Rewrite Rules and Strategies for Molecular Graphs

Oana Andrei, Hélène Kirchner

Rewrite Rules and Strategies for Molecular Graphs

Oana Andrei[1] and Hélène Kirchner[2]

[1] INRIA - LORIA
[2] CNRS - LORIA
Campus scientifique BP 239
F-54506 Vandoeuvre-lès-Nancy Cedex, France
Firstname.Lastname@loria.fr

Abstract. In this paper, we present a rewriting framework for modeling molecular complexes, biochemical reaction rules and generation of biochemical networks based on the representation of molecular complexes as a particular type of multigraphs with ports called molecular graphs. The advantage of this approach is to obtain for free a rewriting calculus which allows defining at the same level transformation rules and strategies for modeling rule selection and application, in order to prototype network generation.

1 Introduction

Many formal notations have already been used for modeling biological systems, and, in particular, protein-protein interactions concerned with the connectivity inside molecular complexes. These models can be classified as qualitative or quantitative, and the employed formal notations range from process calculi with different extensions or variations [1–4], to term-rewriting [5] and graph rewriting [6].

In the biochemical model we consider, the behaviour of a protein is given by its functional domains that determine which other protein it can bind to or interact with. These domains are usually abstracted as *sites* that can be bound or free, visible or hidden. A protein is characterized by the collection of interaction sites on its surface, also called *interface*. Proteins can bind to each other forming molecular complexes. Membranes can also form complexes, called tissues, due to the binding proteins on their surfaces. The structure of a complex is naturally described as a particular class of graphs, a mathematical structure that is easy to understand and use by computer scientists as well as by chemists and biologists.

We propose in this paper a calculus based on multigraph rewriting and rewrite strategies for modeling biochemical systems, in particular the structure and interactions of protein complexes. We use an extended version of multigraphs with ports [7] for describing the structure of complexes as molecular graphs and multigraph rewriting for modeling the interactions between them (Section 3). We encode the molecular graphs as terms, the reaction patterns as rewrite rules, and

the transformation on molecular graphs as a rewriting relation (Section 4). We define a rewriting calculus for molecular graphs ρ_{bio}-calculus, obtained from the rewrite calculus for labeled multigraphs with ports, the ρ_{mg}-calculus, introduced in [7], by adding state information on ports and imposing some conditions on edges. Strategic rewriting allows modeling the control mechanism in biochemical systems and the generation of biochemical networks. This is illustrated in Section 5 on a fragment of the epidermal growth factor receptor (EGFR) signaling cascade.

2 Background

In this section we briefly review some basic definitions of sorted term algebra, term rewriting [8] and strategic rewriting, graph theory and labeled multigraphs with ports developed in [7].

Term Algebra. A *many-sorted signature* is a pair (S, Σ), where S is called the *sort set* and Σ is an $S^* \times S$-sorted family $\{\Sigma_{w,s} \mid w \in S^* \text{ and } s \in S\}$. For $\mathcal{X} = \{\mathcal{X}_s\}$ an S-sorted family of disjoint sets of variables, $\mathcal{T}_\Sigma(\mathcal{X})$ is the smallest set of *Σ-terms over $\mathcal{X}$* built with operators from Σ and variables from $\mathcal{X}$. If $\mathcal{X}$ is empty, then we write $\mathcal{T}_\Sigma$ for $\mathcal{T}_\Sigma(\emptyset)$ and call it the set of *ground Σ-terms*. The set of all terms of sort s is denoted by $\mathcal{T}_{\Sigma,s}(\mathcal{X})$. A *ground substitution* is a partial mapping from $\mathcal{X}$ to ground Σ-terms and it uniquely extends to a Σ-homomorphism from $\mathcal{T}_\Sigma(\mathcal{X})$ to $\mathcal{T}_\Sigma$. A finite ground substitution has the form $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, with $x_i \in \mathcal{X}$ and $t \in \mathcal{T}_\Sigma$, for all $i = 1..n$. Considering S as a partially ordered set leads to quite similar definitions for order-sorted signature, terms and substitutions [9].

Term Rewriting. A set $\mathcal{R}$ of *rewrite rules* is a set of ordered pairs of terms of $\mathcal{T}_\Sigma(\mathcal{X})$, denoted $\ell \to r$, such that l and r belong to the same sort, $\ell \notin \mathcal{X}$ and $Var(r) \subseteq Var(\ell)$. In this paper, we only consider finite sets of rewrite rules. The *rewriting relation* induced by $\mathcal{R}$ is denoted by $\to_\mathcal{R}$ ($\to$ if there is no ambiguity about $\mathcal{R}$), and defined by $s \to t$ iff there exists a substitution σ and a position p in s such that $s = s[\sigma\ell]_p$ for some rule $\ell \to r$ of $\mathcal{R}$, and $t = s[\sigma r]_p$. This is written $s \to_\mathcal{R}^{p,\ell\to r,\sigma} t$ where either $\mathcal{R}$, p, $\ell \to r$, or σ may be omitted. The reflexive transitive closure of the rewriting relation induced by $\mathcal{R}$ is denoted by $\overset{*}{\to}_\mathcal{R}$, and by $\overset{+}{\to}_\mathcal{R}$ its respective transitive closure. A rewriting derivation is a chain of terms $t_1 \to t_2 \to \ldots t_n$. The *source* of a derivation $t_1 \to t_2 \to \ldots$ is t_1, and when the derivation is finite, its last term is called its *target*.

