A Biochemical Calculus Based on Strategic Graph Rewriting

Oana Andrei
INRIA Nancy - Grand-Est & LORIA

Hélène Kirchner
INRIA Bordeaux - Sud-Ouest

France

Biological intuition

Graphs are suitable for describing the structure of complex systems and graph transformations for modeling their dynamic evolution.

We are interested in a particular representation of **molecular complexes** as graphs and of **reaction patterns** as graph transformations:

- the behavior of a protein is given by its functional domains / sites on the surface
- two proteins can interact by binding or changing the states of sites
- bound proteins form complexes that have a graph-like structure
- membranes can also form molecular complexes, called tissues

Port graphs are graphs with multiple edges and loops, where

- nodes have explicit connection points, called *ports*, and
- the edges attach to ports of nodes.

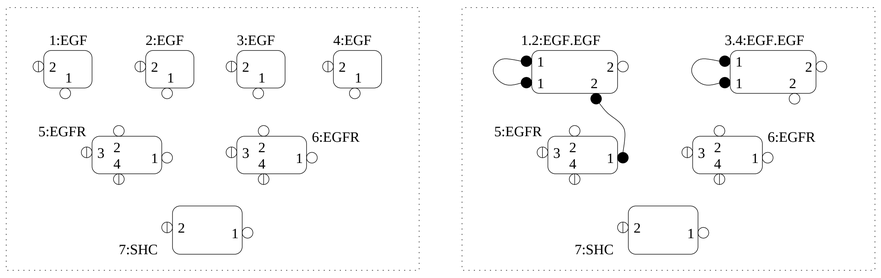
A **port graph rewrite rule** is a pair of port graphs $L \Rightarrow R$ with a correspondence between elements of L and elements of R. If we consider an arrow node embedding this correspondence, then a port graph rewrite rule is also port graph.
We used term rewriting to provide an operational semantics for port graph rewriting [AK07].

A molecular graph is a particular type of port graph as described in the following table:

Molecular graph	Port graph
protein	node
site	port with maximum degree 1
bond	edge

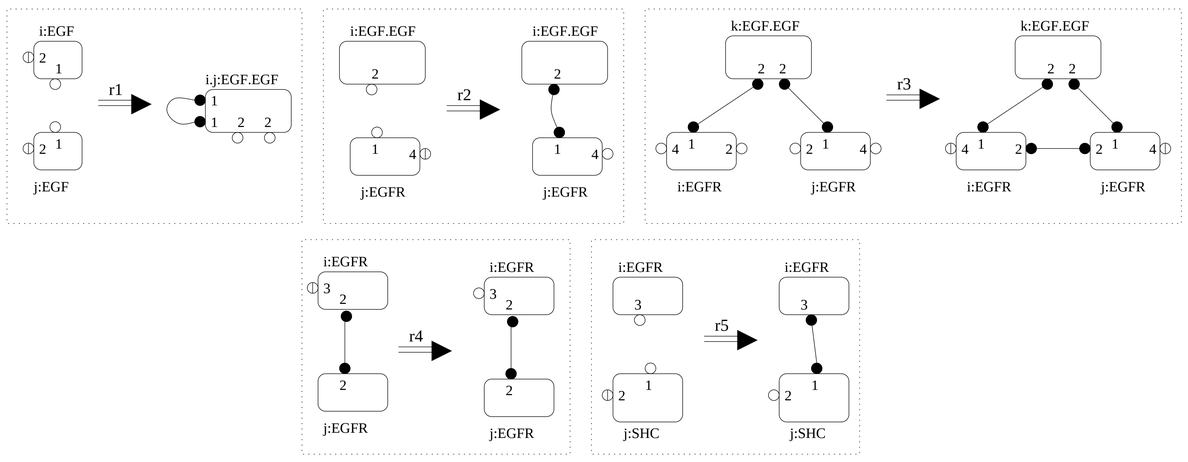
Any **transformation of molecular complexes** is represented as a molecular graph rewrite rule which is a particular port graph rewrite rule, hence a port graph.
Port graphs represent a unifying structure for representing both molecular complexes and the reaction patterns on them.

We illustrate a fragment of the EGFR signalling cascade [DL04]. The protagonists of this model are the signal protein EGF, the transmembrane protein EGFR, and the adapter protein SHC. A protein or node is graphically represent as a box with an unique identifier and a name placed outside the box. A site is represented as a filled, empty, or slashed circle on the surface of the box if its state is respectively bound, free, or hidden. The molecular graph G below represents the initial state of the system modeling a fragment of the EGFR signaling cascade, while G' represents a subsequent state.



G G'

The signalling information is propagated from the outside of the cell to its interior following the reaction patterns depicted below as molecular graph rewrite rules:



A higher-order biochemical calculus

The **chemical model of computation** was introduced by the Γ language [BM86]:

- based on a chemical solution where molecules interact freely according to (conditional) reaction rules
- uses multisets for modeling the chemical solutions
- uses multiset rewrite rules for modeling the reaction rules
- extended to the CHemical Abstract Machine (CHAM) [BB92], the γ-calculus and HOCL [BFR06].

We extend the chemical model with high-level features by considering a molecular graph structure for the data and molecular graph rewrite rules for the computation rules. The result is **a molecular graph rewriting calculus with higher-order capabilities** which generalizes γ-calculus through a more powerful abstraction power that considers for matching not only a variable but a port graph with variables. It also encompasses the rewriting calculus (the ρ-calculus) [CK01] and the term graph rewriting calculus (the ρ_g-calculus) [BBCK05].

