with the objects of communication, are offered by each involved action. Links can be joined in a link chain if they are to some extent "complementary", i.e. if each process contributes with links that are not specified by others. According to the puzzle analogy, different parts of the chain can be composed separately, and, afterwards, assembled without overlays.

The `link`-calculus has been introduced and exploited in [4] to provide an encoding of Mobile Ambients [9], from which many bio-inspired calculi derive, included the Brane Calculi. Mobile Ambients and its descendant calculi are all based on compartmentalisation and on location-awareness. Ambients are bounded locations representing the mobile computational units in which agents interact. The `link`-calculus can naturally capture the spatial aspects due to ambient nesting and the multiparty essence of ambient interactions.

Supported by the "genetic" resemblance with Mobile Ambients, we provide an encoding of Brane Calculi in the `link`-calculus, by following the line of [4]. Also in the case of Brane Calculi, by using the `link`-calculus we can easily handle locality, without introducing any specific operator, just encoding any *membrane compartment* as a separate process. Furthermore, multiparty interactions are naturally rendered.

One of the peculiar features of Brane calculi is the design of different sets of primitives to capture different biological mechanisms, and to choose different levels of granularity for the studied biological compartments. We first encode the MBD version of Brane calculi, and then we introduce similar encodings for the PEP version of the calculus, and for a subset of the calculus able to handle the interaction between molecules. Our flat calculus easily accommodates each new set of primitives. Under this regard, our calculus can be seen as a sort of universal *low-level language* for the family of membrane-based calculi, offering a basic set of interactions able to simply implement the several more abstract primitives of Brane. This shows the expressivity of `link`-calculus, both from the linguistic and biological perspective.

At the same time, we show that the `link`-calculus can also directly represent high-level molecular interactions and biological phenomena, by using exactly the same principles. Actually, an expressivity problem arises when we consider that, in general, biological interactions, such as reactions, can involve more than two elements at the time, but are modelled, by usual biologically-oriented process calculi, as suitable sequences of binary reactions. With the `link`-calculus, we do not have this limitation in the arity of participants in a reaction any longer. We can indeed — as shown in the next Section — directly model a biological process involving multiparty interactions in the `link`-calculus, without passing from another process algebra. One advantage is a tight operational correspondence between biological reactions and process reactions: while biological reductions are often encoded as sequences of interactions in ordinary process calculi, they can be represented as atomic interactions in the `link`-calculus. Furthermore,

in modelling we can choose the abstraction level, depending on the current interest.

Related Works In [19, 20], there are other proposals for changing the abstraction level, using the same formalism: the probabilistic model checker PRISM is indeed applied to simulate biological case studies, both at the individual level, and at the level of interacting populations of cells. In [19] also membrane compartments are modelled. To allow the specification of multi-reactant multi-product reactions in BlenX [16], the work in [13] proposes an extension to model a sequence of elementary actions as if it were atomic. More similar in spirit to our work is instead [27], where an extension of the π-calculus, $\pi@$, is introduced that provides an encoding of Brane Calculi. This calculus is flat, but with dyadic interactions: localisation is expressed by means of polyadic synchronisation, while atomic operations are expressed by means of constraints on the order of communications, obtained with priorities.

Several approaches, among which we recall [17, 22, 24, 5, 6], present encodings of Mobile Ambients. The main motivation is to provide an LTS semantics to Mobile Ambients, as a basis for bisimulation congruence. Also, in [12], the authors present an encoding of the mobile ambients without communication into a subset of the π-calculus, based on the separation of the spatial structure of mobile ambients from their operational semantics. While in all these works *ad hoc* semantics are introduced, the encoding proposed in [4], and the one presented here for Brane Calculi are just illustrative examples of application of our `link`-calculus.

The basic computer interaction, where a message is sent and received by two different processes, is dyadic and there is a long tradition in studying the properties of this kind of interaction. The seminal work [26] that opens the study of the process algebra for modelling biological processes resorts to the π-calculus as it is one with a strong established theory. Until now, one of the main challenges in this area is how translating the classical properties of distributed systems into biological terms. Nevertheless, from the point of view of the intrinsic dynamics of biological processes, the interactions are not dyadic and their formalisation in terms of input-output activities is a strong simplification.

Our proposal goes in the direction of defining a multiparty calculus suitable to compose a large variety of biological interactions, different in the number of their participants and on their level of abstraction (biochemical,

molecular, cellular, etc.), sufficiently general to easily incorporate most of the relevant aspects in biology, such as locality, shapes, concentrations, etc.

Structure of the paper. In Section 2, we give a first introduction to the `link`-calculus and we show its flexibility in modelling biological interactions, by giving a model of the Receptor-mediated Endocytosis mechanism. The example is then resumed and concluded in Section 4, after the formal introduction of the syntax and operational semantics of the `link`-calculus in Section 3. In Section 5, we recall the syntax and operational semantics of the families of Brane calculi, while in Section 6 we show how to encode Brane calculi in the `link`-calculus. In Section 7, we present the `link`-calculus encoding of the Brane Calculus description of the Semliki Forest virus [8] as a concrete example of the translation. Some concluding remarks are drawn in Section 8.

2 The Receptor-mediated Endocytosis Example

Modelling complex biological phenomena is not a trivial task. Consider, for instance, the process known as *receptor-mediated endocytosis* [1], a very general mechanism that allows living cells to transport specific substances (ligands) from the external environment to the cytoplasm.

This process, depicted in Figure 1, consists of various steps and involves the formation of vesicles. The first step occurs on the outer side of the cell membrane, and is represented by the binding between the ligands and a set of specialised molecules (*receptors*), embedded in the cell membrane. The formation of the *ligand-receptor complex* (LR) triggers further events close to the inner side of the cell membrane. First, one molecule of the AP-2 complex[5] binds to each LR complex, thus allowing the clustering of the LR complexes and the subsequent link with a number of proteins called CLATHRIN. These molecules elicit the invagination of the part of the cell membrane that resides close to the receptors, and eventually leads to the formation of a vesicle that contains the LR complex and is coated by an external layer of AP-2 and clathrin molecules. Then, this vesicle moves along the cytoplasm towards other vesicles (*endosomes*). Finally, the vesicle coalesces with the endosome (fusion), immediately after the dissociation of the external AP-2-clathrin coat (uncoating).

[5]The AP-2 complex consists of two large adaptins (α and $\beta 2$), a medium adaptin ($\mu 2$), and a small adaptin ($\sigma 2$).

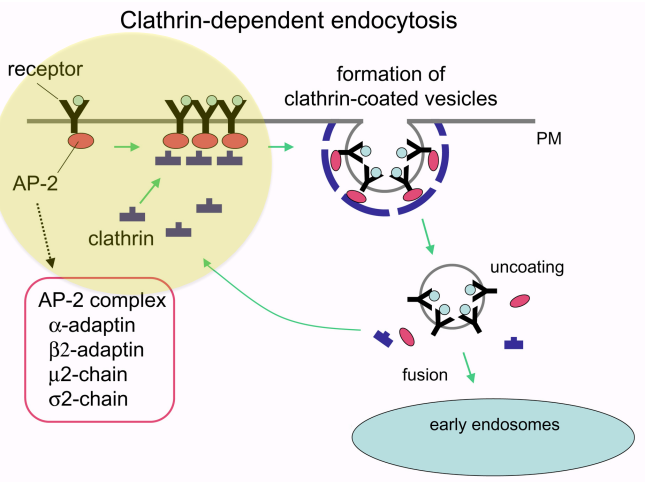

Figure 1: The Receptor-mediated Endocytosis.

We focus our attention on the first two steps: (1) the binding between the ligands and a set of specialised molecules (*receptors*) embedded in the cell membrane; and (2) the further events close to the inner side of the cell membrane triggered by the formation of the *ligand-receptor complex* (LR): (a) the binding of one molecule of the AP-2 complex (a.k.a. adaptor complex) to each LR complex, and (b) the following clustering of the LR complexes and (c), finally, the subsequent link with a number of proteins called CLATHRIN.

To obtain the overall picture of this process, we need to address two different abstract levels, usually modelled separately: the level of membranes and the one of molecules.

The flexibility of the `link`-calculus is exploited here to model both molecular and membrane interactions, without resorting to any extension of the language. This example will also give us the chance to smoothly introduce our calculus, that will be formally presented in Section 3.

In the first part of the example, there are different possibilities for modelling the biological interactions, depending on the particular object of

study; we can choose to model the engulfing of the LIGAND:

(a) as a single multiparty interaction, that involves the cell membrane, the LIGAND, the RECEPTOR, and the two molecules AP-2 and CLATHRIN;

(b) or by resorting to two steps, by making explicit the binding between the ligands, the receptors, and the proteins AP-2, then the ligand between the ligand-receptor-AP-2 complex and the proteins CLATHRIN;

(c) or by choosing a lower abstract level, not considering membranes, and only focussing on the underlying chemical reactions.

Here, we follow the second solution (b), and we model the first part of the example as the two following `link`-calculus multiparty interactions:

- the interaction among the LIGAND, the RECEPTOR, and the cell membrane, which is considered as an active participant; and
- the interaction between the complex LIGAND-RECEPTOR and the proteins AP-2 and CLATHRIN.

In Figure 2, there are the `link`-calculus specifications of the involved elements, whose names are written in capital letters, e.g. LIGAND or RECEPTOR, and where the subscript records the name of the compartment where each element lies (e.g. LIGAND_{top} represents the element LIGAND lying at *top* level in the membrane hierarchy). In membranes also the parent compartments are recorded as superscripts (as in the previous examples): e.g. M_{cell}^{top} is the cellular membrane, lying at the top (of our model).

In traditional process algebras, processes communicate by means of actions and co-actions that synchronise on the same channel and possibly exchange data. In `link`-calculus, interactions are multi-party and can therefore involve more than two processes. Actions come with *links*, i.e. pairs ${}^{\alpha}\backslash_{\beta}$ that record the source and the destination ends of each hop of a communication. At the same time, actions offer a communication tuple $\langle \vec{w} \rangle$, whose names can be used either as values (e.g. *cell*) or as variables (that we distinguish by underlining them, e.g. $\underline{x_{cell}}$). In the cell membrane process, in the link part $({}^{\cdots}\backslash_{\ldots})$ we specify the binding elements placed on the membrane (in our case, the external and the internal part of the receptor), while in the tuple, we specify the two involved localities (*top* and *cell*). Instead, in the link part of the others biological elements, we specify the element

name, on the left side, and the element name that can be bound to, on the right side ($^{ElementName}\backslash_{BindingElement}$). As before, we specify the element location in the tuple. Each party offers its link and its tuple, and a multi-party interaction is possible when they can be suitably combined together, i.e. when:

- links match two by two, e.g. $^{\tau}\backslash_{memRec}$ with $^{memRec}\backslash_{receptor}$; and when
- all the parties agree on the same tuple pattern, by possibly matching values in corresponding position, and by binding the remaining variables with the corresponding values, which here represent the names of their own locations. For instance, in the combination of the tuples $\langle top, \underline{x_{mem_{out}}}, \underline{x_{mem_{in}}}, \underline{x_{cell}}\rangle$ and $\langle \underline{x_{top}}, mem_{out}, \underline{x_{mem_{in}}}, \underline{x_{cell}}\rangle$, the value top can be assigned to the variable $\underline{x_{top}}$, while the value mem_{out} can be assigned to the variable $\underline{x_{mem_{out}}}$.

Therefore, links are combined in a link chain, like pieces in a jigsaw puzzle, where each party contributes with its link. More precisely, we can think about S-shaped tetrominos (see the first link chain produced in our example in Figure 3), joined together when the labels of the edges match.

The following first transition allows the formation of the ligand-receptor-AP2 complex, while the second transition produces $COMPLEX_{cell}$ containing the vesicle membrane and the complex ligand-receptor-AP2-clathrin. The definition of COMPLEX$_{cell}$ is given in Figure 4.

$$\text{LIGAND}_{top}|M^{top}_{cell}|\text{RECEPTOR}_{cell}|\text{AP-2}_{cell}|\text{CLATHRIN}_{cell}$$

$$\downarrow {}^{\tau}\backslash^{memRec}_{memRec}\backslash^{receptor}_{receptor}\backslash^{ap2}_{ap2}\backslash_{\tau}$$

$$M^{top}_{cell}|\text{LIG-REC-AP2}_{cell}|\text{CLATHRIN}_{cell}$$

$$\downarrow {}^{\tau}\backslash^{clathrin}_{clathrin}\backslash_{\tau}$$

$$M^{top}_{cell}|\text{COMPLEX}_{cell}$$

3 The Calculus of Linked Interactions

In the previous section, we intuitively introduce the *calculus of linked interactions*, (`link`-*calculus* for short). Now, we formally introduce its main

$\text{LIGAND}_{top} | M^{top}_{cell} | \text{RECEPTOR}_{cell} | \text{AP-2}_{cell} | \text{CLATHRIN}_{cell}$ `(whole system)`

$\text{LIGAND}_{top} \triangleq {}^{\tau}\backslash_{memRec}\langle top, \underline{x_{mem_{out}}}, \underline{x_{mem_{in}}}, \underline{x_{cell}}\rangle.\mathbf{0}$ `(outside the cell)`

$M^{top}_{cell} \triangleq recX.({}^{memRec}\backslash_{receptor}\langle \underline{x_{top}}, mem_{out}, \underline{x_{mem_{in}}}, \underline{x_{cell}}\rangle.X$ `(the cell membrane)`

$\text{RECEPTOR}_{cell} \triangleq {}^{receptor}\backslash_{ap2}\langle \underline{x_{top}}, \underline{x_{mem_{out}}}, mem_{in}, \underline{x_{cell}}\rangle.\text{LIG-REC-AP2}_{cell}$ `(on the cell membrane)`

$\text{AP-2}_{cell} \triangleq {}^{ap2}\backslash_{\tau}\langle \underline{x_{top}}, \underline{x_{mem_{out}}}, \underline{x_{mem_{in}}}, cell\rangle.\mathbf{0}$ `(within the cell)`

$\text{CLATHRIN}_{cell} \triangleq {}^{clathrin}\backslash_{\tau}\langle \underline{x}, cell\rangle.\mathbf{0}$ `(within the cell)`

$\text{LIG-REC-AP2}_{cell} \triangleq {}^{\tau}\backslash_{clathrin}\langle cell, \underline{y}\rangle.COMPLEX_{cell}$ `(within the cell)`

Figure 2: Model of the formation of the complex `ligand−receptor−ap2−clathrin`.

ingredients, and we briefly recall its syntax and semantics (for further details and examples we refer to [4]). The key feature of this language is that communication actions are given in terms of *links*. A link is a pair that records the source and the target ends of a communication, meaning that the input available at the source end can be forwarded to the target one. Links are combined in link chains to describe the way information can be routed across ends.

Let $\mathcal{C}$ be the set of *channels*, ranged over by $a, b, c, ...$, and let $\mathcal{C} \cup \{\tau, *\}$ be the set of *actions*, ranged over by $\alpha, \beta, \gamma, ...$, where the symbol τ denotes a *silent* action, and the symbol $*$ denotes a non-specified action (i.e. a missing piece of the puzzle according to the analogy proposed before).

A *link* is a pair $\ell = {}^{\alpha}\backslash_{\beta}$: we call α the *source end* of ℓ and β the *target end* of ℓ. A link $\ell = {}^{\alpha}\backslash_{\beta}$ is *valid* if either $\alpha, \beta \neq *$ or $\ell = {}^{*}\backslash_{*}$. In the first case, the link is called *solid*. The link ${}^{*}\backslash_{*}$ is called *virtual*. We let $\mathcal{L}$ be the

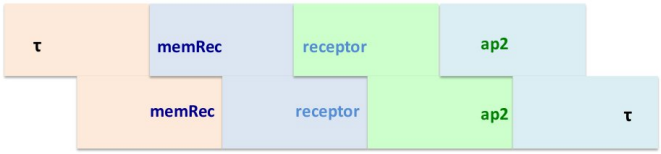

Figure 3: Link Composition seen as Combining Tetrominos with Matching Labels.

set of valid links. Examples of non valid links are ${}^{\tau}\backslash_{*}$ and ${}^{*}\backslash_{a}$, while both ${}^{\tau}\backslash_{a}$ and ${}^{a}\backslash_{b}$ are valid links.