A *rewrite theory* over $\mathcal{T}_\Sigma(\mathcal{X})$ is a triple $(\Sigma, E, \mathcal{R})$ where E is a finite set of (sort-preserving) equalities and $\mathcal{R}$ a finite set of rewrite rules. The relation $\to_{\mathcal{R}/E}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ is $=_E; \to_\mathcal{R}; =_E$ and it induces a relation $\to_{\mathcal{R}/E}$ on the quotient algebra $\mathcal{T}_\Sigma(\mathcal{X})/_{=_E}$ by $[t]_E \to_{\mathcal{R}/E} [t']_E$ iff $t \to_{\mathcal{R},E} t'$. The relation $\to_{\mathcal{R},E}$ (called rewriting modulo E) on $\mathcal{T}_\Sigma(\mathcal{X})$ is defined by $s \to_{\mathcal{R},E} t$ iff there exists a substitution σ and a position p in s such that $s|_p =_E \sigma l$ for some rule $l \to r$ of $\mathcal{R}$, and $t = s[\sigma r]_p$. In this paper, we will consider theories with E defined by specific axioms, namely associativity (A), associativity and commutativity (AC) and unit element (U) of some function symbols.

$$
\begin{array}{l}
[\texttt{id}](t) = \{t\} \\
[\texttt{fail}](t) = \emptyset \\
[l \to r](t) = \{s \mid t \to^{l \to r} s\} \\
[\texttt{onestep}(\zeta_1, \ldots, \zeta_n)](t) = \bigcup_i [\zeta_i](t) \\
[\texttt{seq}(\zeta_1, \zeta_2)](t) = [\zeta_2]([\zeta_1](t)) \\
[\texttt{choice}(\zeta_1, \zeta_2)](t) = \begin{cases} [\zeta_1](t), \text{ if } [\zeta_1](t) \neq \emptyset \\ [\zeta_2](t), \text{ if } [\zeta_1](t) = \emptyset \end{cases}
\end{array}
$$

Fig. 1. Strategy constructors

Strategic Rewriting. The notions of strategy and strategic rewriting have been introduced in order to control rewriting derivations. The notion of strategy can be defined in a general way: a *rewrite strategy* ζ for the rewrite system $\mathcal{R}$ is a subset of the set of all derivations of $\mathcal{R}$. A derivation of the strategy is called a *strategic rewriting derivation*. The *application of a strategy* ζ on a term t is denoted $[\zeta](t)$ and defined as the set of all targets t' of the derivations of source t in ζ. When no derivation in ζ has t as source, we say that the strategy application on t fails. The result of the application of a failing strategy on a term t is the empty set. We extend the application of a strategy to a set of terms as the union of the application of the strategy on each element of the set.

A strategy can be described by enumerating all its elements or more suitably by a *strategy language*. From elementary strategy expressions directly issued from a rewrite system $\mathcal{R}$, more elaborated strategy expressions are built like in ELAN [10], Stratego [11], TOM [12] or, more recently, Maude [13]. The semantics of such a language is naturally described in the rewriting calculus [14]. We describe below some constructs of the strategy language of interest in this paper and available in TOM[1].

Given a rewrite system $\mathcal{R}$ over $\mathcal{T}_\Sigma(\mathcal{X})$, a strategy expression is either a rewrite rule in $\mathcal{R}$ or an expression described below. A strategy operator ζ may take other strategies $\zeta_1, \ldots, \zeta_n$ as arguments, and the result is expressed functionally by $\zeta(\zeta_1, \ldots, \zeta_n)$. We present in Figure 1 some strategy constructors used in this paper. The strategy `onestep` computes all derivations issued from the application of a nonempty set of strategies. The strategy $\texttt{seq}(\zeta_1, \zeta_2)$ first applies ζ_1 and then, if that succeeds, it applies ζ_2; it fails if either ζ_1 fails, or ζ_2 fails. $\texttt{choice}(\zeta_1, \zeta_2)$ applies the strategy ζ_1; if the application of ζ_1 fails, it applies ζ_2. Therefore $\texttt{choice}(\zeta_1, \zeta_2)$ fails if both ζ_1 and ζ_2 fail. Both strategy operators `seq` and `choice` extend naturally to be applicable to a list of strategies.

Along with the elementary rewrite rules, the strategy constructors represent the key-components necessary for defining more complex strategies. For instance,

[1] Web page `http://tom.loria.fr`

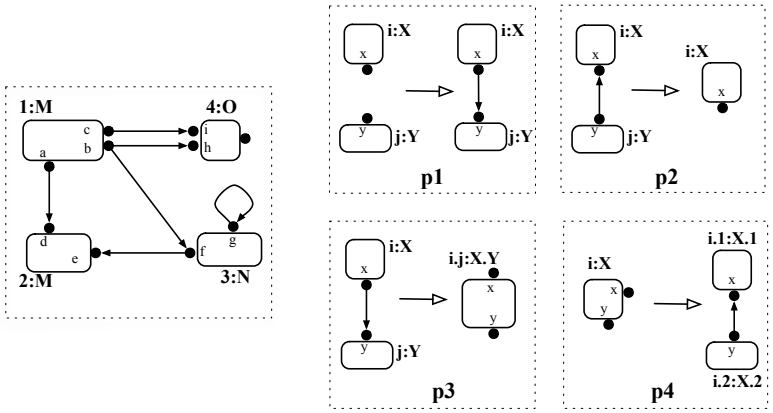


Fig. 2. A multigraph with ports (a) and some multigraph transformations (b)

`try` and `repeat` allow iterating the application of a strategy and never fail.

$$\begin{array}{lcl} \texttt{try}(\zeta) & = & \texttt{choice}(\zeta, \texttt{id}) \\ \texttt{repeat}(\zeta) & = & \texttt{try}(\texttt{seq}(\zeta, \texttt{repeat}(\zeta))) \end{array}$$

Other high-level strategies implement term traversal and normalization on terms and are well-known in the term rewriting literature (see [12] for instance).

Labeled Graphs. A *label alphabet* $\mathcal{L} = (\mathcal{L}_V, \mathcal{L}_E)$ is a pair of sets of node labels and edge labels. A (finite) *graph* over $\mathcal{L}$ is a triple $G = (V, E, l)$ where V is a set $\{v_1, \ldots, v_n\}$ of elements called *nodes*, E is a family $(e_1, \ldots, e_m)$ of elements of the Cartesian product $V \times V$ called *edges*, and $l = (l_V, l_E)$ is the *labeling function* for nodes ($l_V : V \to \mathcal{L}_V$) and edges ($l_E : E \to \mathcal{L}_E$). If G is a graph, we denote by V_G its node set, by E_G its edge set, and by l_G its labeling function. An edge of the form (v, v) is called a *loop*. For an edge (v_1, v_2), v_1 and v_2 are called *end nodes* (or *end points*) with v_1 the *source* and v_2 the *target*; moreover we say that v_1 and v_2 are *adjacent* or *neighbouring* nodes, with v_2 neighbour of v_1. An edge is *incident* to a node if the node is one of its endnodes. An edge is *multiple* if there is another edge with the same source and target; otherwise it is *simple*. A graph is *simple* if it has no multiple edges or loops, otherwise it is a *multigraph*. An *adjacency list* for a node is given by a list of pairs consisting of a neighbour and the corresponding edge label.