Molecular Graphs	$\mathcal{M}$
Abstractions	$\mathcal{A} ::= \mathcal{M} \Rightarrow \mathcal{G}$
Graphs	$\mathcal{G} ::= \mathcal{X} \mid \mathcal{M} \mid \mathcal{A} \mid \mathcal{G}\ \mathcal{G} \mid \varepsilon$
Simple Worlds (States)	$\mathcal{V} ::= \mathcal{Y} \mid [\mathcal{G}]$

Syntax

where the juxtaposition corresponds to a commutative operator, while [ ] corresponds to a permutative variadic operator. All entities enumerated in the syntax of the calculus above have a port graph structure.

Due to the intrinsic parallel nature of rewriting on disjoint redexes and decentralized rule application, we model a kind of *Brownian motion*, a basic principle in the chemical paradigm. An interaction takes place in a system by heating it up. This process isolates an abstraction and a molecular graph for application by connecting them to an application node @. All steps computing the application of abstractions to a molecular graph, including the matching and the replacement operations, are expressible using port graph transformations by considering some more auxiliary nodes and extending the reduction relation.

(Heating)	$[X\ A\ M] \longmapsto [X\ A@M]$
(Application/Success)	$A@M \longmapsto G$ **if** $M \to_A G$
(Application/Fail)	$A@M \longmapsto A\ M$ **otherwise**

Semantics

By introducing an explicit object (node) for failure, stk, we gain in expressivity:

$$(\textbf{Application/Fail}') \qquad A@M \longmapsto \mathsf{stk} \qquad \textbf{if } M \textbf{ is } A - \textbf{irreducible}$$

Expressing control mechanisms in the calculus

Instead of this highly non-deterministic and non-terminating behaviour of abstraction application, one may want to introduce some **control** to compose or choose the abstractions to apply, possibly exploiting failure information. The formalism permitting such concept in a rewriting-based framework is represented by the **rewriting strategies**.

The basic strategies are the molecular graph rewrite rules and the identity (`id`) and failure (`fail`) strategies. Based on them, strategies expressing the control can be constructed, like the sequence (`seq`), the left-biased choice (`first`), the application of a strategy only if it is successful (`try`), and the repeating strategy (`repeat`). All these strategies can be described as abstractions, therefore they become objects of the calculus:

$$\begin{aligned}
\mathsf{id} &\triangleq X \Rightarrow X \\
\mathtt{fail} &\triangleq X \Rightarrow \mathsf{stk} \\
\mathtt{seq}(S_1, S_2) &\triangleq X \Rightarrow S_2@(S_1@X) \\
\mathtt{first}(S_1, S_2) &\triangleq X \Rightarrow (S_1@X)\ \ (\mathsf{stk} \Rightarrow (S_2@X))@(S_1@X) \\
\mathtt{try}(S) &\triangleq \mathtt{first}(S, \mathsf{id}) \\
\mathtt{repeat}(S) &\triangleq \mathtt{try}(\mathtt{seq}(S, \mathtt{repeat}(S)))
\end{aligned}$$

Based on strategies, we can increase the expressivity of the calculus by considering for instance:

1. **Failure catching**: if $S@M$ reduces to the failure construct stk, then the strategy $\mathtt{try}(\mathsf{stk} \Rightarrow S\ M)$ restores the initial entities subject to reduction.

$$(\textbf{Heating}') \qquad [X\ S\ M] \longmapsto [X\ \mathtt{seq}(S, \mathtt{try}(\mathsf{stk} \Rightarrow S\ M))@M]$$

2. **Persistent strategies**: $S!$ applies S to an object and, if successful, replicates itself.

$$S! \triangleq \mathtt{seq}(S, \mathtt{first}(\mathsf{stk} \Rightarrow \mathsf{stk}, Y \Rightarrow Y\ S!))$$

Possible extensions

- One possible refinements concerns the management of a structure of **all possible results** issued from the application rule.
- Verification issues:
  - identifying conditions on abstractions for **accessibility** of **stable states** of modeled systems, or for imposing **fairness** on the application of abstractions;
  - integrating verification techniques in the calculus.
- Another interesting feature worth and quite natural to be defined in the calculus represents the possibility of modifying or deleting abstractions as objects of the calculus, with application in modeling **cellular dedifferentiation** for instance.

References

[AK07] Oana Andrei and Hélène Kirchner. A Rewriting Calculus for Multigraphs with Ports. In *Proceedings of RULE'07*, 2007.

[BB92] Gérard Berry and Gérard Boudol. The Chemical Abstract Machine. *Theoretical Computer Science*, 96(1):217–248, 1992.

[BBCK05] Clara Bertolissi, Paolo Baldan, Horatiu Cirstea, and Claude Kirchner. A Rewriting Calculus for Cyclic Higher-order Term Graphs. *Electronic Notes in Theoretical Computer Science*, 127(5):21–41, 2005.

[BFR06] Jean-Pierre Banâtre, Pascal Fradet, and Yann Radenac. A Generalized Higher-Order Chemical Computation Model. *Electronic Notes in Theoretical Computer Science*, 135(3):3–13, 2006.

[BM86] Jean-Pierre Banatre and Daniel Le Metayer. A New Computational Model and Its Discipline of Programming. Technical Report RR-566, INRIA, 1986.

[CK01] Horatiu Cirstea and Claude Kirchner. The Rewriting Calculus - Part I and II. *Logic Journal of the IGPL*, 9(3):427—498, 2001.

[DL04] Vincent Danos and Cosimo Laneve. Formal Molecular Biology. *Theoretical Computer Science*, 325(1):69–110, 2004.