A *link chain* is a finite sequence $s = \ell_1 ... \ell_n$ of (valid) links $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ such that:

1. for any $i \in [1, n-1]$, $\begin{cases} \beta_i, \alpha_{i+1} \in \mathcal{C} & \text{implies } \beta_i = \alpha_{i+1} \\ \beta_i = \tau & \text{iff } \alpha_{i+1} = \tau \end{cases}$
2. if $\forall i \in [1, n].\alpha_i, \beta_i \in \{\tau, *\}$, then $\forall i \in [1, n].\alpha_i = \beta_i = \tau$.

The first condition says that any two adjacent solid links must agree on their ends: it also imposes that τ cannot be matched by $*$. The second condition disallows chains made of virtual links only. The empty link chain is denoted by ϵ. A non-empty link chain is *solid* if all its links are so. For example, ${}^{\tau}\backslash^{a}_{a}\backslash_{b}$ is a solid link chain, while ${}^{\tau}\backslash^{*}_{a}\backslash^{b}_{*}\backslash_{\tau}$ is not solid.

The `link`-calculus processes are defined by the following grammar:

$$P, Q ::= \mathbf{0} \mid X \mid \ell t.P \mid P + Q \mid P|Q \mid (\nu\, a)P \mid P[a/b] \mid \mathsf{rec}X.\, P$$

where ℓ is a solid link (i.e. $\ell = {}^{\alpha}\backslash_{\beta}$ with $\alpha, \beta \neq *$) and t is a tuple of names.

Roughly, processes are built over a standard syntax (with nil process $\mathbf{0}$, prefix $\ell t.P$, sum $P + Q$, parallel $P|Q$, restriction $(\nu\, a)P$, renaming $P[a/b]$, recursion $\mathsf{rec}X.\, P$, for X a process variable), but where the underlying synchronisation algebra is based on link chains.

The calculus is equipped with the name passing mechanism, borrowed from the (polyadic) π-calculus, but slightly extended. Each link in the whole chain simply carries the same list of arguments, but with different (send/receive) capabilities. We assume channel names are admissible values, i.e. as in π-calculus, we have the possibility to communicate (names of) means of communication. In the tuple $t = \langle \vec{w} \rangle$, names can be used either as values or as variables. To be distinguished, variables are underlined.

$$
\begin{array}{ccc}
P + \mathbf{0} \equiv P & P + Q \equiv Q + P & (P + Q) + R \equiv P + (Q + R) \\
P \mid \mathbf{0} \equiv P & P \mid Q \equiv Q \mid P & (P \mid Q) \mid R \equiv P \mid (Q \mid R) \\
(\nu\, n)\mathbf{0} \equiv \mathbf{0} & & (\nu\, n)(\nu\, m)P \equiv (\nu\, m)(\nu\, n)P \\
recX.P \equiv P\{recX.P/P\} & (\nu\, n)(P \mid Q) \equiv P \mid (\nu\, n)Q, \text{ if } n \notin fn(P) &
\end{array}
$$

Table 1: Congruence rules of the `link`-calculus.

This mechanism allows, e.g. a form of multi-way communication, where all peers involved in the link chain can express arguments to be matched, and provide actual arguments to replace the formal ones of other peers. In general, the same peer can atomically (i.e. executing one single prefix) send some values and receive other values. For a tuple t, we let $vals(t)$ and $vars(t)$ denote the set of values and the set of variables of t, respectively. We say that a tuple t is *ground* if $vars(t) = \emptyset$. For example, the processes $P = {}^{\tau}\backslash_{a}\langle v\rangle.P'$ and $Q = {}^{a}\backslash_{\tau}\langle \underline{x}\rangle.Q'$ can interact by forming the link chain ${}^{\tau}\backslash^{a}_{a}\backslash_{\tau}$ with P sending the value v to Q (x will be substituted by v in Q' after the interaction). As a more general example, processes $P = {}^{\tau}\backslash_{a}\langle v, \underline{y}, m\rangle.P'$ and $Q = {}^{a}\backslash_{\tau}\langle \underline{x}, w, m\rangle.Q'$ can interact by forming again the link chain ${}^{\tau}\backslash^{a}_{a}\backslash_{\tau}$ with P sending the value v to Q (as before), Q sending the value w to P (y will be substituted by w in P' after the interaction) and both P and Q agreeing over the third argument m of the communication (which is matched in both tuples).

As usual, $(\nu\, a)P$ binds the occurrences of a in P. Similarly to $(\nu\, a)$, the prefix $\ell t.P$ binds the occurrences of the variables $vars(t)$ in P. The sets of free and of bound names of a process P are defined in the obvious way and denoted, respectively, by $fn(P)$ and $bn(P)$. Processes are taken up to alpha-conversion of bound names, and we shall often omit trailing $\mathbf{0}$, and empty tuples, e.g. by writing ${}^{a}\backslash_{b}$ instead of ${}^{a}\backslash_{b}\langle\rangle.\mathbf{0}$. In the following, given two set of names S and T, we write $S\#T$ as a shorthand for $S \cap T = \emptyset$.

The operational semantics is defined in terms of an LTS, whose states are `link`-calculus processes, whose labels are pairs sg of link chains s and tuples g, and whose transitions are generated by the SOS rules in Table 2, by relying on the congruence rules in Table 1. In a label of the form sg, if s is solid, then g must be the empty tuple $\langle\rangle$, i.e. it is not possible to observe the arguments of a completed communication.

We denote by $\sigma = [x_1 \mapsto v_1, ..., x_n \mapsto v_n]$ the substitution that replaces each x_i with v_i, and is the identity otherwise, and set $vars(\sigma) = \{\, x_1, ..., x_n \,\}$. Substitution is defined, as usual, in order to avoid name captures, and can

$$\frac{\ell \in s \qquad g \preceq_\sigma t}{\ell t.P \xrightarrow{sg} P\sigma}\ (\text{Act}) \qquad \frac{P \xrightarrow{sg} P'}{P + Q \xrightarrow{sg} P'}\ (\text{Lsum})$$

$$\frac{P \xrightarrow{sg} P' \qquad a \notin g}{(\nu\, a)P \xrightarrow{(\nu\, a)(sg)} (\nu\, a)P'}\ (\text{Res}) \qquad \frac{P \xrightarrow{sg} P' \qquad a \in g}{(\nu\, a)P \xrightarrow{(\nu\, a)(sg)} P'}\ (\text{Open})$$

$$\frac{P[X \mapsto \mathsf{rec}X.\, P] \xrightarrow{sg} P'}{\mathsf{rec}X.\, P \xrightarrow{sg} P'}\ (\text{Rec}) \qquad \frac{P \xrightarrow{sg} P' \qquad ex(g)\#fn(Q)}{P|Q \xrightarrow{sg} P'|Q}\ (\text{Lpar})$$

$$\frac{P \xrightarrow{sg} P' \qquad Q \xrightarrow{s'g'} Q' \qquad \begin{array}{l} ex(g)\#fn(Q) \\ ex(g')\#fn(P) \end{array} \qquad s \bullet s' \text{ is not solid}}{P|Q \xrightarrow{sg \bullet s'g'} P'|Q'}\ (\text{Com})$$

$$\frac{P \xrightarrow{sg} P' \qquad Q \xrightarrow{s'g'} Q' \qquad \begin{array}{ll} ex(g)\#fn(Q) & s \bullet s' \text{ is solid} \\ ex(g')\#fn(P) & g \bullet g' \text{ is ground} \end{array}}{P|Q \xrightarrow{s \bullet s'} (\nu\, ex(g \bullet g'))(P'|Q')}\ (\text{Close})$$

$$\frac{P \xrightarrow{sg} P' \qquad \sigma = [b \mapsto a]}{P[a/b] \xrightarrow{(sg)\sigma} P'[a/b]}\ (\text{Ren}) \qquad \frac{P' \equiv P \qquad P \xrightarrow{sg} Q \qquad Q \equiv Q'}{P' \xrightarrow{sg} Q'}\ (\text{Struct})$$

Table 2: SOS Semantics of the `link`-calculus (rules (*Rsum*) and (*Rpar*) omitted).

be applied to tuples, links, and to processes (see [4] for further details).

We say that g is a *full instance* of t and write $g \preceq_\sigma t$ if $vars(\sigma) = vars(t)$ and $g = t\sigma$.

Certain actions of the link chain can be hidden, by restricting the channel where they take place. Formally, for $s = \ell_1 ... \ell_n$, with $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ for $i \in [1, n]$, we define the restriction operation $(\nu\, a)s$ by letting[6]

[6]This operation and the following ones are not defined in all the cases not explicitly mentioned.

$$
\begin{array}{rcl}
(\nu\, a)s & \triangleq & {}^{\alpha_1}\backslash(\nu\, a)({}^{\alpha_2}_{\beta_1})\backslash ... \backslash(\nu\, a)({}^{\alpha_n}_{\beta_{n-1}})\backslash_{\beta_n} \text{ if } \alpha_1, \beta_n \neq a \\
(\nu\, a)({}^{\alpha}_{\beta}) & \triangleq & \begin{cases} {}^{\tau}_{\tau} & \text{if } \alpha = \beta = a \\ {}^{\alpha}_{\beta} & \text{if } \alpha, \beta \neq a \end{cases}
\end{array}
$$

Based on this definition and like in the π-calculus, names in the tuple can be extruded during the communication. In the labels of transitions, we need to annotate positions in the tuple to distinguish between arguments that are taken in input (i.e. they are guessed instances), or that are extruded. We underline the former ones, while we overline the latter ones. A name can be extruded when it is not already annotated; after the extrusion, it will be overlined. Formally, given a link chain s, an (annotated) tuple g, $(\nu\, a)(sg)$ and $(\nu\, a)g$ are defined as follows:

$$
\begin{array}{rcl}
(\nu\, a)(sg) & \triangleq & ((\nu\, a)s)((\nu\, a)g) \\
(\nu\, a)\langle w_1, ..., w_n\rangle & \triangleq & \langle(\nu\, a)w_1, ..., (\nu\, a)w_n\rangle
\end{array}
$$

$$
(\nu\, a)w \triangleq \begin{cases} w & \text{if } w \neq a, \overline{a}, \underline{a} \\ \overline{a} & \text{if } w = a \end{cases}
$$

We let $ex(g)$ denote the set of extruded (i.e. overlined) names appearing in g, and extend the definition of substitution application to overlined names in the obvious way. We write $a \in g$ if the name a appears in the tuple g (with or without annotation).

Two link chains can be merged if they are to some extent "complementary", i.e. if they have the same length, each provides links that are not specified in the other, and together they form a (valid) link chain. Formally, for $s = \ell_1 ... \ell_n$ and $s' = \ell'_1 ... \ell'_n$, with $\ell_i = {}^{\alpha_i}\backslash_{\beta_i}$ and $\ell'_i = {}^{\alpha'_i}\backslash_{\beta'_i}$ for any $i \in [1, n]$, we define $s \bullet s'$ by letting:

$$
\begin{array}{rcl}
s \bullet s' & \triangleq & (\ell_1 \bullet \ell'_1)...(\ell_n \bullet \ell'_n) \\
{}^{\alpha}\backslash_{\beta} \bullet {}^{\alpha'}\backslash_{\beta'} & \triangleq & {}^{(\alpha \bullet \alpha')}\backslash_{(\beta \bullet \beta')}
\end{array}
\qquad
\alpha \bullet \beta \triangleq \begin{cases} \alpha & \text{if } \beta = * \\ \beta & \text{if } \alpha = * \end{cases}
$$

Two annotated tuples can be merged when they list exactly the same values in the same order, and if the values in matching positions are annotated in some compatible way. Formally, if $g = \langle w_1, ..., w_n\rangle$ and $g' = \langle w'_1, ..., w'_n\rangle$:

$$
\begin{array}{rcl}
sg \bullet s'g' & \triangleq & (s \bullet s')(g \bullet g') \\
g \bullet g' & \triangleq & \langle w_1 \bullet w'_1, ..., w_n \bullet w'_n\rangle
\end{array}
$$

$$w \bullet w' \triangleq \begin{cases} w & \text{if } (w = w' = v) \vee (w = w' = \underline{v}) \\ v & \text{if } (w = v \wedge w' = \underline{v}) \vee (w = \underline{v} \wedge w' = v) \\ \overline{v} & \text{if } (w = \overline{v} \wedge w' = \underline{v}) \vee (w = \underline{v} \wedge w' = \overline{v}) \end{cases}$$

The SOS rules in Table 2 resemble the early semantic rules of π-calculus. The main difference is that we are dealing with a multi-party form of communication, and, consequently, the "close" rule must be applied when the communication has been completed.

Rule *(Act)* allows the process $\ell t.P$ to offer the tuple t in a communication on the link ℓ. More precisely, following an early style, the actual tuple to be communicated must be a full instance of t (see the condition $g \preceq_\sigma t$): the communication is that of sg, where variables appearing in t are replaced in $g = t\sigma$ by actual parameters. The substitution σ is also applied to the continuation, after the transition ($P\sigma$).

In rules *(Res)* and *(Open)*, we leave implicit the side condition $(\nu\, a)sg \neq \bot$. Rule *(Res)* is applicable whenever $a \notin g$, in which case $(\nu\, a)g = g \neq \bot$ and thus $(\nu\, a)(sg) = ((\nu\, a)s)g$. Rule *(Open)* models the extrusion of a. Note that, since $(\nu\, a)sg \neq \bot$ and $a \in g$, we have that the only possibility for a to appear in g is without annotations (otherwise $(\nu\, a)g = \bot$). Then, by definition of $(\nu\, a)g$, all occurrences of a within g are overlined in $(\nu\, a)g$ (to denote the name extrusion).

In rule *(Com)*, the annotated tuples are "complementary" and can be merged, by merging both the link chains and the two tuples. Note that we leave implicit the side condition $sg \bullet s'g' \neq \bot$, because the $sg \bullet s'g'$ annotates the label of the conclusion transition. In addition, rule *(Com)* checks that (i) the extruded names of one process do not clash with the free names of the other process (like in ordinary π-calculus), and finally that (ii) the communication is not completed yet ($s \bullet s'$ not solid).

Rule *(Close)* checks differ from the *(Com)* one, because *(Close)* is applicable only when the communication has been fully completed and cannot be further extended ($s \bullet s'$ is solid). In this case we must close, i.e. put back the restriction of all names extruded in the communication $((\nu\, ex(g \bullet g')))$. Still, we need to make sure that $g \bullet g'$ has no unresolved input, i.e. that all requested values have been issued ($g \bullet g'$ is ground). Moreover, the observed label is just $s \bullet s'$, as explained before. This is similar to the π-calculus mechanism, according to which the synchronisation of e.g. $a\langle x\rangle$ and $\overline{a}\langle x\rangle$ yields τ and not $\tau\langle x\rangle$.

While rules *(Lsum)*, *(Rsum)*, *(Rec)* and *(Struct)* are straightforward, rules *(Lpar)* and *(Rpar)* need just to check that extruded names of one

process do not clash with free names of the other process (like in ordinary π-calculus).

For some simple examples of the LTS semantics of `link`-calculus, we refer to [4]. Here, we just point out with an example the way in which the merge of link chains is performed. We recall that the definition of the $\bullet$ operator requires that the two involved link chains must be of the same length, and that, by rule *(Act)* we can arbitrarily add virtual links, in any position. Let $s_1 = {}^{\tau}\backslash_{b}$, $s_2 = {}^{b}\backslash_{c}$, and $s_3 = {}^{c}\backslash_{\tau}$ be three links, then there are three possible combinations for applying the $\bullet$ operator:

- we can first define $s'_1 = {}^{\tau}\backslash_{b}{}^{*}\backslash_{*}{}^{*}\backslash_{*}$, $s'_2 = {}^{*}\backslash_{*}{}^{b}\backslash_{c}{}^{*}\backslash_{*}$, and compute $s_4 = s'_1 \bullet s'_2 = {}^{\tau}\backslash_{b}{}^{b}\backslash_{c}{}^{*}\backslash_{*}$; then we can define $s'_3 = {}^{*}\backslash_{*}{}^{*}\backslash_{*}{}^{c}\backslash_{\tau}$, and, finally, we can compute $s = s_4 \bullet s'_3 = {}^{\tau}\backslash_{b}{}^{b}\backslash_{c}{}^{c}\backslash_{\tau}$; or

- we can first define $s'_1 = {}^{\tau}\backslash_{b}{}^{*}\backslash_{*}{}^{*}\backslash_{*}$, $s'_3 = {}^{*}\backslash_{*}{}^{*}\backslash_{*}{}^{c}\backslash_{\tau}$, and compute $s_4 = s'_1 \bullet s'_3 = {}^{\tau}\backslash_{b}{}^{*}\backslash_{*}{}^{c}\backslash_{\tau}$; then we can define $s'_2 = {}^{*}\backslash_{*}{}^{b}\backslash_{c}{}^{*}\backslash_{*}$, and finally we can compute $s = s'_2 \bullet s_4 = {}^{\tau}\backslash_{b}{}^{b}\backslash_{c}{}^{c}\backslash_{\tau}$; or

- we can first define $s'_3 = {}^{*}\backslash_{*}{}^{*}\backslash_{*}{}^{c}\backslash_{\tau}$, $s'_2 = {}^{*}\backslash_{*}{}^{b}\backslash_{c}{}^{*}\backslash_{*}$, and compute $s_4 = s'_2 \bullet s'_3 = {}^{*}\backslash_{*}{}^{b}\backslash_{c}{}^{c}\backslash_{\tau}$; then we can define $s'_1 = {}^{\tau}\backslash_{b}{}^{*}\backslash_{*}{}^{*}\backslash_{*}$; and finally we can compute $s = s'_1 \bullet s_4 = {}^{\tau}\backslash_{b}{}^{b}\backslash_{c}{}^{c}\backslash_{\tau}$.