Labeled Multigraphs with Ports. A *labeled multigraph with ports* is obtained from a labeled multigraph by associating to each node a set of *ports* such that each edge is determined by two ports, one from each end node. A node label consists of a node identifier and a set of ports, while a simple edge label is the pair of source and target ports. Hereinafter we use multigraph instead of labeled multigraph with ports if there is no risk of confusion. We illustrate in Figure 2(a) such a multigraph.

Transforming Multigraphs with Ports. A *transformation rule on multigraphs with ports* $L \rightsquigarrow R$ consists of two multigraphs L and R called the left- and right-hand side respectively, and a correspondence between nodes of L and nodes of R, $\xi : V_L \rightharpoonup \mathcal{P}(V_R)$, which we call *node-substitution*. This correspondence is provided by some unique identifiers associated to nodes. The application of a multigraph transformation rule $L \rightsquigarrow R$ to a multigraph G produces a new multigraph G' by the following steps: first, find and remove a subgraph of G isomophic with L via a matching morphism m, resulting in a context graph G^-; then add and reconnect $m(R)$ to G^- with the help of $m(\xi)$.

We illustrate in Figure 2 (b) some simple multigraph transformations: **p1** creates an edge between two ports, **p2** removes the source node of an edge, **p3** merges two adjacent nodes, and **p4** splits a node containing (at least) two ports in two adjacent nodes by distributing a port to each node.

The Rewriting Calculus for Multigraphs with Ports. The rewriting calculus for labeled multigraphs with ports (or ρ_{mg}-calculus), fully detailed in [7], is a variation of the ρ-calculus [15] with particular features related to multigraph transformation. It provides a clear operational semantics to multigraph transformation. We benefit from its capability of encoding rewrite strategies and conditional rewrite rules. Moreover the ρ-calculus integrates λ-calculus and rewriting in a powerful higher-order pattern calculus.

3 The Biochemical Model

A *molecular complex* consists of molecules connected through bonds between sites. A molecule is characterized by a pair of name and interface (set of sites). In order to make a distinction between molecules having the same name, we assign to each molecule a *unique identifier* consisting of a sequence of integers. Each site is characterized by a name and a state: b for bound, v and h for free site that are visible and hidden respectively.

We represent a molecular complex as a connected labeled multigraph with ports where molecules are nodes, sites are ports with states, and bonds are simple edges, together with the restriction that a port can be the end node of at most one edge. We call this kind of multigraphs, not necessarily connected, **molecular graphs**. In Figure 3 we define their abstract syntax, with "," the juxtaposition

Id i	$::= Int^+$
Name m, s	$::= String$
State t	$::= \mathsf{b} \mid \mathsf{v} \mid \mathsf{h}$
Sites S	$::= s^\frown t \mid S, S \mid \epsilon$
Molecules M	$::= \langle i : m \parallel S \rangle \mid M, M \mid \epsilon$
Bonds B	$::= (i^\frown s, i^\frown s) \mid B, B \mid \epsilon$
Molecular graph G	$::= (M, B)$

Fig. 3. The syntax of molecular graphs

operator and ϵ the empty set. An unbound molecule has also a molecular graph representation as a single node with an empty list of bonds.

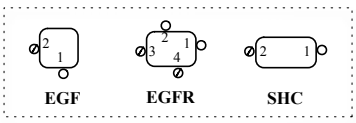
Using the same graphical representation as for multigraphs, inspired by [6, 3], we represent a molecule as an empty box having the identifier placed at the exterior and the sites as small points on the surface of the box. We represent the state of a site as follows: a filled circle for bound sites, an empty one for visible sites, and a slashed circle for hidden sites.

A biochemical system is represented as a discrete system consisting of interacting components which give rise to structural and behavioural transformation of the components and of the system as a whole. Such a system is dynamic, has an emergent behaviour, is highly concurrent and non-deterministic.

The interactions are abstracted as reaction patterns: each of the reaction patterns corresponds to a class of reactions characterized by common features. There are two types of protein interactions: (i) state changing by attaching or removing small phosphate groups at specific sites, and (ii) complexation by binding sites or decomplexation by breaking a bond between two proteins. An interaction may reveal new sites or hide some of the existing ones.

Given an initial set of biochemical species (as molecular graphs) and reactions described by a set of patterns (as molecular graph transformation rules), we automatically obtain a network by iteratively applying the reactions on the existing biochemical species until a termination condition is satisfied or until the exhaustive generation of all possible species is achieved.

Example 1. We illustrate a fragment of the EGFR signaling cascade already formalized in papers like [3, 4]. The protagonists of this model, depicted below, are: the epidermal growth factor *EGF* situated outside the cell acting as a ligand, the transmembrane protein *EGFR* with two extracellular sites and two intracellular sites as a receptor, and the adapter protein *SHC* inside the cell.



The signalling information is propagated from outside the cell to its interior following the reaction patterns depicted in Figure 4. We note that, for the molecules participating in the reaction patterns, only the relevant sites are made precise, the rest of them remain unchanged. The ligand *EGF* does not enter the cell, instead it transmits the information as a signal across the cell membrane. The function of information transfer across the membrane is performed by the receptor *EGFR*. One extracellular binding site of the receptor recognizes the signal protein in a dimeric form *EGF.EGF* produced by **(r1)** leading to the creation of a bond between the *EGF* dimer and the receptor, and to the activation by phosphorylation of a second extracellular site of the receptor **(r2)**.

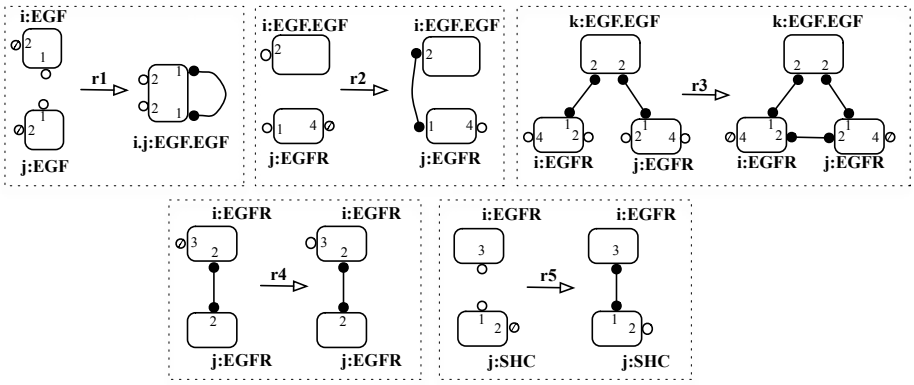


Fig. 4. The reaction patterns in the EGFR signaling cascade fragment

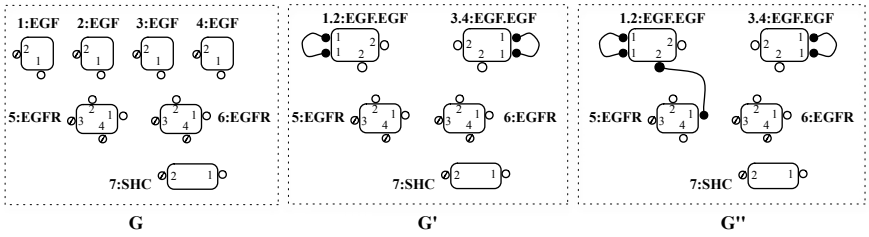


Fig. 5. G is the initial configuration; **G'** results from applying twice **r1** on **G**; **G"** results from an application of **r2** on **G'**

Two receptors activated by the same *EGF* dimer bind by **(r3)** creating an active dimer – a receptor tyrosine kinase (RTK). The RTK attaches phosphate groups to certain tyrosines, first on itself **(r4)**, and then on other proteins by activating them, as it is the case for the adapter *SHC* **(r5)**. Some of these latter proteins are also tyrosine kinases and, consequently, a cascade of phosphorylations occurs. We illustrate in Figure 5 an initial molecular graph **G** we consider for this example, and two molecular graphs **G'** and **G"** resulting from it by applying some of the reactions above.

After the initiation of the signaling process, the signaling cascade must be terminated at a certain point since cells must be able to stop responding to a signal. If not, the signaling process may lead to uncontrolled cell growth as, for instance, in cancer. This process is ended by quickly engulfing and destroying the ligand-receptor complex by receptor-mediated endocytosis.

4 Term Rewriting Semantics for Molecular Graphs Transformations

We present in this section a signature for terms encoding molecular graphs, an encoding of reaction patterns as rewriting rules, and a rewriting relation encoding the application of reactions patterns on molecular graphs.

4.1 Encoding the Molecular Graphs

We encode molecular graphs as terms using an order-sorted signature $\Sigma = (\mathcal{S}, <, \mathcal{F})$ similarly to the term rewriting encoding of multigraphs with ports [7] up to sort renaming and extension of site information with a state. We give here only the operator set:

$$
\begin{array}{rl}
\epsilon_{\mathrm{X}} & : \longrightarrow \mathrm{X}Set \\
\_,\_ & : \mathrm{X}Set \times \mathrm{X}Set \longrightarrow \mathrm{X}Set\ [ACU(\epsilon_{\mathrm{X}})] \\
\_\hat{}\_ & : SiteName \times \{\mathsf{b}, \mathsf{v}, \mathsf{h}\} \to Site \\
\langle\_ : \_ \parallel \_\rangle & : Id \times MoleculeName \times SiteSet \longrightarrow \\
 & \quad Molecule \\
(\_,\_) & : SiteName \times SiteName \longrightarrow Bond \\
\_\frown\_ & : Id \times BondSet \longrightarrow Neighbour \\
\_ \doteq \_ & : Id \times NeighbourSet \longrightarrow AdjacencyEq \\
\_(\!|\_|\!) & : MoleculeSet \times AdjacencyEqSet \longrightarrow \\
 & \quad MGraph
\end{array}
$$

where X takes sort values from the set $\{Molecule, Bond, Neighbour, AdjacencyEq\}$. The associative-commutative operator $\_,\_$ (union) is overloaded on each of the set sorts, and ϵ_{X} denotes the identity element (the empty set) for the operation $\_,\_$. We use ϵ instead of ϵ_{X} whenever the sort X can be easily deduced from the context. The subsorting relation $\mathrm{X} < \mathrm{X}Set$ states that each term of sort X can be seen as a set with a single element.

Since the molecular graphs represent a particular case of multigraphs with ports and we encoded them in the oriented version, we impose an orientation of bonds by choosing an arbitrary order $\succ$ on molecule names in the model and by considering that:

- a bond between two molecules of different name corresponds to a bond with the source labeled by the greater molecule name and the target labeled by the smaller molecule name, and
- a bond between two molecules of the same name is not oriented (or we can represent it equivalently as a bidirectional edge).

Sometimes biochemical reactions may occur without knowing all intermediate states or molecule names. This motivates us in using "unknown" names for molecules and sites as in [16] for beta-binders. We then choose to represent a bond between two molecules where at least one of them has an unknown name by an unoriented bond (or, equivalently, a bidirectional bond), and call this assumption *supra-estimation*. When a new molecule name is produced by

a reaction, the total order on names is extended for including it. One simple way to handle this extension is to consider any new name to be smaller than all existing names.

Let $\mathcal{X}$ be an $(\mathcal{S}, <)$-sorted family of variables. The axiomatizing involves theories with associativity, commutativity and unit element axioms (ACU) for the constructors $\_,\_$ of each set-like sort with ϵ the unit.