We will see in the next sections many medium-sized examples of `link`-calculus at work for modelling biological mechanisms and for encoding Brane Calculi processes.

4 Back to the Receptor-mediated Endocytosis Example

Once the calculus has been formally introduced, we can revisit and complete our example, introduced in Section 2, based on the Receptor-mediated Endocytosis mechanism.

To formally derive the first transition of the system, we start by deriving the first part of the synchronisation between the LIGAND_{top} and the membrane M^{top}_{cell}. Note that we have extended the link prefix with virtual links in order to execute the merging of the two links.

$$ {}^{\tau}\backslash^{*}_{memRec}\backslash^{*}_{*}\backslash_{*} \bullet {}^{*}\backslash^{memRec}_{*}\backslash^{*}_{receptor}\backslash^{*}_{*}\backslash_{*} = {}^{\tau}\backslash^{memRec}_{memRec}\backslash^{*}_{receptor}\backslash^{*}_{*}\backslash_{*} : $$

$$\frac{LIGAND_{top} \xrightarrow{\tau\backslash^{*}_{memRec}\backslash^{*}_{*}\backslash_{*}\quad t} \mathbf{0} \qquad M^{top}_{cell} \xrightarrow{{}^{*}\backslash^{memRec}_{*}\backslash^{*}_{receptor}\backslash^{*}_{*}\backslash_{*}\quad t'} M^{top}_{cell}}{LIGAND_{top}|M^{top}_{cell} \xrightarrow{\tau\backslash^{memRec}_{memRec}\backslash_{receptor}\langle top, mem_{out}, \underline{mem_{in}}, \underline{cell}\rangle} \mathbf{0}|M^{top}_{cell}}$$

where the tuples of $LIGAND_{top}$ and of M^{top}_{cell}, i.e. $\langle top, \underline{x_{mem_{out}}}, \underline{x_{mem_{in}}}, \underline{x_{cell}}\rangle$ and $\langle \underline{x_{top}}, mem_{out}, \underline{x_{mem_{in}}}, \underline{x_{cell}}\rangle$ match, and, as a result of their combination, we obtain the tuple $\langle top, mem_{out}, \underline{mem_{in}}, \underline{cell}\rangle$. The value top can be assigned to the variable x_{top}, while the value mem_{out} can be assigned to the variable x_{mem}. Recall that our semantics is in early style.
Now, we proceed our derivation with the synchronisation with the $AP\text{-}2_{cell}$ molecule:

$$\frac{LIGAND_{top}|M^{top}_{cell} \xrightarrow{\tau\backslash^{memRec}_{memRec}\backslash^{*}_{receptor}\backslash^{*}_{*}\backslash_{*}\quad t''} \mathbf{0}|M^{top}_{cell} \qquad AP\text{-}2_{cell} \xrightarrow{{}^{*}\backslash^{*}_{*}\backslash^{*}_{*}\backslash^{ap2}_{\tau}\quad t'''} \mathbf{0}}{LIGAND_{top}|M^{top}_{cell}|AP\text{-}2_{cell} \xrightarrow{\tau\backslash^{memRec}_{memRec}\backslash^{*}_{receptor}\backslash^{ap2}_{*}\backslash_{\tau}\langle top, mem_{out}, \underline{mem_{in}}, cell\rangle} \mathbf{0}|M^{top}_{cell}|\mathbf{0}}$$

where the tuple of the first premise, $\langle top, mem_{out}, \underline{mem_{in}}, \underline{cell}\rangle$ matches with the tuple of the second premise, $\langle \underline{top}, \underline{mem_{out}}, \underline{mem_{in}}, cell\rangle$, and, as a result of their combination, we obtain the tuple $\langle top, mem_{out}, \underline{mem_{in}}, cell\rangle$.

The derivation of the transition can be concluded by the interaction with the $RECEPTOR_{cell}$:

$$RECEPTOR_{cell} \xrightarrow{{}^{*}\backslash^{*}_{*}\backslash^{receptor}_{*}\backslash^{*}_{ap2}\backslash_{*}\langle top, mem_{out}, mem_{in}, cell\rangle} LIG\text{-}REC\text{-}AP2_{cell}$$

that leads us to the final transition, where the tuple of the first premise, $\langle \underline{top}, \underline{mem_{out}}, mem_{in}, \underline{cell}\rangle$, matches with the tuple of the second premise $\langle top, mem_{out}, \underline{mem_{in}}, cell\rangle$, and, as a result of their combination, we obtain the tuple $\langle top, mem_{out}, mem_{in}, cell\rangle$:

$$LIGAND_{top}|M^{top}_{cell}|AP\text{-}2_{cell}|RECEPTOR_{cell}$$

$$\downarrow {}^{\tau\backslash^{memRec}_{memRec}\backslash^{receptor}_{receptor}\backslash^{ap2}_{ap2}\backslash_{\tau}}$$

$$\mathbf{0}|M^{top}_{cell}|\mathbf{0}|LIG\text{-}REC\text{-}AP2_{cell}$$

We can now pass to model the second part of the mechanism. We keep the same modelling style used for the first part, that allow us to model the creation of a vesicle membrane and its fusion with the endosome membrane. As a consequence, we focus our attention on the last two

$$M_{cell}^{top} \triangleq recX.(\ \ldots \qquad\qquad) \\ \qquad\qquad\qquad + \\ \qquad\qquad\qquad {}^{\tau}\backslash_{complex}\langle cell, \underline{y}\rangle.X$$

(the cell membrane)

$$\text{COMPLEX}_{cell} \triangleq {}^{complex}\backslash_{\tau}\langle \underline{x}, cell\rangle.(V_{vsl}^{cell}|\text{COMPLEX}_{vsl})$$

(within the cell)

$$\text{V}_{vsl}^{cell} \triangleq {}^{vsl}\backslash_{endosome}\langle \underline{x_{vsl}}, \underline{x_{new}}\rangle.\mathbf{0}$$

(the vesicle membrane)

$$\text{COMPLEX}_{vsl} \triangleq {}^{\tau}\backslash_{vsl}\langle vsl, \underline{x_{new}}\rangle.(\text{AP-2}_{cell}|\text{CLATHRIN}_{cell}|\text{LIG-REC}_{new})$$

(within the vesicle)

$$\text{ENDOSOME}_{endo}^{cell} \triangleq {}^{endosome}\backslash_{\tau}\langle \underline{x_{vsl}}, new\rangle.NEW_{new}^{cell}$$

(within the cell)

Figure 4: Encoding of the Second Part of the Receptor-mediated Endocytosis Example.

biological interactions needed for the formation of the vesicle (V_{vsl}^{cell}) and for the releasing of the AP−2 and CLATHRIN molecules, as shown in Figure 5.

In Figure 4, we give the `link`-calculus complete code of the example, where we follow the same modelling style used in Section 2. Here, we model the formation of the vesicle as the result of a further interaction between the COMPLEX_{cell} (composed of the LIGAND, the RECEPTOR, and the two AP−2, and CLATHRIN proteins) and the cell membrane. Then, we model the uncoating of the vesicle and its fusion with the ENDOSOME as a single interaction between the membrane vesicle, the COMPLEX_{vsl} (i.e. the same complex located inside the vesicle), and the ENDOSOME. This last reaction releases the AP−2 and the CLATHRIN proteins in the cell, and the fusion between the vesicle and the endosome gives rise to a new membrane inside the cell: NEW_{new}^{cell}.

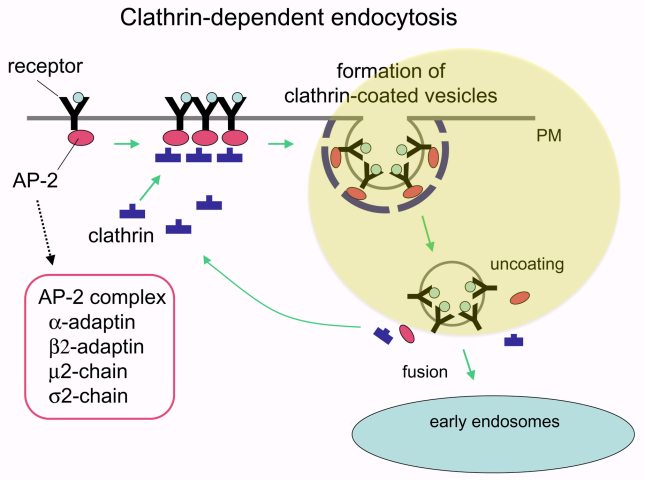

Figure 5: The Receptor-mediated Endocytosis, Second Part.

$$\mathrm{M}^{top}_{cell}|\mathrm{COMPLEX}_{cell}|\mathrm{ENDOSOME}^{cell}_{endo}$$

$$\downarrow {}^{\tau}\backslash^{complex}_{complex}\backslash_{\tau}$$

$$M^{top}_{cell}|\mathrm{COMPLEX}_{vsl}|\mathrm{V}^{cell}_{vsl}|\mathrm{ENDOSOME}^{cell}_{endo}$$

$$\downarrow {}^{\tau}\backslash^{vsl}_{vsl}\backslash^{endosome}_{endosome}\backslash_{\tau}$$

$$M^{top}_{cell}|\mathrm{LIG\text{-}REC}_{new}|\mathrm{AP\text{-}2}_{cell}|\mathrm{CLATHRIN}_{cell}|\mathbf{0}|\mathrm{NEW}^{cell}_{new}$$

`Link`-calculus offers a natural and simple encoding of the Receptor-mediated Endocytosis mechanism, thus encouraging us in investigating the possible use of `link`-calculus in directly modelling other biological phenomena.

5 An Overview on Brane Calculi

The Brane Calculi [8] are a family of calculi for describing biological membrane interactions. Membranes are not only containers, by constituting the boundaries of compartments, but they are also active entities: they represent the place where processes are located and interact with the surrounding environment. In this section, we briefly recall the Mate/Bud/Drip (MBD) version of the Brane Calculi [8], in order to show its encoding in the `link`-calculus, in the next section.

As described by the following syntax, a membrane system consists of nested membranes, where each membrane is associated with a membrane process, that describes its interaction capabilities, in terms of possible membrane transformations.

$$\begin{array}{lll} P, Q ::= & \diamond \mid P \circ Q \mid !P \mid \sigma(\!|P|\!) & \text{systems } \Pi \\ \sigma, \tau ::= & 0 \mid \sigma|\tau \mid !\sigma \mid a.\sigma & \text{membrane processes } \Sigma \\ a, b ::= & mate_n \mid mate_n^{\perp} \mid bud_n \mid bud_n^{\perp}(\rho) \mid drip(\rho) & \text{MBD actions } \Xi_{MBD} \end{array}$$

The basic structure of a system consists of (sub-)system composition, represented by the monoidal operator $\circ$ (associative, commutative and with $\diamond$ as neutral element). Replication ! represents the composition of an unbounded number of replicas of the same system or membrane process. The system $\sigma(\!|P|\!)$ is a *membrane* with content P and interaction capabilities represented by the process σ.

$$(Par)\ \frac{P \to Q}{P \circ R \to Q \circ R} \qquad (Brane)\ \frac{P \to Q}{\sigma(\!|P|\!) \to \sigma(\!|Q|\!)} \qquad (Struct)\ \frac{P \equiv P' \ \wedge\ P' \to Q' \ \wedge\ Q' \equiv Q}{P \to Q}$$

$$(Mate)\quad mate_n.\sigma|\sigma_0(\!|P|\!) \ \circ\ mate_n^{\perp}.\tau|\tau_0(\!|Q|\!) \to \sigma|\sigma_0|\tau|\tau_0(\!|P \circ Q|\!)$$

$$(Bud)\quad bud_n^{\perp}(\rho).\tau|\tau_0(\!|bud_n.\sigma|\sigma_0(\!|P|\!) \circ Q|\!) \to \rho(\!|\sigma|\sigma_0(\!|P|\!)|\!) \circ \tau|\tau_0(\!|Q|\!)$$

$$(Drip)\quad drip(\rho).\sigma|\sigma_0(\!|P|\!) \to \rho(\!|\diamond|\!) \circ \sigma|\sigma_0(\!|P|\!)$$

Table 3: Reduction Semantics for Brane Calculi.

The MBD set of actions are inspired by the possible spatial transformations of biological membranes, such as membrane fusion and splitting. The former is modelled by the *mating* operation, the latter both by *budding*, that consists in splitting off exactly one internal membrane, and *dripping*, that consists in splitting off one empty membrane.

Membrane processes σ consist of the empty process 0, the parallel composition of two processes, represented by the monoidal operator $|$ with 0 as neutral element, the replication of a process and the prefixes of a process by an MBD *action* a. Actions for mating ($mate_n$) and budding (bud_n) have the corresponding co-actions ($mate_n^{\perp}$ and $bud_n^{\perp}$, respectively) to synchronise with. Here n, which identifies a pair of complementary actions, is taken by a countable set Λ of names. The actions $bud_n^{\perp}(\rho)$ and $drip(\rho)$ are equipped with an argument ρ that represent a process, i.e. the actions associated with the membrane created when performing budding and dripping interactions, respectively. As usual, terms of the Brane Calculus can be rearranged according to a structural congruence relation $\equiv$, whose rules (i.e. monoidality of operators $\circ$ and $|$) are omitted for brevity (see [8]). The semantics of the calculus is given by the reduction rules in Table 3. Besides the standard reduction rule for congruence (Struct), and contextual propagation of reductions across parallel composition (Par) and membrane nesting (Brane) (cf. the upper part of Table 3), there are the axioms specific of MBD (cf. lower part of Table 3).

Rule (Mate) models the fusion of two parallel membranes. The effect of a mate synchronisation is twofold: (i) it creates a new membrane; and (ii) it dissolves the two existing ones. In the rule (Bud) a membrane expels a child membrane $\sigma|\sigma_0(\!|P|\!)$ and surrounds it with a new membrane, associated with the process ρ, where ρ is the parameter of the co-action $bud_n^{\perp}$. Finally, in the rule (Drip), a membrane creates a new empty membrane associated with the process ρ, which is the parameter of $drip_n$.

6 From Membranes to Links

6.1 The MBD Brane Calculus

The encoding of the MBD Brane Calculus into the `link`-calculus follows the same style of the one provided for Mobile Ambients in [4]. The encoding relies on the idea that each interaction not only involves the two processes triggering the interaction, but it also has an impact on the membranes and processes of the context, that result reconfigured, after the reaction. To take the implicitly multiparty nature of interactions into account, the encoding offers a `link`-calculus counterpart for each actor involved.

More precisely, the rule ($Mate$) requires a four-party interaction (at least), involving: 1) the membrane process with the capability $mate_n$; 2) the membrane associated with this process; 3) the membrane process with the capability $mate_n^{\perp}$; 4) the membrane associated with this process. The rule has a double effect: (i) it creates a new membrane, and (ii) it dissolves the two existing ones. When the two membranes are dissolved, their contents must be relocated, which may consist of an unbounded (and not known a priori) number of parallel processes, all therefore involved in the interaction! As in [4], we adopt here a syntactic solution with no semantic impact on the rest: we replace the membranes with some sort of blind forwarders (reminiscent of those in [18]) that leave their contents unaware of the deletion.

The rule (Bud) requires four-party interaction as well, while the rule ($Drip$) is simpler, as it only involves a membrane process and its corresponding membrane.

Membranes with Brackets Forwarders impact on the encodings that have to deal with them, as they represent a marker (a side effect) of when and where a membrane has been dissolved. Since in the Brane code, no similar side effect is generated, the correspondence between MBD processes and their encodings is not direct.