Definition 2. *We encode a molecular graph $G = (M, B)$ as an algebraic term $t_1(\!|t_2|\!)$ of sort MGraph where:*

- *$t_1 \in \mathcal{T}_{\Sigma,MoleculeSet}(\mathcal{X})$ is the set of all molecules in G, and*
- *$t_2 \in \mathcal{T}_{\Sigma,AdjacencyEqSet}(\mathcal{X})$ is the set of adjacency equations providing the neighbours for each molecule in M (if any) together with the labels of the incident bonds.*

Algebraic terms encoding molecular graphs must satisfy some structural properties in order to be considered *well-formed*: (i) each molecule identifier occurs at most once: in the molecule set, in the adjacency equation set as left-hand side of an adjacency equation, in the neighbour set of a molecule identifier; (ii) each molecule identifier or site occurring in the adjacency equation set must also occur in the molecule set; (iii) each site has at most one incident bond and its state is correct w.r.t. the incidence information provided in the adjacency equation set. Additionally, these terms must satisfy some *canonical form* requirements: (i) right-hand sides of adjacency equations are non-empty sets of neighbours; (ii) only non-empty set of bonds occur in neighbour terms.

Example 3. The molecular graph occurring in the left-hand side of the reaction pattern **r3** illustrated in Figure4 is encoded as the following term, assuming the total order on molecule names $EGFR \succ EGF \succ SHC \succ EGF.EGF$:

$$(\langle i : EGFR \parallel 1\hat{}\mathsf{b}, 2\hat{}\mathsf{v}, 4\hat{}\mathsf{v}\rangle, \langle j : EGFR \parallel 1\hat{}\mathsf{b}, 2\hat{}\mathsf{v}, 4\hat{}\mathsf{v}\rangle,$$
$$\langle k : EGF.EGF \parallel 2\hat{}\mathsf{b}, 2\hat{}\mathsf{b}\rangle)(\!|i \doteq k^\frown(1,2),\ j \doteq k^\frown(1,2)|\!)$$

4.2 Encoding the Reaction Patterns

For all rewrite rules over $\mathcal{T}_{\Sigma,MGraph}(\mathcal{X})$ we impose molecule identifiers occurring in the left-hand side to be variables. We say that a rewrite rule $t_1 \twoheadrightarrow t_2$ over $\mathcal{T}_{\Sigma,MGraph}(\mathcal{X})$ is well-formed (in canonical form) if both t_1 and t_2 are well-formed (in canonical form respectively). We call *bio-rewrite rule* a well-formed rewrite rule in canonical form.

Definition 4. *We encode a molecular graph rewrite rule $G_1 \rightsquigarrow G_2$ as a term rewrite rule $t_1 \twoheadrightarrow t_2$ where t_i encodes G_i, for $i = 1, 2$.*

The encoding of a molecular graph rewrite rule is a bio-rewrite rule since by definition the term encoding of a molecular graph is well-formed and in canonical form.

The *node-substitution* corresponding to a bio-rewrite rule, like for the encoding of transformations on multigraphs with ports, can be extracted automatically by means of a procedure GetMap analyzing the identifier occurrences of both sides of a rule. We usually omit the identity mappings. We make explicit the deletion of a molecule (resp. site) in the node-substitution by mapping it to a new molecule (resp. site) $\bullet$ called *black hole*.

Example 5. The molecular graph transformation **r1** illustrated in Figure 4 has the following encoding as a rewrite rule:
$(\langle i : EGF \parallel 1\hat{}\mathsf{v}, 2\hat{}\mathsf{h}\rangle, \langle j : EGF \parallel 1\hat{}\mathsf{v}, 2\hat{}\mathsf{h}\rangle)(\!|\epsilon|\!) \twoheadrightarrow$
$\langle i.j : EGF.EGF \parallel 1\hat{}\mathsf{b}, 1\hat{}\mathsf{b}, 2\hat{}\mathsf{v}, 2\hat{}\mathsf{v}\rangle(\!|i.j \doteq i.j\frown(1,1)|\!)$
with with the corresponding node-substitution:
$\langle i : EGF \parallel 1\hat{}\mathsf{v}, 2\hat{}\mathsf{h}\rangle \mapsto \langle i.j : EGF.EGF \parallel 1\hat{}\mathsf{b}, 1\hat{}\mathsf{b}, 2\hat{}\mathsf{v}, 2\hat{}\mathsf{v}\rangle$,
$\langle j : EGF \parallel 1\hat{}\mathsf{v}, 2\hat{}\mathsf{h}\rangle \mapsto \langle i.j : EGF.EGF \parallel 1\hat{}\mathsf{b}, 1\hat{}\mathsf{b}, 2\hat{}\mathsf{v}, 2\hat{}\mathsf{v}\rangle$.

4.3 Encoding the Reaction Pattern Application

The rewriting relation is the encoding of the multigraph transformation as presented in Section 2 with some particularities given by the instantiation of multigraphs with ports as molecular graphs.

The main ideas behind a rewriting step $t \xrightarrow{\overline{r}} t'$ with $r : t_1 \twoheadrightarrow t_2$ are the following:

- $\overline{r} : \overline{t_1 \twoheadrightarrow t_2}$ is the extended rule obtained from r by appending extension variables to each set-sorted subterm in the left-hand side and, accordingly, in the right-hand side.
- GetMap computes for a rewrite rule on *MGraph*-sorted terms the associated node-substitution using an analysis on the identifier occurrences.
- We use an ACU matching algorithm together with a matching rule which allows us to solve match equations where the left hand-side is a *NeighbourSet*- or *BondSet*-sorted term $t_1, \ldots, t_k$ (with $k \geq 2$) not containing extension variables, and the right-hand side is a singleton, t. Such cases arise due to supra-estimated bonds. The problem is solved by matching only one term t_i from the left-hand side against the term t from the right-hand side, discarding the other terms t_j, with $j \neq i$, and matching the extension variables against the set unit ϵ.
- We apply the substitution σ as usual. However, the result of the application of a substitution $\{x \mapsto t\}$ with $x, t : MoleculeId$ on adjacency equations requires a special treatment. For example, let us assume by supra-estimation that the nodes represented by x and y are connected by a bidirectional bond. If, after the application of a substitution σ, the name of $\sigma(y)$ is greater than the name of $\sigma(x)$, then the bond becomes unidirectional with $\sigma(x)$ the target and $\sigma(y)$ the source of the bond.
- We apply the instantiated node-substitution ξ^σ on $\sigma(\overline{t_2})$ by sequentially applying each of its elementary node-substitution on the term. The application of a node-substitution on an adjacency equation (or a neighbour) transforms

it in n adjacency equations (resp. neighbours), one for each corresponding node in the right-hand side of the mapping, and propagates the mapping application on the set of neighbours. After applying a node-substitution, the *MGraph*-terms may be neither well-formed, nor in canonical form.
- $\mathcal{S}$ is the rewrite system defining transformations on terms into well-formed *MGraph*-sorted terms in canonical form. Such rules (i) delete the adjacency equations for black holes and the black hole neighbours taking care of updating the states of incident sites, (ii) delete the extra-bonds (bonds whose endpoints do not appear among the sites of the connected nodes), (iii) merge elements having the same identifier, and (iv) delete neighbours with empty bond sets and adjacency equations with an empty neighbour sets.

Definition 6. *A term t of sort MGraph rewrites to a term t' using a* bio*-rewrite rule $r : t_1 \twoheadrightarrow t_2$ where $t_1, t_2 \in \mathcal{T}_{\Sigma,MGraph}(\mathcal{X})$ with $\overline{r} : \overline{t_1 \twoheadrightarrow t_2}$ and $\xi = \mathsf{GetMap}(\overline{r})$, if there exists a substitution σ solution of the* ACU*-matching problem $\overline{t_1} \ll t$ such that $t' = \xi^{\sigma}(\sigma(\overline{t_2})) \downarrow_{\mathcal{S}}$. We call this relation* bio-rewriting*, we say that t* bio-rewrites *to t' by r and denote it by $t \xrightarrow{\overline{r}} t'$.*

We do not illustrate here an example of a bio-rewriting step since it would be too technical (see [7] for such an example). However, it is important to emphasize that, according to the correspondence theorem in [7], the rewriting relation defined here on *MGraph*-terms encodes the transformation relation on molecular graphs.

4.4 ρ_{bio}-Calculus

Following closely the same steps for encoding the more general multigraphs with ports, we obtain for free a high-level calculus for molecular complexes which we call ρ_{bio}-calculus, similar to the ρ_{mg}-calculus. By extending the λ-calculus and term rewriting, the ρ_{bio}-calculus provides a way to combine rewriting and high-order functions. In addition, it fits in the direction given by Fontana and Buss [17] of using the λ-calculus as a formalism for studying biochemical systems.

5 Biochemical Network Generation

A biochemical system is not completely described by its components and the way they interact by means of reactions, but also by the behaviour of the system as a whole. Modeling the generation of a biochemical network amounts to defining how a set of reaction patterns is applied on a collection (set or multiset) of molecules. *Strategic rewriting* provides a formal model for expressing the control on the reaction rule application.

One rough way of describing the behaviour of the EGFR signalling cascade is to try to apply repeatedly all reaction patterns on the initial collection of molecules (see ζ_1 in Figure 6). We can easily prove that this strategy terminates since the number of free binding sites decreases or remains constant with every rule application.

$\zeta_1 = \texttt{repeat}(\texttt{onestep}(r_1, r_2, r_3, r_4, r_5))$
$\zeta_2 = \texttt{seq}(\texttt{repeat}(\texttt{onestep}(r_1, r_2, r_3)),\ \texttt{repeat}(\texttt{onestep}(r_4, r_5)))$
$\zeta_3 = \texttt{seq}(\texttt{repeat}(r_1),\ \texttt{repeat}(r_2),\ \texttt{repeat}(r_3),\ \texttt{repeat}(\texttt{onestep}(r_4, r_5)))$
$\zeta_4 = \texttt{seq}(\texttt{repeat}(\texttt{choice}(r_2, r_1)),\ \texttt{repeat}(r_3),\ \texttt{repeat}(\texttt{onestep}(r_4, r_5)))$
$\zeta_5 = \texttt{seq}(\texttt{repeat}(\texttt{choice}(r_2', r_2, r_1)),\ \texttt{repeat}(r_3),\ \texttt{repeat}(\texttt{onestep}(r_4, r_5)))$

Fig. 6. Different strategies for the modeling the network generation

The execution can be separated in two stages: the *first* one, $\{r_1, r_2, r_3\}$, is concerned with the extracellular interactions between the signals and the receptors, while the *second* one, $\{r_4, r_5\}$, with the RTK cascade. Consequently, the control is specified by the strategy ζ_2 as a repeated sequential composition of two strategies, one for each stage. We can specify that all *EGF* dimers are created in a first step, and only after that they bind to receptors by means of the strategy ζ_3, or we can specify by means of the strategy ζ_4 that an *EGF* dimeric form binds a receptor as soon as it is created by giving r_2 priority over r_1 using the `choice` strategy.

In order to disallow the occurrence of molecular graphs like the one in Figure 7(a) where each of the *EGF* dimeric forms binds one of the receptor and the binding of receptors is no longer possible, we introduce a new reaction rule **r2'** depicted in Figure 7(b). Providing **r2'** with a higher priority than **r2** corresponds to a dimerization of the receptors as soon as possible. We specify this using the strategy $\texttt{choice}(r_2', r_2)$ which tries to apply r_2' first, and only if this does not succeed, it applies the rule r_2. Hence the *EGF* dimer binds two receptors as soon as it is created by means of the strategy ζ_5. This last strategy applied on the initial molecular graph **G** illustrated in Figure 5 produces the molecular graph from Figure 7(c) modulo a relabeling of the dimeric forms of *EGF*. Usually we obtain an exhaustive generation of the network by repeating the application of the control strategies, for instance $\texttt{repeat}(\zeta_i)$, for $i \in \{2, \ldots, 5\}$.