To simplify our presentation, we directly introduce forwarders in the syntax of MBD, with no effect whatsoever on the semantics. We only need to slightly modify the syntax with the possibility to enclose a system P or a membrane process σ within a pair of parentheses or *brackets*, as in [4].

$$P ::= \cdots \mid [P] \qquad\qquad \sigma ::= \cdots \mid [\sigma]$$

To make the presence of parentheses inessential with respect to the behaviour of the process, we introduce the additional structural congruence axioms:

$$P \equiv Q \Rightarrow [P] \equiv [Q] \qquad\qquad \sigma \equiv \tau \Rightarrow [\sigma] \equiv [\tau]$$

Finally, we define the notion of *passive context*, both for systems ($\mathbb{C}$), and for processes ($\mathbb{S}$). Passive contexts allow us to adjust the basic reduction rules to deal with the presence of an arbitrary number of balanced parentheses in a monoidal composition.

$$\mathbb{C}, \mathbb{D} ::= \cdot \mid [\mathbb{C}] \mid \mathbb{C} \circ P \mid P \circ \mathbb{C} \qquad\qquad \mathbb{S}, \mathbb{T} ::= \cdot \mid [\mathbb{S}] \mid \mathbb{S}|\sigma \mid \sigma|\mathbb{S}$$

and write $\mathbb{C}(P)$ to denote the system obtained by replacing the hole $\cdot$ in $\mathbb{C}$ with P. Similarly, we write $\mathbb{S}(\sigma)$ to denote the process obtained by replacing the hole $\cdot$ in $\mathbb{S}$ with σ. Thus we add the following suitable axiom:

$$(Mate) \quad \mathbb{C}(\mathbb{S}(mate_n.\sigma)(\!|P|\!)) \circ \mathbb{D}(\mathbb{T}(mate_n^{\perp}.\tau)(\!|Q|\!)) \to \mathbb{C}([\mathbb{S}(\sigma)]|[\mathbb{T}(\tau)](\!|[P] \circ [Q]|\!)) \circ \mathbb{D}(\diamond)$$

$$(Bud) \quad \mathbb{T}(bud_n^{\perp}(\rho).\tau)(\!|\mathbb{C}(\mathbb{S}(bud_n.\sigma)(\!|P|\!))|\!) \to \rho(\!|[\mathbb{S}(\sigma)](\!|[P]|\!)|\!) \circ \mathbb{T}(\tau)(\!|\mathbb{C}(\diamond)|\!)$$

$$(Drip) \quad \mathbb{S}(drip_n(\rho).\sigma)(\!|P|\!) \to \rho(\!|\diamond|\!) \circ \mathbb{S}(\sigma)(\!|P|\!)$$

Furthermore, we introduce the following reduction rule to propagate reductions across contexts.

$$(Brac) \quad \frac{P \to Q}{[P] \to [Q]}$$

In the above rules, we introduce only the contexts involved by the modifications due to the application of the rules. Note that, thanks to the use of contexts, we have eliminated some siblings of the interacting systems and membrane processes, which were necessary in Table 3. More precisely, σ_0 and τ_0 of the original rules are now subsumed by $\mathbb{S}$ and $\mathbb{T}$, respectively (in all rules), while the system Q of the original rule (Bud) is now subsumed by $\mathbb{C}$.

Structural Encoding The encoding function is defined in Figure 6, and works similarly to the ones in [7, 4]:

- The hierarchy of membranes is rendered as a flat system of parallel `link`-calculus processes, where the inclusion relation is recorded by interaction channels.

- The encoding $[\![P]\!]_m$ of a process P is parametric with respect to a given family of channels m that represents the enclosing membrane or, the topmost compartment *top* in which the membranes reside. Informally, each process resides in some "location", to which it addresses all its requests about the next actions to perform.

- Each membrane $\sigma(\!|P|\!)$ is encoded by a `link`-calculus process of the form $M^p_{i,j}$, in parallel with the process $[\![\sigma]\!]_i$ that encodes the associated membrane processes σ, and with the one $[\![P]\!]_j$, encoding the included process P. Intuitively, $M^p_{i,j}$ represents the "control or management unit" of the membrane, a sort of interface that offers a family of channels, or ports, for each kind of action capabilities. Families of channels are indeed indexed by subscripts *mate*, $mate^{\perp}$, *bud*, $bud^{\perp}$ and *drip*. In the following, such families are called *locations* (in Figure 6, families are written for brevity as $m, m', ...$ (and the like)). The links of these channels are *composable* with the corresponding channels of: (i) the encoding of the actual membrane process σ, that represent the active capabilities of the process, (ii) the encoding of the parent membrane, and (iii) the encoding of the included process P. More precisely, in $M^p_{i,j}$, i is the family of channels shared with $[\![\sigma]\!]_i$, j is the family of channels shared with $[\![P]\!]_j$, while p is the family of channels shared with the parent membrane.

- After a fusion, the membrane hierarchy changes: the sub-membranes of the two fused membranes become siblings, and, as a consequence, their interaction capabilities change. On the encoding side, this implies that locations must be updated and that the interactions relative to the processes originally included in the dissolved membrane must be redirected. To perform this task, the translated processes make use of the *forwarding* mechanism every time a membrane is dissolved. The forwarding activity is managed by processes of the form Fwd^p_m, located between the dissolved membranes and the parent ones.

$$
\begin{array}{rcl@{\qquad}rcl}
[\![\diamond]\!]_m & \triangleq & \mathbf{0} & [\![P \circ Q]\!]_m & \triangleq & [\![P]\!]_m \,|\, [\![Q]\!]_m \\
[\![!P]\!]_m & \triangleq & \mathsf{rec}X.\,([\![P]\!]_m | X) & [\![[P]]\!]_m & \triangleq & (\nu\, m')(Fwd^m_{m'} \,|\, [\![P]\!]_{m'}) \\
 & & & [\![\sigma(\!|P|\!)]\!]_m & \triangleq & (\nu\, m_1, m_2)(M^m_{m_1,m_2} \,|\, [\![\sigma]\!]_{m_1} \,|\, [\![P]\!]_{m_2})
\end{array}
$$

$$
\begin{array}{rcl@{\qquad}rcl}
[\![0]\!]_m & \triangleq & \mathbf{0} & [\![\sigma \,|\, \tau]\!]_m & \triangleq & [\![\sigma]\!]_m \,|\, [\![\tau]\!]_m \\
[\![!\sigma]\!]_m & \triangleq & \mathsf{rec}X.\,([\![\sigma]\!]_m | X) & [\![[\sigma]]\!]_m & \triangleq & (\nu\, m')(Fwd^m_{m'} \,|\, [\![\sigma]\!]_{m'})
\end{array}
$$

$$
\begin{array}{rcl}
[\![\, mate_n.\sigma \,]\!]_m & \triangleq & {}^{\tau}\backslash_{m_{mate}}\langle n, \underline{x_1}, \underline{x_2}\rangle.[\![\,\sigma\,]\!]_m \\
[\![\, mate^{\perp}_n.\sigma \,]\!]_m & \triangleq & {}^{m_{mate^{\perp}}}\backslash_{\tau}\langle n, \underline{x_1}, \underline{x_2}\rangle.[\![\,\sigma\,]\!]_m \\
[\![\, bud_n.\sigma \,]\!]_m & \triangleq & {}^{\tau}\backslash_{m_{bud}}\langle n, \underline{x_1}, \underline{x_2}\rangle.[\![\,\sigma\,]\!]_m \\
[\![\, bud^{\perp}_n(\rho).\sigma \,]\!]_m & \triangleq & {}^{m_{bud^{\perp}}}\backslash_{\tau}\langle n, \underline{x_1}, \underline{x_2}\rangle.([\![\,\rho\,]\!]_{x_1} | [\![\,\sigma\,]\!]_m) \\
[\![\, drip(\rho).\sigma \,]\!]_m & \triangleq & {}^{\tau}\backslash_{m_{drip}}\langle \underline{x_1}\rangle.([\![\,\rho\,]\!]_{x_1} | [\![\,\sigma\,]\!]_m)
\end{array}
$$

$$
\begin{array}{rcl}
M^p_{i,j} & \triangleq & \mathsf{rec}X.\quad (\nu\, m_1, m_2)^{i_{mate}}\backslash_{p_{mate}}\langle \underline{name}, m_1, m_2\rangle.(Fwd^{m_1}_i \mid Fwd^{m_2}_j \mid X[m_1/i, m_2/j]) \\
 & & \quad + \; {}^{p_{mate}}\backslash_{i_{mate^{\perp}}}\langle \underline{name}, \underline{x_1}, \underline{x_2}\rangle.(Fwd^{x_1}_i \mid Fwd^{x_2}_j) \\
 & & \quad + \; {}^{i_{bud}}\backslash_{p_{bud}}\langle \underline{name}, \underline{x_1}, \underline{x_2}\rangle.(X[x_2/p]) \\
 & & \quad + \; (\nu\, m_1, m_2)^{j_{bud}}\backslash_{i_{bud^{\perp}}}\langle \underline{name}, m_1, m_2\rangle.(X[m_1/i, m_2/j] \mid X) \\
 & & \quad + \; (\nu\, m_1, m_2)^{i_{drip}}\backslash_{\tau}\langle m_1\rangle.(X[m_1/i, m_2/j] \mid X)
\end{array}
$$

$$
\begin{array}{rcl}
Fwd^p_m & \triangleq & \mathsf{rec}X.\,(\quad {}^{m_{mate}}\backslash_{p_{mate}}\langle \underline{name}, \underline{x_1}, \underline{x_2}\rangle.X \;+\; {}^{p_{mate^{\perp}}}\backslash_{m_{mate^{\perp}}}\langle \underline{name}, \underline{x_1}, \underline{x_2}\rangle.X \\
 & & \quad + \; {}^{m_{bud}}\backslash_{p_{bud}}\langle \underline{name}, \underline{x_1}, \underline{x_2}\rangle.X \;+\; {}^{p_{bud^{\perp}}}\backslash_{m_{bud^{\perp}}}\langle \underline{name}, \underline{x_1}, \underline{x_2}\rangle.X \\
 & & \quad + \; {}^{m_{drip}}\backslash_{p_{drip}}\langle \underline{x_1}\rangle.X
\end{array}
$$

Figure 6: Structural encoding of Brane in `link`-calculus.

We only comment on the translation of the reductions (Mate), (Bud) and (Drip).

As said before, the mate synchronisation is a four-party one. For the sake of clarity, we focus the discussion on the Brane code below, illustrated in Figure 7, where we tag the involved membranes and brackets by a, b, c (see below) to refer them in the text.

$$mate_n.\sigma | \sigma_0(\!|P|\!)_a \circ mate^{\perp}_n.\tau | \tau_0(\!|Q|\!)_b \to [\sigma | \sigma_0]_a | [\tau | \tau_0]_b(\!|[P]_a \circ [Q]_b|\!)_c$$

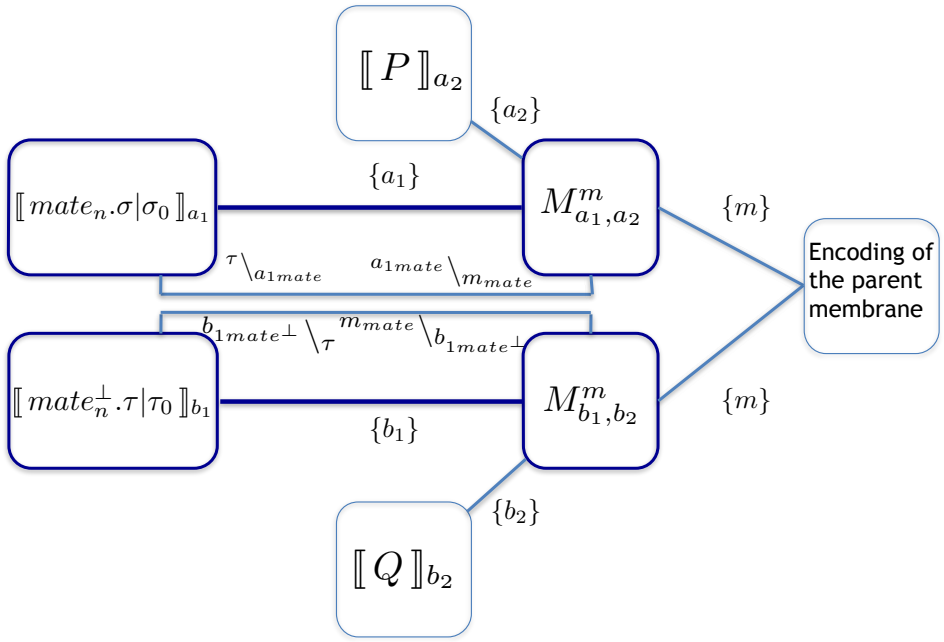

Figure 7: Illustration of the Mate Synchronisation Encoding: the Source Process.

First, we note that $[\![mate_n.\sigma|\sigma_0(\!|P|\!)_a \circ mate_n^{\perp}.\tau|\tau_0(\!|Q|\!)_b]\!]_m$ amounts to

$$(\nu\, a_1, a_2, b_1, b_2)(\; M^m_{a_1,a_2} | [\![mate_n.\sigma]\!]_{a_1} | [\![\sigma_0]\!]_{a_1} | [\![P]\!]_{a_2} \mid$$

$$M^m_{b_1,b_2} | [\![mate_n^{\perp}.\tau]\!]_{b_1} | [\![\tau_0]\!]_{b_1} | [\![Q]\!]_{b_2} \;)$$

- The $mate_n.\sigma$ party: the process $[\![mate_n.\sigma]\!]_{a_1}$ provides the link prefix ${}^{\tau}\backslash_{a_1 mate}\langle n, \underline{x_1}, \underline{x_2}\rangle$ that requires the synchronisation with the corresponding link, offered by the translation $M^m_{a_1,a_2}$ of *its* membrane. Here the message n is sent, and the names x_1, x_2 are received (but they are irrelevant).

- The party $M^m_{a_1,a_2}$ for the membrane a: the translation of the membrane a holding the $mate_n$ operation is:

$$\mathsf{rec}X.\,(\nu\, c_1, c_2)^{a_1 mate}\backslash_{m_{mate}}\langle \underline{name}, c_1, c_2\rangle.(Fwd^{c_1}_{a_1} \mid Fwd^{c_2}_{a_2} \mid X[c_i/a_i]) + \ldots$$

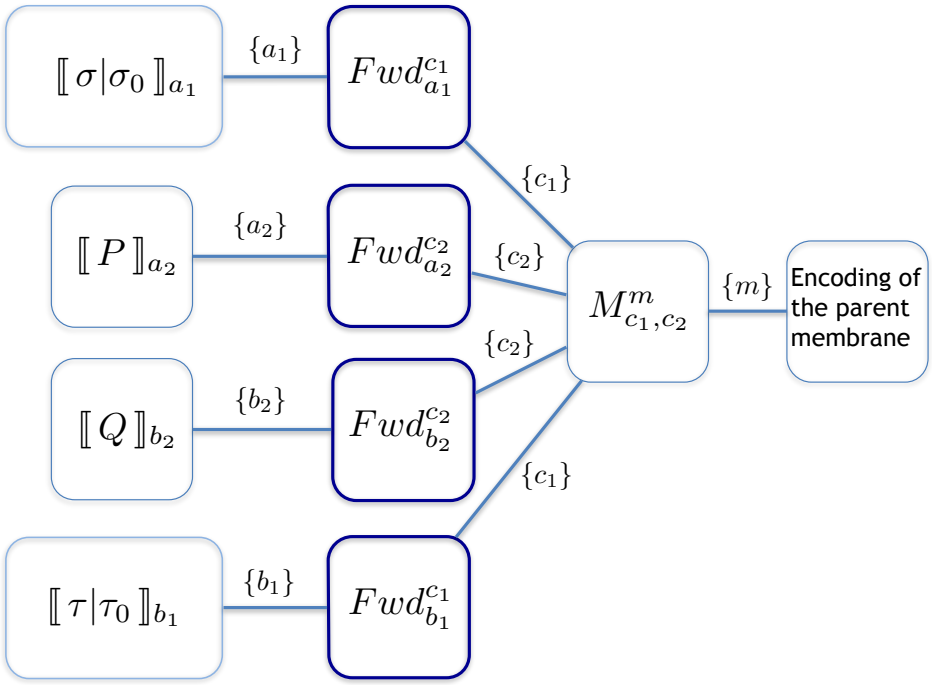

Figure 8: Illustration of the Mate Synchronisation Encoding: the Target Process.

where $i \in [1,2]$ and c_1, c_2 are two freshly created locations to be communicated to all the other parties, $X[c_1/a_1, c_2/a_2]$ can be just read as $M^m_{c_1,c_2}$ (i.e. it represents the new membrane c), the link ${}^{a_{1mate}}\backslash_{m_{mate}}$ is needed for the synchronisation with the $mate_n.\sigma$ party, and two new forwarders, $Fwd^{c_1}_{a_1} \,|\, Fwd^{c_2}_{a_2}$ are created for redirecting the communications with components that are still located at the updated locations a_1 and a_2 (because such components do not actively participate to the interaction). Note that the variable *name* is used to receive the parameter n of the (Mate) reduction, even if it is not used in the continuation.