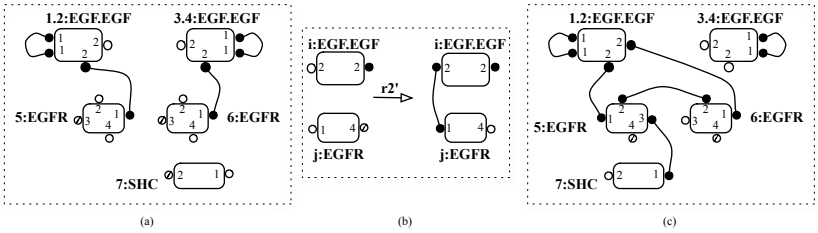


Fig. 7. (a) A possible graph obtained from applying **r2** on **G"** from Figure 5. (b) An additional rule which urges the binding of a dimeric form of *EGF* to a second receptor. (c) A molecular graph resulting from applying the strategy ζ_5 on the initial molecular graph from Figure 5.

The use of strategies at this stage provides our calculus with a declarative formalism for modeling a flexible control of rule applications. Different control strategies may be easily prototyped and experimented by selecting specific rules, grouping rules and applying each group with different strategies, and then linking them up. The advantages of using strategic rewriting for the automatic network generation have already been experimented in [18, 19] for a chemical application.

6 Conclusion

An inspiring starting point for our work was the graphical formalism presented in [6] for modeling biochemical networks where the protein complexes are represented by typed attributed graphs, and classes of reactions are modeled by graph transformation rules. This model considers also quantitative information on reactions. Our approach, developed on a rule-based modeling framework and extensively using expressive graphical representations (like the one of [6]), focuses on providing in addition a strategic rewriting dimension. This new component allows a flexible modeling for automatic generation of biochemical networks.

The ρ_{bio}-calculus introduced in this paper for modeling the interactions between molecular complexes is an extension of the ρ_{mg}-calculus [7]. The connectivity information in molecular complexes is essential in the definition of some reaction patterns since it can directly affects their reactivity; this kind of information is easily modeled by graph transformation rules. It is also possible to express negative constraints in the term encoding of the reaction patterns by means of recently developed anti-patterns [20], a concept already implemented in TOM. This allows us to specify for instance the absence of certain molecules or forbidden bonds, and hence to model halting conditions as forbidden occurrences of some particular molecular graphs.

Using graph rewriting and rule-based formalisms for modeling biological systems has already been done by several authors. A review of such rule-based formalisms can be found in [21] with emphasis on the capability of representing the topology of complexes. κ-calculus [3] is a language of formal proteins which models complexes as graphs-with-sites and their interactions as a particular graph-rewriting operation. It is derived from process calculus, and bonds are represented in complexes by shared names. A step forward, bioκ-calculus [4] combines the κ-calculus with the brane calculi [1] providing a more expressive formalism by modeling the effects of protein interactions on the interaction capabilities of membranes. In this paper, we initiate the development of a similar approach based on a different calculus.

Pathway Logic [5] is a rewriting system formalism, where proteins and cells are modeled by algebraic terms, and reactions by rewrite rules. It is designed to work on two levels of abstraction, one concerning the protein states, and the other concerning the protein-protein interactions handled by means of graph rewriting. The Maude [22] system is used for implementing Pathway Logic, providing executability of the specifications and analytic tools. As in Pathway Logic, we use algebraic terms and rewrite rules for modeling molecules and reactions

respectively, with the differences that we represent graphs using adjacency lists, and the control of rewrite rule applications is expressed in the calculus.

Based on our previous experience on implementing molecular chemistry [18, 19], we are currently implementing this calculus using TOM, a tool providing matching, normalization, and strategic rewriting in high-level programming languages like Java. An aspect we cannot neglect in modeling protein-protein interactions is the combinatorial complexity [21]. In order to obtain an efficient implementation, we make extensive use of the maximal term sharing provided by TOM.

Several further extensions of this work are possible:

We have conceived a qualitative model; by imposing kinetic information on reactions and on the initial set of biochemical species we should obtain biochemical networks allowing us to derive quantitative models as ODE-based models, stoichiometric models, time depending. An inspiring rule-based formalism for a qualitative modeling of biochemical process is found in BIOCHAM [23].

We focus in this paper on interactions between proteins at the level of functional domains. However, the model we propose can be easily tuned to represent other types of biomolecular interactions, such as protein-DNA or protein-lipid interactions [24], or even membrane interactions [1, 2].

Strategic rewriting is a suitable formalism for modeling the highly concurrent and non-deterministic behaviour of complex systems in general; in particular, the use of strategic rewriting makes possible to describe biochemical processes by expressions in a suitable strategy language. A first link between strategies and computation models inspired by biology is presented in [25] where control mechanisms in membrane systems are expressed by means of rewrite strategies.

Strategic rewriting opens many possibilities for modeling different aspects of biological systems at different abstraction levels. In particular, we plan to explore the capabilities of this formalism to model and reason about the adaptability and flexibility of cell behaviour. We do not limit our aim to modeling some well-known biological systems, but to help understand their behaviour and deduce new organization and behavioural principles. In the same vein as Păun in [26], we consider that reasoning at the level of strategies of computing (rewrite strategies), rather than at the tactic level (rewrite rules), is an incentive direction of formally studying biological systems.