- The party $M^m_{b_1,b_2}$ for the membrane b: the translation of the membrane b holding the $mate^{\perp}_n$ operation is even simpler:

$$\mathsf{rec}X.\ \cdots + {}^{m_{mate}}\backslash_{b_{1mate^{\perp}}} \langle \underline{name}, \underline{x_1}, \underline{x_2} \rangle.(Fwd^{x_1}_{b_1} \,|\, Fwd^{x_2}_{b_2}) + \ldots$$

where the link ${}^{m_{mate}}\backslash_{b_{1mate^\perp}}$ is for the synchronisation with the translation of $mate_n^\perp.\tau$ party, and two new forwarders $Fwd_{b_1}^{x_1} \mid Fwd_{b_2}^{x_2}$ are created to redirect the synchronisations on the old locations b_1 and b_2 to the received locations x_1 and x_2 (they will correspond to the freshly created locations c_1, c_2 extruded by the party M_{a_1,a_2}^m).

- The $mate_n^\perp.\tau$ party: the translation of $mate_n^\perp.\tau$ is similar to that of $mate_n.\sigma$. It provides the prefix ${}^{b_{1mate^\perp}}\backslash_\tau\langle n, \underline{x_1}, \underline{x_2}\rangle$, where $b_{1mate^\perp}$ is for the synchronisation with the encoding of *its* membrane, the message n is sent for the matching with the one sent by the $mate_n.\sigma$ party, and the received names x_1, x_2 are still irrelevant.

Putting all pieces together, we can observe that the four-party synchronisation forms the following link chain, once reached the agreement on the associated tuple $\langle n, \overline{c_1}, \overline{c_2}\rangle$

$$\tau\backslash^{a_{1mate}}_{a_{1mate}}\backslash^{m_{mate}}_{m_{mate}}\backslash^{b_{1mate^\perp}}_{b_{1mate^\perp}}\backslash_\tau$$

where the freshly created names c_1, c_2 are extruded to the other parties. We can then observe the correspondence of the resulting process with the encoding of the target system $[\sigma|\sigma_0]_a|[\tau|\tau_0]_b(\!([P]_a \circ [Q]_b)\!)_c$, illustrated in Figure 8, by noting that $[\![\, [\sigma|\sigma_0]_a|[\tau|\tau_0]_b(\!([P]_a \circ [Q]_b)\!)_c \,]\!]_m$ amounts to

$$(\nu\, a_1, a_2, b_1, b_2, c_1, c_2)(\ M_{c_1,c_2}^m \mid [\![\, \sigma|\sigma_0 \,]\!]_{a_1}|[\![\, P \,]\!]_{a_2}|Fwd_{a_1}^{c_1}|Fwd_{a_2}^{c_2} \mid$$

$$[\![\, \tau|\tau_0 \,]\!]_{b_1}|[\![\, Q \,]\!]_{b_2}|Fwd_{b_1}^{c_1}|Fwd_{b_2}^{c_2}\ )$$

As, in general, we would need to consider the presence of contexts $\mathbb{C}$ and $\mathbb{S}$ (i.e. of brackets in the syntax), the four-party synchronisation can become a larger multiparty synchronisation, where an unbounded number of forwarders is involved. The same holds for the cases we discuss next.

Also, the translation of the bud_n operation involves four entities. In this case no forwarder will be introduced. The Brane code we refer to is:

$$bud_n^\perp(\rho).\tau|\tau_0(\!(bud_n.\sigma|\sigma_0(\!(P)\!)_a \circ Q)\!)_b \to \rho(\!(\sigma|\sigma_0(\!(P)\!)_a)\!)_c \circ \tau|\tau_0(\!(Q)\!)_b$$

where we still distinguish the two involved membranes a and b, holding the two co-actions bud_n and $bud_n^\perp(\rho)$, respectively.

- The $bud_n.\sigma$ party is translated as ${}^{\tau}\backslash_{a_{1bud}}\langle n, \underline{x_1}, \underline{x_2}\rangle.[\![\,\sigma\,]\!]_{a_1}$. The link ${}^{\tau}\backslash_{a_{1bud}}$ is for the synchronisation with the translation $M^{b_2}_{a_1,a_2}$ of *its* membrane, the parameter n plays the same role as before, and the received locations x_1, x_2 are irrelevant.

- The party $M^{b_2}_{a_1,a_2}$ for representing the membrane a is:

$$\mathsf{rec}X.\ \cdots + {}^{a_{1bud}}\backslash_{b_{2bud}}\langle \underline{name}, \underline{x_1}, \underline{x_2}\rangle.(X[x_2/b_2]) + \ldots$$

  where $X[x_2/b_2]$ can be just read as $M^{x_2}_{a_1,a_2}$ (i.e. it represents the membrane a after the relocation in the new membrane c), and the link ${}^{a_{1bud}}\backslash_{b_{2bud}}$ is needed for the synchronisation with the $bud_n.\sigma$ party. The variable *name* is used to receive the parameter n of the (Bud) reduction, even if it is not used in the continuation. The variable x_1 is irrelevant.

- The party $M^{m}_{b_1,b_2}$ for representing the membrane b is:

$$\mathsf{rec}X.\ \cdots + (\nu\, c_1, c_2)^{b_{2bud}}\backslash_{b_{1bud^{\perp}}}\langle \underline{name}, c_1, c_2\rangle.(X[c_1/b_1, c_2/b_2]\,|\,X) + \ldots$$

  where two fresh locations c_1, c_2 and a new membrane $X[c_1/b_1, c_2/b_2]$ (to be read as $M^{m}_{c_1,c_2}$) are created to represent the new membrane c, and the link ${}^{b_{2bud}}\backslash_{b_{1bud^{\perp}}}$ is for the synchronisation with the translation of $bud^{\perp}_n(\rho).\tau$ party. Note that locations c_1, c_2 are extruded in the communication, as they will be needed by the other parties to (re)position ρ and the membrane a.

- The $bud^{\perp}_n(\rho).\tau$ party is translated as ${}^{b_{1bud^{\perp}}}\backslash_{\tau}\langle n, \underline{x_1}, \underline{x_2}\rangle.([\![\,\rho\,]\!]_{x_1}|[\![\,\sigma\,]\!]_{b_1})$. The link ${}^{b_{1bud^{\perp}}}\backslash_{\tau}$ is for the synchronisation with the translation of *its* membrane $M^{m}_{b_1,b_2}$, the parameter n plays the usual role, and the received location x_1 is used to locate ρ (only the location x_2 is irrelevant). Note that x_1 will be substituted by the freshly created location c_1.

The four-party synchronisation forms the link chain below, once reached the agreement on the associated tuple $\langle n, \overline{c_1}, \overline{c_2}\rangle$

$$\begin{smallmatrix}\tau\\ a_{1bud}\end{smallmatrix}\backslash\begin{smallmatrix}b_{2bud}\\ b_{2bud}\end{smallmatrix}\backslash\begin{smallmatrix}b_{1bud^{\perp}}\\ b_{1bud^{\perp}}\end{smallmatrix}\backslash_{\tau}$$

where the freshly created names c_1, c_2 are extruded to the other parties.

Finally, we can discuss the translation of the action *drip*, that, since it involves only one membrane, is the simplest one:

$$drip(\rho).\sigma|\sigma_0(\!|P|\!)_a \to \rho(\!|\diamond|\!)_b \circ \sigma|\sigma_0(\!|P|\!)_a.$$

The $drip(\rho).\sigma$ party is translated as ${}^{\tau}\backslash_{a_{1drip}}\langle \underline{x_1}\rangle.([\![\,\rho\,]\!]_{x_1}|[\![\,\sigma\,]\!]_{a_1})$, while the party $M^m_{a_1,a_2}$ for representing the membrane a is:

$$\mathsf{rec}X.\ \cdots + (\nu\, b_1, b_2)^{a_{1drip}}\backslash_{\tau}\langle b_1\rangle.(X[b_1/a_1, b_2/a_2]\,|\,X)$$

where the fresh locations b_1, b_2 and the process $X[b_1/a_1, b_2/a_2]$ (to be read as $M^m_{b_1,b_2}$) represent the fresh membrane b to be used for locating ρ. The two-party synchronisation forms the link chain below, once reached the agreement on the associated tuple $\langle\overline{b_1}\rangle$

$${}^{\tau}\backslash^{a_{1drip}}_{a_{1drip}}\backslash_{\tau}$$

Example We will now show a fully detailed example that illustrates the encoding of a MBD process in `link`-calculus, and also two of its possible derivations. Let $P \equiv mate_n.drip(\rho)\,|\,\sigma(\!|\tau(\!|R|\!)|\!) \circ mate^{\perp}_n.\delta(\!|Q|\!)$ be a process, where two membranes merge, and then a drip reaction creates a new empty membrane.

$$\begin{array}{ll} P \to P_1 \equiv drip(\rho)\,|\,\sigma|\delta(\!|\tau(\!|R|\!) \circ Q|\!) & (mate) \\ P_1 \to P_2 \equiv \rho(\!|\,|\!) \circ \sigma|\delta(\!|\tau(\!|R|\!) \circ Q|\!) & (drip) \end{array}$$

In the following encoding, we colour the parts involved in the mate transition, and we call, for brevity, σ_1, τ_2, δ_3 the translations of the corresponding brane actions, where the index is the number of the membranes that initially include them. The encoding is $[\![\,P\,]\!]_{top} \equiv P_{\mathtt{link}}|P'_{\mathtt{link}}$, where

$$\begin{array}{rll} P_{\mathtt{link}} \equiv (\nu\, m_1, s_1)(M^{top}_{m_1,s_1}\,|\,{}^{\tau}\backslash_{m_{1mate}}\langle n, \underline{x_1}, \underline{x_2}\rangle.{}^{\tau}\backslash_{m_{1drip}}\langle \underline{y_1}\rangle.([\![\,\rho\,]\!]_{y_1}|\sigma_1) & (\mathtt{membrane}\ m_1) \\ \qquad (\nu\, m_2, s_2)(M^{s_1}_{m_2,s_2}\,|\,\tau_2\,|\,[\![\,R\,]\!]_{s_2})) & (\mathtt{membrane}\ m_2\ \mathtt{in}\ m_1) \\ P'_{\mathtt{link}} \equiv (\nu\, m_3, s_3)(M^{top}_{m_3,s_3}\,|\,{}^{m_{3mate^{\perp}}}\backslash_{\tau}\langle n, \underline{x_1}, \underline{x_2}\rangle.\delta_3\,|\,[\![\,Q\,]\!]_{s_3}) & (\mathtt{membrane}\ m_3) \end{array}$$

For simplicity, we just give a sketch of the derivation of the two operations.

$$\dfrac{P_{\mathtt{link}} \xrightarrow{{}^{\tau}\backslash^{m_{1mate}}_{m_{1mate}}\backslash^{*}_{top_{mate}}\backslash^{*}_{*}\backslash_{*}\langle n,\overline{m'},\overline{s'}\rangle} Q_{\mathtt{link}} \qquad P'_{\mathtt{link}} \xrightarrow{{}^{*}\backslash^{*}_{*}\backslash^{top_{mate}}_{*}\backslash^{m}_{m_{3mate^{\perp}}}\backslash_{\tau}\langle n,\underline{m'},\underline{s'}\rangle} Q'_{\mathtt{link}}}{P_{\mathtt{link}}|P'_{\mathtt{link}} \xrightarrow{{}^{\tau}\backslash^{\tau}_{\tau}\backslash^{top_{mate}}_{top_{mate}}\backslash^{\tau}_{\tau}\backslash_{\tau}} P^1_{\mathtt{link}}}$$

In $P^1_{\texttt{link}} \equiv (\nu\, m', s')(Q_{\texttt{link}}|Q'_{\texttt{link}})$, we colour the parts involved in the drip transition, where also the forwarder process is involved, as the membrane m_1 is no more active:

$$P^1_{\texttt{link}} \equiv [\![\, P_1 \,]\!]_{top} \equiv (\nu\, m', s')(\quad (\nu\, m_1, s_1)(Fwd^{m'}_{m_1} \mid Fwd^{s'}_{s_1} \mid M^{top}_{m',s'} \mid {}^{\tau}\backslash_{m_{1drip}}\langle \underline{y_1} \rangle.([\![\, \rho \,]\!]_{y_1}|\sigma_1) \mid$$
$$(\nu\, m_2, s_2)(M^{s_1}_{m_2,s_2} \mid \tau_2 \mid [\![\, R \,]\!]_{s_2})) \mid$$
$$(\nu\, m_3, s_3)(Fwd^{m'}_{m_3} \mid Fwd^{s'}_{s_3} \mid \delta_3 \mid [\![\, Q \,]\!]_{s_3}))$$

$$\dfrac{\dfrac{}{P^{1a}_{\texttt{link}} \xrightarrow{{}^{*}\backslash^{m_{1drip}}_{*}\backslash_{m'_{drip}}{}^{*}_{*}\backslash\langle m'' \rangle} Q^{1a}_{\texttt{link}}} \quad \dfrac{}{P^{1b}_{\texttt{link}} \xrightarrow{{}^{*}\backslash^{*}_{*}\backslash^{m'_{drip}}_{*}\backslash\tau\langle m'' \rangle} Q^{1b}_{\texttt{link}}}}{P^{1a}_{\texttt{link}}|P^{1b}_{\texttt{link}} \xrightarrow{{}^{*}\backslash^{m_{1drip}}_{*}\backslash^{m'_{drip}}_{m'_{drip}}\backslash\tau\langle m'' \rangle} Q^{1a}_{\texttt{link}}|Q^{1b}_{\texttt{link}}} \quad \dfrac{}{P^{1c}_{\texttt{link}} \xrightarrow{\tau\backslash^{*}_{m_{1drip}}\backslash^{*}_{*}\backslash\langle m'' \rangle} Q^{1c}_{\texttt{link}}}$$

$$\dfrac{P^{1a}_{\texttt{link}}|P^{1b}_{\texttt{link}}|P^{1c}_{\texttt{link}} \xrightarrow{\tau\backslash^{m_{1drip}}_{m_{1drip}}\backslash^{m'_{drip}}_{m'_{drip}}\backslash\tau} Q^{1a}_{\texttt{link}}|Q^{1b}_{\texttt{link}}|Q^{1c}_{\texttt{link}}}{\vdots}$$

$$P^1_{\texttt{link}} \xrightarrow{\tau\backslash^{\tau}_{\tau}\backslash^{m'_{drip}}_{m'_{drip}}\backslash\tau} P^2_{\texttt{link}}$$

where $P^2_{\texttt{link}} \equiv [\![\, P_2 \,]\!]_{top}$ is equivalent to

$$(\nu\, m', s') \quad ((\nu\, m'', s'')((\nu\, m_1, s_1)(Fwd^{m'}_{m_1} \mid Fwd^{s'}_{s_1} \mid M^{top}_{m',s'} \mid M^{top}_{m'',s''} \mid [\![\, \rho \,]\!]_{m''}) \mid \sigma_1) \mid$$
$$(\nu\, m_2, s_2)(M^{s_1}_{m_2,s_2} \mid \tau_2 \mid [\![\, R \,]\!]_{s_2})) \mid$$
$$(\nu\, m_3, s_3)(Fwd^{m'}_{m_3} \mid Fwd^{s'}_{s_3} \mid \delta_3 \mid [\![\, Q \,]\!]_{s_3}))$$

The result of the above transitions is that membranes $m1$ and $m3$ have been substituted by the new merged membrane m' lying on the top position, and the four processes $Fwd^{m'}_{m_1}$, $Fwd^{s'}_{s_1}$,$Fwd^{m'}_{m_3}$,$Fwd^{s'}_{s_3}$ keep track of this event. Another new membrane m'', lying on the top position, is introduced as the result of the *drip* operation.

Operational Correspondence We can state an operational correspondence result to show the correctness of the proposed encoding.

Intuitively, the idea is to define an encoding assigning to any MBD system P a corresponding `link`-calculus process $[\![\, P \,]\!]$.

- for any reduction $P \to P'$ there is a step $[\![\, P \,]\!] \to [\![\, P' \,]\!]$;
- and vice versa, for any silent step $[\![\, P \,]\!] \to Q$ there is a MBD system P' such that $Q = [\![\, P' \,]\!]$ and $P \to P'$.

Unfortunately, a direct encoding has to deal with the presence of forwarders, so that in general:

- for any reduction $P \to P'$ we can find a step $[\![P]\!] \to Q$ but Q can differ from $[\![P']\!]$ because of the presence of forwarders;
- and vice versa, for any silent step $[\![P]\!] \to Q$ we can find a MBD system P' such that $P \to P'$ but, again, Q can differ from $[\![P']\!]$ because of the presence of forwarders.

We obtain a correspondence based on the forwarders introduced in the syntax of MBD, with no effect whatsoever on the semantics and expressiveness, and that allows us to recover the stronger correspondence result sketched above, with exact matching between the MBD reductions and the silent steps of the `link`-calculus (modulo some standard structural laws imposed by the MBD structural congruence).

Proposition 1 *The encoding in Fig. 6 is well-defined, in the sense that if* $P \equiv Q$ *then* $[\![P]\!] \equiv [\![Q]\!]$.

We can filter out non solid transitions (representing partial interactions) of encoded processes by letting: $[\![P]\!] \triangleq (\nu\, t_{mate}, t_{mate^\perp}, t_{bud}, t_{bud^\perp}, t_{drip})[\![P]\!]_{\mathtt{t}}$

We say that a link chain s is *silent* if it consists of τ actions only, and write $P \xrightarrow{\tau} Q$ if $P \xrightarrow{s} Q$ for some silent s. (Note that any silent link chain is solid.)

Lemma 1 *If* $[\![P]\!] \xrightarrow{s} Q$, *then* s *is silent.*

Sketch of the proof: The proof is by cases on the type of capability simulated by the encoded process $[\![P]\!]$ ($mate_n$, bud_n, or $drip_n$). The proof of each case is, in turn, by induction on the length of the derivation of the transition.

Theorem 1 *Let* P *be a MBD system, then* $P \to P'$ *if and only if there exists* Q *such that* $[\![P]\!] \xrightarrow{s} Q$, *and* $Q \equiv [\![P']\!]$.

Sketch of the proof: The proof for the *only if* part is by induction on the length of the derivation of the reduction $P \to P'$; the *if* part can be proved by cases on the type of the capability simulated by the encoded process. Here, some explanations about the derivation of a silent move: $[\![P]\!] \xrightarrow{s} Q$ are in order. First, the silent move s must have been obtained as a chain

of link prefixes that come equipped with compatible tuples, and that are combined along restricted channels. Of course, at the left and right hand sides of this link chain, we need two link prefixes presenting a τ on its left and on its right, respectively. Then, thanks to rule (*Close*), in Table 2, we can derive a link chain s where only τ appears and the tuple of values has been discarded.

Now, we detail the construction of the link chain. At the left hand side of the link chain, a link offering a silent action τ is required. By the rules in Figure 6, the links offering an action τ on their left are: ${}^{\tau}\backslash_{a_{mate}}$, ${}^{\tau}\backslash_{a_{bud}}$, and ${}^{\tau}\backslash_{a_{drip}}$ deriving from the encoding of the prefixes $mate_n$, bud_n, $drip_n$. At the right hand side of the link chain, a silent action τ is also required. By the rules in Figure 6, the only links offering a τ action on their right are: ${}^{b_{mate^{\perp}}}\backslash_{\tau}$, ${}^{b_{bud^{\perp}}}\backslash_{\tau}$, and ${}^{a_{drip}}\backslash_{\tau}$. The first two links derive from the encoding of the prefixes $mate_n^{\perp}$, and $bud_n^{\perp}$; the last one, ${}^{a_{drip}}\backslash_{\tau}$, derives from the encoding of the membrane holding the prefix $drip_n$. Back to the encoding of the capability *mate*, two membranes are needed to pass from the channel a_{mate} to the channel m_{mate}, and from $b_{mate^{\perp}}$ to m_{mate}: on their surface, the two membranes host the $mate_n$ and the $mate_n^{\perp}$ prefixes, respectively. As fas as the encoding of the capability *bud* is concerned, two membranes are needed to pass from the channel a_{bud} to the channel m_{bud}, and from m_{bud} to $b_{bud}^{\perp}$: on their surface, the two membranes host the bud_n and the $bud_n^{\perp}$ prefixes, respectively. The case of the capability *drip* is simpler, because it does not require additional links. Forwarders can provide links to the chains, but without changing the type of the channels. Participants to the action can be arranged in parallel, according to their position in s, by using the `link`-calculus structural congruence rules. By induction on the proof of the derivation, we can rebuild the tree of the bracketed membranes involved in the reduction.

6.2 The PEP Brane Calculus

We now extend our framework, in order to include also the Phago/ Exo/Pino (PEP) version of the Brane Calculus, where the three new reactions represent the operations that model endocytosis and exocytosis. The first indicates the process of incorporating external material into a cell, by engulfing it with the cell membrane, while the second one indicates the reverse process. Endocytosys is rendered by two more basic operations: phagocytosis (*phago*), that consists in engulfing just one external membrane, and pinocytosis (*pino*), consists in engulfing zero external membranes. Exocytosis is instead denoted

$$a, b ::= \quad ... \mid phago_n \mid phago_n^{\perp}(\rho) \mid exo_n \mid exo_n^{\perp} \mid pino(\rho)$$

$$(Phago) \quad phago_n.\sigma|\sigma_0(\!|P|\!) \circ phago_n^{\perp}(\rho).\tau|\tau_0(\!|Q|\!) \rightarrow \tau|\tau_0(\!|\rho(\!|\sigma|\sigma_0(\!|P|\!)|\!) \circ Q|\!)$$

$$(Exo) \quad exo_n^{\perp}.\tau|\tau_0(\!|exo_n.\sigma|\sigma_0(\!|P|\!) \circ Q|\!) \rightarrow P \circ \sigma|\sigma_0|\tau|\tau_0(\!|Q|\!)$$

$$(Pino) \quad pino.(\rho).\sigma|\sigma_0(\!|P|\!) \rightarrow \sigma|\sigma_0(\!|\rho(\!|\diamond|\!) \circ P|\!)$$

$$\begin{array}{rcl}
[\![\, phago_n.\sigma \,]\!]_m & \triangleq & {}^{\tau}\backslash_{m_{phg}}\langle n, \underline{x_1}, \underline{x_2}\rangle.[\![\, \sigma \,]\!]_m \\
[\![\, phago_n^{\perp}(\rho).\sigma \,]\!]_m & \triangleq & {}^{m_{phg^{\perp}}}\backslash_{\tau}\langle n, \underline{x_1}, \underline{x}\rangle.([\![\, \rho \,]\!]_{x_1} \mid [\![\, \sigma \,]\!]_m) \\
[\![\, pino(\rho).\sigma \,]\!]_m & \triangleq & {}^{\tau}\backslash_{m_{pino}}\langle \underline{x_1}\rangle.([\![\, \rho \,]\!]_{x_1} \mid [\![\, \sigma \,]\!]_m) \\
[\![\, exo_n.\sigma \,]\!]_m & \triangleq & {}^{\tau}\backslash_{m_{exo}}\langle n, \underline{x_1}, \underline{x_2}\rangle.[\![\, \sigma \,]\!]_m \\
[\![\, exo_n^{\perp}.\sigma \,]\!]_m & \triangleq & {}^{m_{exo^{\perp}}}\backslash_{\tau}\langle n, \underline{x_1}, \underline{x}\rangle.[\![\, \sigma \,]\!]_m
\end{array}$$

$$\begin{array}{rcl}
M_{i,j}^{p} & \triangleq & \mathsf{rec}X.\ \cdots + {}^{i_{phg}}\backslash_{p_{phg}}\langle \underline{name}, \underline{x_1}, \underline{x_2}\rangle.X[x_2/p] \\
& & + \ (\nu m_1, m_2)^{p_{phg}}\backslash_{i_{phg^{\perp}}}\langle \underline{name}, m_1, m_2\rangle.(M_{m_1,m_2}^{j} \mid X) \\
& & + \ {}^{i_{exo}}\backslash_{p_{exo}}\langle \underline{name}, \underline{x_1}, \underline{x}\rangle.(Fwd_i^{x_1} \mid Fwd_j^{x}) \\
& & + \ {}^{j_{exo}}\backslash_{i_{exo^{\perp}}}\langle \underline{name}, i, p\rangle.X \\
& & + \ (\nu\, m_1, m_2)^{i_{pino}}\backslash_{\tau}\langle m_1\rangle.(M_{m_1,m_2}^{m}|X)
\end{array}$$

Table 4: The Phago/Exo/Pino (PEP) Syntax, Semantics and Encoding.

by *exo*. The extended syntax and semantics are in the upper part of Table 4, while the corresponding encoding is in the lower part of the table.

As usual, the membranes are encoded as separate entities, and each synchronisation involves the membrane and one or two prefixes, corresponding to the operators $pino(\rho)$, or $phago_n$ and $phago_n^{\perp}(\rho)$, or exo_n and $exo_n^{\perp}$. For the operators *pino* and *phago*, a new membrane is created and extrusion is exploited to relocate pieces of code: actually some biological elements are forced to change their location, as they are sent to the new membrane. We omit the obvious extension of the definition of forwarders and the detailed discussion of the various cases, as the technical development is analogous to the MBD case.

6.3 Comparing the MBD and PEP encodings

As discussed by Cardelli, the MBD set of membrane capabilities (mate, bud, drip) can be derived from the PEP membrane primitives (phago, exo, pino). Nevertheless, since they model particular biological phenomena, it is worth having them as primitives as well.

For instance, the mate interaction can be obtained by a suitable combination of phago and exo, as illustrated below.

$$\begin{array}{l} mate_n \triangleq phago_n.exo_n \\ mate_n^{\perp} \triangleq phago_n^{\perp}(exo_n^{\perp}.exo_{n''}).exo_{n''}^{\perp} \end{array}$$

It would be interesting to investigate the connection between the MBD encoding of membrane primitives and the similar PEP one into the link-calculus. To this aim, we try to compare the encoding of the MBD process

$$\begin{array}{lcl} P_{MBD} & \triangleq & mate_n.\sigma|\sigma_0(\!|P|\!) \circ mate_n^{\perp}.\tau|\tau_0(\!|Q|\!) \\ [\![\, P_{MBD} \,]\!]_t & = & (\nu\, n1, n2, m1, m2) \\ & & M^t_{n1,n2}|^{\tau}\backslash_{n1_{mate}}\langle n, \underline{x1}, \underline{x2}\rangle.[\![\, \sigma \,]\!]_{n1}|[\![\, \sigma_0 \,]\!]_{n1}|[\![\, P \,]\!]_{n2}| \\ & & M^t_{m1,m2}|^{m1^{\perp}_{mate}}\backslash_{\tau}\langle n, \underline{x1}, \underline{x2}\rangle.[\![\, \tau \,]\!]_{m1}|[\![\, \tau_0 \,]\!]_{m1}|[\![\, Q \,]\!]_{m2} \end{array}$$

and the encoding of its corresponding PEP version, obtained by using the above combination of phago and exo.

$$\begin{array}{lcl} P_{PEP} & \triangleq & phago_n.exo_n\sigma|\sigma_0(\!|P|\!) \circ phago_n^{\perp}(exo_n^{\perp}.exo_{n''}).exo_{n''}^{\perp}.\tau|\tau_0(\!|Q|\!) \\ [\![\, P_{PEP} \,]\!]_t & = & (\nu\, n1, n2, m1, m2) \\ & & {}^{\tau}\backslash_{n1_{phg}}\langle n, \underline{x1}, \underline{x2}\rangle.{}^{\tau}\backslash_{n1_{exo}}\langle n', \underline{x1}, \underline{x2}\rangle.[\![\, \sigma \,]\!]_{n1}|[\![\, \sigma_0 \,]\!]_{n1}|M^t_{n1,n2}|[\![\, P \,]\!]_{n2}| \\ & & {}^{m1^{\perp}_{phg}}\backslash_{\tau}\langle n, \underline{x1}, \underline{x2}\rangle.([\![\, exo_n^{\perp}.exo_{n''} \,]\!]_{x1}|^{m1^{\perp}_{exo}}\backslash_{\tau}\langle n'', \underline{x1}, \underline{x2}\rangle.[\![\, \tau \,]\!]_{m1})| \\ & & [\![\, \tau_0 \,]\!]_{m1}|M^t_{m1,m2}|[\![\, Q \,]\!]_{m2} \end{array}$$

We show the transition of $[\![\, P_{MBD} \,]\!]$ corresponding to the mate interaction, and then the three transitions of $[\![\, P_{PEP} \,]\!]$ corresponding to the three interactions needed for simulating the mate (one phago, and two exo). Finally we compare the two results. To help the intuition, we leave the channel names in the link chains and the tuples of values in the transition labels,

although the semantics of the language would remove them.

$$
\begin{array}{lcl}
[\![P_{MBD}]\!]_t & = & (\nu\, n1, n2, m1, m2) \\
 & & M^t_{n1,n2} |^{\tau} \backslash_{n1_{mate}} \langle n, \underline{x1}, \underline{x2} \rangle . [\![\sigma]\!]_{n1} | [\![\sigma_0]\!]_{n1} | [\![P]\!]_{n2} | \\
 & & M^t_{m1,m2} |^{m1^{\perp}_{mate}} \backslash_{\tau} \langle n, \underline{x1}, \underline{x2} \rangle . [\![\tau]\!]_{m1} | [\![\tau_0]\!]_{m1} | [\![Q]\!]_{m2} \\
 & & \\
mate & \downarrow & {}^{\tau} \backslash^{n1_{mate}}_{n1_{mate}} \backslash^{t_{mate}}_{t_{mate}} \backslash^{m1^{\perp}_{mate}}_{m1_{mate}} \backslash_{\tau} \langle n, \overline{n1}, \overline{n2} \rangle \\
 & & \\
 & & (\nu\, n1, n2, m1, m2, new1, new2) \\
 & & [\![\sigma]\!]_{n1} | [\![\sigma_0]\!]_{n1} | Fwd^{new1}_{n1} | Fwd^{new2}_{n2} | M^t_{new1,new2} | [\![P]\!]_{n2} \\
 & & [\![\tau]\!]_{m1} | [\![\tau_0]\!]_{m1} | Fwd^{new1}_{m1} | Fwd^{new2}_{m2} | [\![Q]\!]_{m2} \\
 & & \\
[\![P_{PEP}]\!]_t & = & (\nu\, n1, n2, m1, m2) \\
 & & {}^{\tau} \backslash_{n1_{phg}} \langle n, \underline{x1}, \underline{x2} \rangle . {}^{\tau} \backslash_{n1_{exo}} \langle n', \underline{x1}, \underline{x2} \rangle . [\![\sigma]\!]_{n1} | [\![\sigma_0]\!]_{n1} | M^t_{n1,n2} | [\![P]\!]_{n2} | \\
 & & {}^{m1^{\perp}_{phg}} \backslash_{\tau} \langle n, \underline{x1}, \underline{x2} \rangle . ([\![exo^{\perp}_{n} . exo_{n''}]\!]_{x1} |^{m1^{\perp}_{exo}} \backslash_{\tau} \langle n^{''}, \underline{x1}, \underline{x2} \rangle . [\![\tau]\!]_{m1}) | \\
 & & [\![\tau_0]\!]_{m1} | M^t_{m1,m2} | [\![Q]\!]_{m2} \\
 & & \\
phago & \downarrow & {}^{\tau} \backslash^{n1_{phg}}_{n1_{phg}} \backslash^{t_{phg}}_{t_{phg}} \backslash^{m1^{\perp}_{phg}}_{m1_{phg}} \backslash_{\tau} \langle n, \overline{new1}, \overline{new2} \rangle \\
 & & \\
 & & (\nu\, n1, n2, m1, m2, new1, new2) \\
 & & {}^{\tau} \backslash_{n1_{exo}} \langle n', \underline{x1}, \underline{x2} \rangle . [\![\sigma]\!]_{n1} | [\![\sigma_0]\!]_{n1} | M^{new2}_{n1,n2} | [\![P]\!]_{n2} | \\
 & & {}^{new1^{\perp}_{exo}} \backslash_{\tau} \langle n', \underline{x1}, \underline{x2} \rangle . {}^{\tau} \backslash_{new1_{exo}} \langle n^{''}, \underline{x1}, \underline{x2} \rangle | [\![\tau]\!]_{m1}) | \\
 & & [\![\tau_0]\!]_{m1} | M^{m2}_{new1,new2} | M^t_{m1,m2} | [\![Q]\!]_{m2} \\
exo\ n' & \downarrow & {}^{\tau} \backslash^{n1_{exo}}_{n1_{exo}} \backslash^{new2_{exo}}_{new2_{exo}} \backslash^{new1^{\perp}_{exo}}_{new1^{\perp}_{exo}} \backslash_{\tau} \langle n', \overline{new1}, \overline{m2} \rangle \\
 & & \\
 & & (\nu\, n1, n2, m1, m2, new1, new2) \\
 & & [\![\sigma]\!]_{n1} | [\![\sigma_0]\!]_{n1} | Fwd^{new1}_{n1} | Fwd^{m2}_{n2} | [\![P]\!]_{n2} | \\
 & & {}^{\tau} \backslash_{new1_{exo}} \langle n^{''}, \underline{x1}, \underline{x2} \rangle | [\![\tau]\!]_{m1} | \\
 & & [\![\tau_0]\!]_{m1} | M^{m2}_{new1,new2} | M^t_{m1,m2} | [\![Q]\!]_{m2} \\
 & & \\
exo\ n^{''} & \downarrow & {}^{\tau} \backslash^{new1_{exo}}_{new1_{exo}} \backslash^{m2_{exo}}_{m2_{exo}} \backslash^{m1^{\perp}_{exo}}_{m1^{\perp}_{exo}} \backslash_{\tau} \langle n^{''}, \overline{m1}, \bar{t} \rangle \\
 & & \\
 & & (\nu\, n1, n2, m1, m2, new1, new2) \\
 & & [\![\sigma]\!]_{n1} | [\![\sigma_0]\!]_{n1} | Fwd^{new1}_{n1} | Fwd^{m2}_{n2} | [\![P]\!]_{n2} | [\![\tau]\!]_{m1} | [\![\tau_0]\!]_{m1} | \\
 & & Fwd^{m1}_{new1} | Fwd^{t}_{new2} | M^t_{m1,m2} | [\![Q]\!]_{m2}
\end{array}
$$

We can now compare the two final configurations (where the systems are rearranged in order to facilitate the comparison).