References

1. Cardelli, L.: Brane Calculi. In Danos, V., Schächter, V., eds.: Computational Methods in Systems Biology, International Conference CMSB 2004, Paris, France, May 26-28, 2004, Revised Selected Papers. Volume 3082 of Lecture Notes in Computer Science., Springer (2005) 257–278
2. Cardelli, L., Pradalier, S.: Where Membranes Meet Complexes. In: Proceedings of BioCONCUR 2005: A workshop on concurrent models in molecular biology. (2005)
3. Danos, V., Laneve, C.: Formal molecular biology. Theoretical Computer Science **325**(1) (2004) 69–110

4. Laneve, C., Tarissan, F.: A simple calculus for proteins and cells. In: Proceedings of the Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC), Venice, Italy. (2006)
5. Talcott, C.L., Eker, S., Knapp, M., Lincoln, P., Laderoute, K.: Pathway Logic Modeling of Protein Functional Domains in Signal Transduction. In Altman, R.B., Dunker, A.K., Hunter, L., Jung, T.A., Klein, T.E., eds.: Biocomputing 2004, Proceedings of the Pacific Symposium, Hawaii, USA, 6-10 January 2004, World Scientific (2004) 568–580
6. Blinov, M.L., Yang, J., Faeder, J.R., Hlavacek, W.S.: Graph Theory for Rule-Based Modeling of Biochemical Networks. In Priami, C., Ingólfsdóttir, A., Mishra, B., Nielson, H.R., eds.: Transactions on Computational Systems Biology VII. Volume 4230 of Lecture Notes in Computer Science., Springer (2006) 89–106
7. Andrei, O., Kirchner, H.: A Rewriting Calculus for Multigraphs with Ports. In: Accepted in RULE'07. (2007)
8. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
9. Goguen, J.A., Meseguer, J.: Order-Sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations. Theoretical Computer Science **105**(2) (1992) 217–273
10. Kirchner, C., Kirchner, H., Vittek, M.: Designing Constraint Logic Programming Languages using Computational Systems. In Van Hentenryck, P., Saraswat, V., eds.: Principles and Practice of Constraint Programming. The Newport Papers. MIT Press (1995) 131–158
11. Visser, E.: Stratego: A Language for Program Transformation based on Rewriting Strategies. System Description of Stratego 0.5. In Middeldorp, A., ed.: Rewriting Techniques and Applications (RTA'01). Volume 2051 of Lecture Notes in Computer Science., Springer-Verlag (2001) 357–361
12. Balland, E., Brauner, P., Kopetz, R., Moreau, P.E., Reilles, A.: Tom: Piggybacking Rewriting on Java. In: To appear in Proceedings of RTA'07. (2007)
13. Eker, S., Martí-Oliet, N., Meseguer, J., Verdejo, A.: Deduction, Strategies, and Rewriting. In M. Archer and T. Boy de la Tour and C. A. Muñoz, ed.: 6th International Workshop on Strategies in Automated Deduction, STRATEGIES'06, Seattle, Washington, August 16, 2006, Part of FLOC 2006. Electronic Notes in Theoretical Computer Science, Elsevier (2007)
14. Cirstea, H., Kirchner, C.: The rewriting calculus - Part I and II. Logic Journal of the IGPL **9**(3) (2001) 427—498
15. Cirstea, H., Kirchner, C., Liquori, L.: Matching Power. In Middeldorp, A., ed.: RTA. Volume 2051 of Lecture Notes in Computer Science., Springer (2001) 77–92
16. Priami, C., Quaglia, P.: Beta Binders for Biological Interactions. In Danos, V., Schächter, V., eds.: Computational Methods in Systems Biology, International Conference CMSB 2004, Paris, France, May 26-28, 2004, Revised Selected Papers. Volume 3028 of Lecture Notes in Computer Science., Springer (2004) 20–33
17. Fontana, W., Buss, L.W.: The barrier of objects: From dynamical systems to bounded organizations. In Casti, J., Karlqvist, A., eds.: Boundaries and Barriers. Addison-Wesley (1996) 56–116
18. Bournez, O., Côme, G.M., Conraud, V., Kirchner, H., Ibanescu, L.: A Rule-Based Approach for Automated Generation of Kinetic Chemical Mechanisms. In Nieuwenhuis, R., ed.: Rewriting Techniques and Applications (RTA 2003). Volume 2706 of Lecture Notes in Computer Science., Springer (2003) 30–45

19. Andrei, O., Ibanescu, L., Kirchner, H.: Non-intrusive Formal Methods and Strategic Rewriting for a Chemical Application. In Futatsugi, K., Jouannaud, J.P., Meseguer, J., eds.: Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday. Volume 4060 of Lecture Notes in Computer Science., Springer (2006) 194–215
20. Kirchner, C., Kopetz, R., Moreau, P.E.: Anti-Pattern Matching. In: Proceedings of the 16th European Symposium on Programming - ESOP'07. Volume 4421 of Lecture Notes in Computer Science., Springer (2007) 110–124
21. Hlavacek, W.S., Faeder, J.R., Blinov, M.L., Posner, R.G., Hucka, M., Fontana, W.: Rules for Modeling Signal-Transduction Systems. Science STKE 334 (2006)
22. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Quesada, J.F.: Maude: specification and programming in rewriting logic. Theoretical Computer Science **285**(2) (2002) 187–243
23. Chabrier-Rivier, N., Fages, F., Soliman, S.: The Biochemical Abstract Machine BIOCHAM. In Danos, V., Schächter, V., eds.: Computational Methods in Systems Biology, International Conference CMSB 2004, Paris, France, May 26-28, 2004, Revised Selected Papers. Volume 3082 of Lecture Notes in Computer Science., Springer (2005) 172–191
24. Faeder, J.R., Blinov, M.L., Hlavacek, W.S.: Graphical rule-based representation of signal-transduction networks. In Haddad, H., Liebrock, L.M., Omicini, A., Wainwright, R.L., eds.: Proceedings of the 2005 ACM Symposium on Applied Computing (SAC), Santa Fe, New Mexico, USA, ACM (2005) 133–140
25. Andrei, O., Ciobanu, G., Lucanu, D.: Expressing Control Mechanisms in P systems by Rewriting Strategies. In Hoogeboom, H., Paun, G., Rozenberg, G., Salomaa, A., eds.: Membrane Computing, International Workshop, WMC7, Leiden, The Netherlands, 2006, Selected and Invited Papers. Volume 4361 of Lecture Notes in Computer Science., Springer (2006) 154–169
26. Paun, G.: Twenty Six Research Topics About Spiking Neural P Systems. Available at the P Systems Web Page: `http://psystems.disco.unimib.it`. (2006)