(MBD)
$(\nu\, n1, n2, m1, m2, new1, new2)$
$[\![\, \sigma \,]\!]_{n1} | [\![\, \sigma_0 \,]\!]_{n1} | Fwd_{n1}^{new1} | Fwd_{n2}^{new2} | M^t_{new1,new2} | [\![\, P \,]\!]_{n2} |$
$[\![\, \tau \,]\!]_{m1} | [\![\, \tau_0 \,]\!]_{m1} | Fwd_{m1}^{new1} | Fwd_{m2}^{new2} | [\![\, Q \,]\!]_{m2}$

(PEP)
$(\nu\, n1, n2, m1, m2, new1, new2)$
$[\![\, \sigma \,]\!]_{n1} | [\![\, \sigma_0 \,]\!]_{n1} | Fwd_{n1}^{new1} | Fwd_{n2}^{m2} | M^t_{m1,m2} | [\![\, P \,]\!]_{n2} |$
$[\![\, \tau \,]\!]_{m1} | [\![\, \tau_0 \,]\!]_{m1} | Fwd_{new1}^{m1} | Fwd_{new2}^{t} | [\![\, Q \,]\!]_{m2}$

As it can be observed, in both cases we obtain the same hierarchical membrane structure. More precisely, in both cases we obtain a unique membrane, called $M^t_{new1,new2}$ in the MBD case, and $M^t_{m1,m2}$ in the PEP one. The two membranes hold the same subprocesses and expose the same capabilities on their surfaces. Thus, we can say that we obtain the model of the same biological system.
From the syntactical point of view, and looking at the different forwarder processes, we can instead observe that there is a different use of the restricted names in the two cases. This phenomenon is due to our encoding technique, which uses the forwarder processes to leave track of the dissolved membranes. As a consequence, the different membrane evolution histories in the two computations are recorded in different forwarder processes. This seems reasonable, from a biological point of view, because it reflects the different traces, produced in the two cases. Technically, this amounts to a slightly different parsing of the same term, due to a different insertion of brackets.

6.4 Molecules and Molecular actions

In Brane calculi [8], Cardelli also models small molecules that can easily cross or be transported across membranes, and whose movements are completely mediated by membranes. The new set of actions models the binding and the releasing of the molecules on both sides of membrane surfaces.

The molecular extensions of the syntax and of the semantics of the Brane calculi is in Table 5. We slightly modify the original syntax by considering only the set M of molecules (ranged over by t, r), i.e. we exclude molecular complexes.

The encoding of the molecules and of the molecular reaction are in Figure 9.

Systems	P, Q	::=	$\ldots \mid t$	systems with molecules $t, r \in M$
Actions	a, b	::=	$\ldots \mid t_1(t_2) \rightrightarrows t_3(t_4)$	bind & release of molecules

B&R	$t_1 \circ t_1(t_2) \rightrightarrows t_3(t_4).\sigma \mid \sigma_0(\!\!(t_2 \circ P)\!\!) \;\rightarrow\; t_3 \circ \sigma \mid \sigma_0(\!\!(t_4 \circ P)\!\!)$

Table 5: Molecules and the Molecular Reaction.

We slightly modify the encoding function style, to keep track of both names associated to a membrane, i.e. $[\![\, P \,]\!]^{m'}_{m}$, so that we initially have $[\![\, P \,]\!] \triangleq (\nu\, t)[\![\, P \,]\!]^{t}_{t}$. The only encoding rule that results modified is the one for membranes

$$[\![\, \sigma(\!|P|\!) \,]\!]^{m'}_{m} \triangleq (\nu\, m_1, m_2)(M^{m}_{m_1,m_2} \mid [\![\, \sigma \,]\!]^{m_2}_{m_1} \mid [\![\, P \,]\!]^{m_2}_{m_2}),$$

while the other rules will keep the second membrane name as superscript for the encoding function $[\![\ldots]\!]^{m'}_{m}$, without any change.

For the encoding of the B&R operation we distinguish four cases depending on the number and the position of the molecules that will disappear. We also make use of three names: **in**, **out** and **inout** (in bold in Figure 9 to point out that are special names) to record the fact that the molecule which is inside, or outside the membrane, or both the molecules, will disappear after the B&R operation, respectively. The two prefixes offered by the membrane $M^{m}_{m_1,m_2}$ are equipped with the link ${}^{m}\backslash_{m_1}$ that will connect the outside molecule with one of the previous prefixes encoding the B&R operation. Finally, a molecule t is rendered as a process Mol^{m}_{t} located in membrane m that can recursively fire two types of prefixes: one at the beginning of the link communication, ${}^{\tau}\backslash_{m}$, and the other at the end ${}^{m}\backslash_{\tau}$; for each type of prefix, according to the encoding of the B&R operation, there are four possibilities, depending on the number and the position of the molecules that will disappear.

As an example, we show the encoding of a molecular reaction, in which both the molecules disappear: $p_1 \circ p_1(p_2) \rightrightarrows \diamond(\diamond)(\!|p_2 \circ P|\!)$. Our encoding

$$
\begin{array}{lcl}
[\![\, t \,]\!]^{m_2}_{m_1} & \triangleq & Mol^{m_2}_t \\
[\![\, t_1(t_2) \rightrightarrows t_3(t_4).\sigma \,]\!]^{m_2}_{m_1} & \triangleq & \begin{cases}
{}^{m_1}\backslash_{m_2}\langle t_1, t_2, t_3, t_4\rangle.[\![\, \sigma \,]\!]^{m_2}_{m_1} & \text{if } t_3 \neq \diamond \neq t_4 \\
{}^{m_1}\backslash_{m_2}\langle \mathbf{in}, t_1, t_2, \mathit{empty}, t_4\rangle.[\![\, \sigma \,]\!]^{m_2}_{m_1} & \text{if } t_3 = \diamond \\
{}^{m_1}\backslash_{m_2}\langle \mathbf{out}, t_1, t_2, t_3, \mathit{empty}\rangle.[\![\, \sigma \,]\!]^{m_2}_{m_1} & \text{if } t_4 = \diamond \\
{}^{m_1}\backslash_{m_2}\langle \mathbf{inout}, t_1, t_2, \mathit{empty}, \mathit{empty}\rangle.[\![\, \sigma \,]\!]^{m_2}_{m_1} & \text{if } t_3 = \diamond = t_4
\end{cases}
\end{array}
$$

$$
\begin{array}{lcl}
M^m_{m_1,m_2} & \triangleq & \mathsf{rec}X.\ \ldots\ + \\
 & & {}^{m}\backslash_{m_1}\langle \underline{t_1}, \underline{t_2}, \underline{t_3}, \underline{t_4}\rangle.M^m_{m_1,m_2} \\
 & & \ + \\
 & & {}^{m}\backslash_{m_1}\langle \underline{\mathit{type}}, \underline{t_1}, \underline{t_2}, \underline{t_3}, \underline{t_4}\rangle.M^m_{m_1,m_2}
\end{array}
$$

$$
\begin{array}{lcl}
Mol^m_t & \triangleq & \mathsf{rec}X.\,(\quad {}^{\tau}\backslash_{m}\langle t, \underline{t_2}, \underline{t_3}, \underline{t_4}\rangle.X[t_4/t] \\
 & & \qquad + \ {}^{\tau}\backslash_{m}\langle \mathbf{in}, t, \underline{t_2}, \mathit{empty}, \underline{t_4}\rangle.X[t_4/t] \\
 & & \qquad + \ {}^{\tau}\backslash_{m}\langle \mathbf{out}, t, \underline{t_2}, \underline{t_3}, \mathit{empty}\rangle.\mathbf{0} \\
 & & \qquad + \ {}^{\tau}\backslash_{m}\langle \mathbf{inout}, t, \underline{t_2}, \mathit{empty}, \mathit{empty}\rangle.\mathbf{0} \\
 & & \qquad + \ {}^{m}\backslash_{\tau}\langle \underline{t_1}, t, \underline{t_3}, \underline{t_4}\rangle.X[t_3/t] \\
 & & \qquad + \ {}^{m}\backslash_{\tau}\langle \mathbf{in}, \underline{t_1}, t, \mathit{empty}, \underline{t_4}\rangle.\mathbf{0} \\
 & & \qquad + \ {}^{m}\backslash_{\tau}\langle \mathbf{out}, \underline{t_1}, t, \underline{t_3}, \mathit{empty}\rangle.X[t_3/t] \\
 & & \qquad + \ {}^{m}\backslash_{\tau}\langle \mathbf{inout}, \underline{t_1}, t, \mathit{empty}, \mathit{empty}\rangle.\mathbf{0}
\end{array}
$$

Figure 9: The Encoding of the Molecules and of the Molecular Reaction.

generates the following `link`-calculus code:

$$(\nu\, t)(Mol^t_{p_1} \,|(\nu\, m_1, m_2)(M^t_{m_1,m_2} \,|\, {}^{m_1}\backslash_{m_2}\langle \mathbf{inout}, p_1, p_2, \mathit{empty}, \mathit{empty}\rangle \,|\, Mol^{m_2}_{p_2}))$$

$$\downarrow\ {}^{\tau}\backslash^{t}_{t}\backslash^{m_1}_{m_1}\backslash^{m_2}_{m_2}\backslash_{\tau}\langle \mathbf{inout}, p_1, p_2, \mathit{empty}, \mathit{empty}\rangle$$

$$(\nu\, t, m_2)(\mathbf{0} \,|(\nu\, m_1)(M^t_{m_1,m_2} \,|\, \mathbf{0} \,|\, \mathbf{0}))$$

where the transition label has been kept for clarity, and each *link* in the label has been offered by the subprocesses in the corresponding position, i.e. the first link, ${}^{\tau}\backslash_{t}$, has been offered by the first subprocess $Mol^t_{p_1}$, and so on.

This further encoding shows how the `link`-calculus can easily simu-

late different biological entities, considering also their locations, without introducing any new operator.

7 Our encoding at work

To show how our encoding works, we consider the example of the Semliki Forest virus, presented in [8]. Briefly: (1) a virus enters a cell via phagocytosis, then an endosome compartment merges with the virus, and the virus deposits its nucap particle (NUCAP in the code) into the cytoplasm, i.e. the liquid inside the cell, via an exocytosis (Figure 7, in [8]); (2) a series of molecular reactions reproducing the RNA virus replication follow (Section 4.6, in [8]); (3) the virus reproduction cycle ends with two further steps (Figure 8, in [8], here reported as Figure 11).

For the sake of simplicity, we only show the execution of the encoding of the part (1), which consists of membrane interactions, and the first reaction of part (2), which amounts to a molecular interaction. A nucap particle is defined as a membrane containing vRNA. The nucap surface can disassemble the nucap membrane, by pushing the molecule vRNA outside, in response to some trigger molecule found in the cytosol, (molecule dis-trg in the code).

The Brane code is as follows:

$$
\begin{array}{ll}
phago_n.exo_n(\!|NUCAP|\!) \ \circ & \texttt{(virus)} \\
phago_n^{\perp}(mate_n)(\!|mate_n^{\perp}|exo_n^{\perp}(\!|\diamond|\!) \circ \text{dis-trg}|\!) & \texttt{(cell)} \\
\texttt{where} & \\
NUCAP \triangleq dis\_trg(\text{vRNA}) \rightrightarrows \text{vRNA}(\diamond)(\!|\text{vRNA}|\!) &
\end{array}
$$

As there are no molecular interactions, we use the standard version of the encoding function that only uses one subscript name, i.e. $[\![P]\!]_t$. The corresponding `link`-calculus code is as follow:

$$
\begin{array}{ll}
\text{Phago}_{vrs1}.\text{Exo}_{vrs1}|M^t_{vrs1,vrs2}|\text{Nucap}| & \texttt{(virus)} \\
\text{Phago}^{\perp}_{cell}|M^t_{cell1,cell2}| & \texttt{(cell membrane)} \\
\text{Mate}^{\perp}_{bare}|\text{Exo}^{\perp}_{bare1}|M^{cell2}_{bare1,bare2}|Mol^{cell2}_{dis\_trg} & \texttt{(inside the cell)}
\end{array}
$$

$$
\begin{array}{lcl}
\mathrm{Phago}_{vrs1} & \triangleq & {}^{\tau}\backslash_{vrs1_{phg}}\langle n, \underline{x_1}, \underline{x_2}\rangle \\
\mathrm{Phago}^{\perp}_{cell1} & \triangleq & {}^{cell1_{phg^{\perp}}}\backslash_{\tau}\langle n, \underline{x_1}, \underline{x_2}\rangle.[\![\, mate_n \,]\!]_{x_1} \\
\mathrm{Nucap} & \triangleq & {}^{nucap1}\backslash_{nucap2}\langle \underline{m_1}, \underline{m_2}, dis\_trg, vRNA, vRNA, empty\rangle \mid \\
 & & M^{vrs2}_{nucap1,nucap2} | Mol^{nucap2}_{vRNA} \\
\mathrm{Mate}^{\perp}_{bare1} & \triangleq & {}^{bare1_{mate^{\perp}}}\backslash_{\tau}\langle n, \underline{x_1}, \underline{x_2}\rangle \\
\mathrm{Exo}_{vrs1} & \triangleq & {}^{\tau}\backslash_{vrs1_{exo}}\langle n, \underline{x_1}, \underline{x_2}\rangle \\
\mathrm{Exo}^{\perp}_{bare1} & \triangleq & {}^{bare1_{exo^{\perp}}}\backslash_{\tau}\langle n, \underline{x_1}, \underline{x_2}\rangle
\end{array}
$$

To help the intuition, we leave the channel names in the link chains in the transition labels, although the semantics of the language would remove them.

In Figure 10, we show the execution of the `link`-calculus process. Initially, there are a virus, whose membrane is encoded as $M^{t}_{vrs1,vrs2}$, and a cell, $M^{t}_{cell1,cell2}$. Both lie next to each other, in the t location. With the *phago* reaction, the virus enters the cell: the new membrane *vesicle* is created inside the cell ($M^{cell2}_{vesicle1,vesicle2}$), and the virus membrane changes its location ($M^{vesicle2}_{vrs1,vrs2}$). Then, the *bare* membrane, which models the *endosome* compartment inside the cell, merges with the *vesicle*. The effects of this operation are that (i) a new membrane *new* is created, and that (ii) the two membranes, *vesicle* and *bare*, are dissolved. As a side effect, four forwarders $Fwd^{new1}_{vesicle1}$, $Fwd^{new2}_{vesicle2}$, Fwd^{new1}_{bare1} and Fwd^{new2}_{bare2} are created to redirect any request from the old locations to the new ones. The last membrane reaction is the *exo* that allows the virus to exit the membrane *new*, and to dissolve itself, leaving proceses $Fwd^{new1}_{vrs1}|Fwd^{cell2}_{vrs2}$. Please note that the forwarder Fwd^{new1}_{vrs1} redirects the actions, previously located on the surface of the virus membrane, on the surface of the new membrane; this is a direct consequence of the behaviour of the action exocytosis, see Table 4 for the details of the Exo transition. The nucleocapsid $M^{vrs2}_{nucap1,nucap2}$ can interact with a molecule that lies in the cytoplasm ($Mol^{cell2}_{dis\_trg}$), which trigger the disassembly of the nucleocapsid. Thus the *mol* interaction can take place. The result is that the molecule $vRNA$, initially inside the nuclecapsid membrane Mol^{nucap2}_{vRNA}, is now free in the cell Mol^{cell2}_{vRNA}, and that the $Mol^{cell2}_{dis\_trg}$ has been "consumed".

Our encoding renders each biological location, i.e. each membrane, as a process that takes part in the multiparty interactions. Forwarder processes can record the history of the computation, giving the exact position of the membranes when they have been dissolved. This could be useful for an exact analysis of the dynamics of the computation. Alternatively, we can instead garbage these pieces of code, as in [7], by applying a renaming operator.

$\text{Phago}_{vrs1}.\text{Exo}_{vrs1}\vert M^{t}_{vrs1,vrs2}\vert\text{Nucap}$	(virus)
$\text{Phago}^{\perp}_{cell1}\vert M^{t}_{cell1,cell2}\vert$	(cell membrane)
$\text{Mate}^{\perp}_{bare1,bare2}\vert\text{Exo}^{\perp}_{bare1}\vert M^{cell2}_{bare1,bare2}\vert Mol^{cell2}_{dis\_trg}$	(inside the cell)
$phago\downarrow {}^{\tau}\backslash^{vrs1_{phg}}_{vrs1_{phg}}\backslash^{t_{phg}}_{t_{phg}}\backslash^{cell1_{phg^{\perp}}}_{cell1_{phg^{\perp}}}\backslash_{\tau}$	
$M^{t}_{cell1,cell2}\vert$	(cell membrane)
$(\nu\, vesicle1, vesicle2)({}^{\tau}\backslash_{vesicle1_{mate}\langle n,\underline{x_1},\underline{x_2}\rangle}\vert M^{cell2}_{vesicle1,vesicle2}\vert$	(vesicle inside the cell)
$\text{Exo}_{vrs1}\vert M^{vesicle2}_{vrs1,vrs2}\vert\text{Nucap})\vert$	(virus inside the vesicle)
$\text{Mate}^{\perp}_{bare1}\vert\text{Exo}^{\perp}_{bare1}\vert M^{cell2}_{bare1,bare2}\vert Mol^{cell2}_{dis\_trg}$	(inside the cell)
$mate\downarrow {}^{\tau}\backslash^{vesicle1_{mate}}_{vesicle1_{mate}}\backslash^{cell2_{mate}}_{cell2_{mate}}\backslash^{bare1^{\perp}_{mate}}_{bare1^{\perp}_{mate}}\backslash_{\tau}$	
$M^{t}_{cell1,cell2}\vert(\nu\, vesicle1, vesicle2, new1, new2)($	(cell membrane)
$Fwd^{new1}_{vesicle1}\vert Fwd^{new2}_{vesicle2}\vert Fwd^{new1}_{bare1}\vert Fwd^{new2}_{bare2}\vert$	(dissolved membranes)
$\text{Exo}_{vrs1}\vert M^{vesicle2}_{vrs1,vrs2}\vert\text{Nucap}\vert$	(virus inside the new membrane)
$\text{Exo}^{\perp}_{bare1}\vert M^{cell2}_{new1,new2}\vert Mol^{cell}_{dis\_trg}$	(new membrane and)
	(dis-trg molecule)
$exo\downarrow {}^{\tau}\backslash^{vrs1_{exo}}_{vrs1_{exo}}\backslash^{vesicle2_{exo}}_{vesicle2_{exo}}\backslash^{new2_{exo}}_{new2_{exo}}\backslash^{new1^{\perp}_{exo}}_{new1^{\perp}_{exo}}\backslash^{bare1^{\perp}_{exo}}_{bare1^{\perp}_{exo}}\backslash_{s\tau}$	
$M^{t}_{cell1,cell2}\vert(\nu\, vesicle1, vesicle2, new1, new2)($	(cell membrane)
$Fwd^{new1}_{vrs1}\vert Fwd^{cell2}_{vrs2}\vert Fwd^{new1}_{vesicle1}\vert Fwd^{new2}_{vesicle2}\vert Fwd^{new1}_{bare1}\vert Fwd^{new2}_{bare2}\vert$	(dissolved membranes)
${}^{nucap1}\backslash_{nucap2}\langle\underline{\text{out}},\underline{\text{in}},\text{dis-trg},\text{vRNA},\text{vRNA},\text{empty}\rangle\vert M^{vrs2}_{nucap1,nucap2}\vert$	(nuclecapsid membrane)
$M^{cell2}_{new1,new2}\vert Mol^{cell2}_{dis\_trg})$	(new membrane and)
	(dis-trig molecule)
$mol\downarrow {}^{\tau}\backslash^{cell2}_{cell2}\backslash^{vrs2}_{vrs2}\backslash^{nucap1}_{nucap1}\backslash^{nucap2}_{nucap2}\backslash_{\tau}$	
$M^{t}_{cell1,cell2}\vert(\nu\, vesicle1, vesicle2, new1, new2)($	(cell membrane)
$Fwd^{cell1}_{vrs1}\vert Fwd^{cell2}_{vrs2}\vert Fwd^{new1}_{vesicle1}\vert Fwd^{new2}_{vesicle2}\vert Fwd^{new1}_{bare1}\vert Fwd^{new2}_{bare2}\vert$	(dissolved membranes)
$M^{vrs2}_{nucap1,nucap2}\vert$	(nuclecapsid membrane)
$M^{cell_2}_{new1,new2}\vert Mol^{cell_2}_{vRNA})$	(new membrane and)
	(vRNA molecule)

Figure 10: Execution of the Encoding of the Brane Phago, Mate, Exo and of a Molecular Interaction.

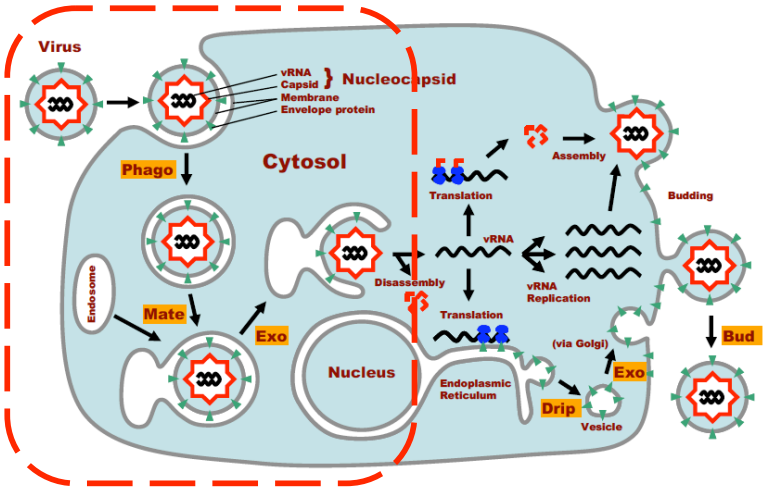

Figure 11: Viral Infection (highlighted part) and Reproduction. [Adapted from [8]].

8 Conclusion

We presented the application of the `link`-calculus to Systems Biology, showing that it is suitable for representing biological interactions. In this field, interactions vary on the number of elements, on their locations, and on their shapes. Here, we have shown that the `link`-calculus can easily take into account interactions with a different number of participants, and that can easily manage membrane compartments. In particular, we can encode the MBD version of Brane calculi with the `link`-calculus, and we can also offer similar encodings for further versions of the calculus, based on different sets of primitives. Our flat calculus puts its set of low-level primitives at our disposal to simply implement the various more abstract primitives of Brane. We have an operational correspondence result that shows the correctness of the proposed encoding. In [8], Cardelli shortly addresses the problem of encoding Brane Calculi in BioAmbients [25], by observing some critical

aspects such as atomicity of actions and the difficulty in managing the interactions through the ambient structure. The encodings presented here make us confident that such limitations do not apply to a flat and multiparty calculus like the `link`-calculus.

Furthermore, we can directly model other biological interactions, like those involved in the Receptor-mediated Endocytosis, without passing from another process calculus. The simplicity and expressivity of our encoding of this mechanism encourages us to think that `link`-calculus can be adopted to provide straight models of biological processes. In future work, we would like to enrich the `link`-calculus in order to take into account also the shape and geometry of the biological elements, that influence their interactions.

It would be interesting to further investigate and test the encoding capability of `link`-calculus, by considering other compartment-based calculi, such as BioAmbients [25], the calculus for proteins and cells [21], introduced as an extension of κ-calculus [15], Bio-PEPA [14], the Calculus of Looping Sequences [2], Beta Binders [23], the Shape calculus [3]; and Membrane Systems [10, 11].

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments.

This research has been supported by the MIUR project PRIN CINA Prot. 2010LHT4KM.

References

[1] B. Alberts, A. Johnson, P. Water, J. Lewis, M. Raff, K. Roberts, and N. Orme. *Molecular Biology of the Cell*. Garland, 2007.

[2] R. Barbuti, G. Caravagna, A. Maggiolo-Schettini, P. Milazzo, and G. Pardini. The calculus of looping sequences. In *Formal Methods for Computational Systems Biology, 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM 2008)*, volume 5016 of *Lecture Notes in Computer Science*, pages 387–423, 2008. doi:10.1007/978-3-540-68894-5_11.

[3] E. Bartocci, F. Corradini, M.R. Di Berardini, E. Merelli, and L. Tesei. Shape calculus. a spatial mobile calculus for 3d shapes. *Scientific Annals of Computer Science*, 20, 2010.

[4] C. Bodei, L. Brodo, and R. Bruni. Open multiparty interaction. In *Recent Trends in Algebraic Development Techniques, 21st International Workshop (WADT 2012)*, volume 7841 of *Lecture Notes in Computer Science*, pages 1–23, 2013. doi:10.1007/978-3-642-37635-1_1.

[5] F. Bonchi, F. Gadducci, and G. V. Monreale. Labelled transitions for mobile ambients (as synthesized via a graphical encoding). *Electronic Notes in Theoretical Computer Science*, 242(1):73–98, 2009. doi:10.1016/j.entcs.2009.06.014.

[6] F. Bonchi, F. Gadducci, and G. V. Monreale. Reactive systems, barbed semantics, and the mobile ambients. In *Foundations of Software Science and Computational Structures (FoSSaCS 2009)*, volume 5504 of *Lecture Notes in Computer Science*, pages 272–287, 2009. doi:10.1007/978-3-642-00596-1_20.

[7] L. Brodo. On the expressiveness of the π-calculus and the mobile ambients. In M. Johnson and D. Pavlovic, editors, *Algebraic Methodology and Software Technology - 13th International Conference (AMAST 2010)*, volume 6486 of *Lecture Notes in Computer Science*, pages 44–59. Springer, 2010. doi:10.1007/978-3-642-17796-5_3.

[8] L. Cardelli. Brane calculi - interactions of biological membranes. In *Computational Methods in Systems Biology, 7th International Conference (CMSB 2004)*, volume 3082 of *Lecture Notes in Computer Science*, pages 257–280. Springer, 2005. doi:10.1007/978-3-540-25974-9_24.

[9] L. Cardelli and A. D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000. doi:10.1016/S0304-3975(99)00231-5.

[10] G. Ciobanu, R. Desai, and A. Kumar. Membrane systems and distributed computing. In G. Păun, G. Rozenberg, A. Salomaa, and C. Zandron, editors, *Membrane Computing, International Workshop (WMC-CdeA 2002)*, volume 2597 of *Lecture Notes in Computer Science*, pages 187–202. Springer, 2003. doi:10.1007/3-540-36490-0_12.

[11] G. Ciobanu, M.J. Pérez-Jiménez, and G. Păun, editors. *Applications of Membrane Computing*. Natural Computing Series. Springer, 2006.

[12] G. Ciobanu and V. A. Zakharov. Encoding mobile ambients into the π -calculus. In I. Virbitskaite and A. Voronkov, editors, *Ershov Memorial*

Conference, 2006, volume 4378 of *Lecture Notes in Computer Science*, pages 148–165. Springer, 2007. doi:10.1007/978-3-540-70881-0_15.

[13] F. Ciocchetta. The BlenX language with biological transactions. In C. Priami, editor, *Transactions on Computational Systems Biology IX*, volume 5121 of *Lecture Notes in Computer Science*, pages 114–152. Springer, 2008. doi:10.1007/978-3-540-88765-2_4.

[14] F. Ciocchetta and J. Hillston. Bio-PEPA: An extension of the process algebra PEPA for biochemical networks. *Electronic Notes in Theoretical Computer Science*, 194(3):103–117, 2008. doi:10.1016/j.entcs.2007.12.008.

[15] V. Danos and C. Laneve. Graphs for core molecular biology. In *Computational Methods in Systems Biology, 7th International Conference (CMS 2003)*, volume 2602 of *Lecture Notes in Computer Science*, pages 34–46, 2003. doi:10.1007/3-540-36481-1_4.

[16] L. Dematté, C. Priami, and A. Romanel. The BlenX language: a tutorial. In *Formal Methods for Computational Systems Biology, School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM 2008)*, volume 5016 of *Lecture Notes in Computer Science*, pages 313–365. Springer, 2008. doi:10.1007/978-3-540-68894-5_9.

[17] G. L. Ferrari, U. Montanari, and E. Tuosto. A LTS semantics of ambients via graph synchronization with mobility. In *Theoretical Computer Science, 7th Italian Conference (ICTCS 2001)*, volume 2202 of *Lecture Notes in Computer Science*, pages 1–16, 2001. doi:10.1007/3-540-45446-2_1.

[18] P. Gardner, C. Laneve, and L. Wischik. Linear forwarders. *Information and Computation*, 205(10):1526–1550, 2007. doi:10.1016/j.ic.2007.01.006.

[19] J. Heath, M.Z. Kwiatkowska, G. Norman, and D. Parker O. Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 391(3):239–257, 2008. doi:10.1016/j.tcs.2007.11.013.

[20] M. Kwiatkowska, G. Norman, and D. Parker. Using probabilistic model checking in systems biology. *ACM SIGMETRICS Performance Evaluation Review*, 35(4):14–21, 2008. doi:10.1145/1364644.1364651.

[21] C. Laneve and F. Tarissan. A simple calculus for proteins and cells. *Electronic Notes in Theoretical Computer Science*, 171(2):139–154, 2007. doi:10.1016/j.entcs.2007.05.013.

[22] M. Merro and F. Zappa Nardelli. Behavioral theory for mobile ambients. *Journal of the ACM*, 52(6):961–1023, 2005. doi:10.1145/1101821.1101825.

[23] C. Priami and P. Quaglia. Beta binders for biological interactions. In *Computational Methods in Systems Biology, 7th International Conference (CMSB 2004)*, volume 3082 of *Lecture Notes in Computer Science*, pages 20–33, 2005. doi:10.1007/978-3-540-25974-9_3.

[24] J. Rathke and P. Sobociński. Deriving structural labelled transitions for mobile ambients. *Information and Computation*, 208(10):1221–1242, 2010. doi:10.1007/978-3-540-85361-9_36.

[25] A. Regev, E.M. Panina, W. Silverman, L. Cardelli, and E.Y. Shapiro. BioAmbients: An abstraction for biological compartments. *Theoretical Computer Science*, 325(1):141–167, 2004. doi:10.1016/j.tcs.2004.03.061.

[26] A. Regev, W. Silverman, and E.Y. Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In *Pacific Symposium of Biocomputing 2001*, volume 6, pages 459–470, 2001.

[27] C. Versari. A core calculus for a comparative analysis of bio-inspired calculi. In *Programming Languages and Systems, 16th European Symposium on Programming (ESOP 2007)*, volume 4421 of *Lecture Notes in Computer Science*, pages 411–425, 2007. doi:10.1007/978-3-540-71316-6_28.