Bioware Languages

Luca Cardelli

Microsoft Research

Reflecting joint work with Ehud Shapiro and Aviv Regev,
Weizmann Institute of Science.

Introduction

This work can be seen as example of an emerging class of languages for describing, and possibly programming, biological systems (bioware). A living cell is, to a rather surprising extent, an information processing device [1]. One can envision describing precisely such complex biological systems, and then driving simulation and analysis from such descriptions. One can even imagine one day "compiling" bioware languages into real biological systems, just like silicon chips are today compiled from hardware languages.

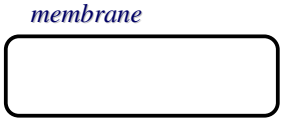
Biological systems, far from being unstructured chemical soups, employ membranes to organize and isolate chemical reactions and their products. Hierarchies of membranes are a necessary component of any description of such system. The π-calculus [3] has been used to model chemical reactions [6]. As an extension [7], the ambient calculus [2], which is based on a dynamic hierarchy of containers, can be used to model biological interactions. (Stochastic aspects can be handled, but are not discussed here [5].)

We represent biological systems with a graphical (rather than textual) notation; this is somewhat natural because of the aspect and hierarchical structure of many such systems. It is also possible to provide a formal textual notation and related semantics, using standard techniques from process calculi. Moreover, it is possible to provide a formal graphical notation and related semantics, as a special case of Milner's BiGraphs. But here we just present a (formalizable) graphical notation: the graphical language of *biographs*.

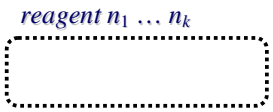
Biographs

A biograph represents a biological system via three primitive constructions and eight basic reactions. (The number of reactions could be reduced, but it then becomes harder to program 'instantaneous' reactions.)

Membranes. For our purposes, a *membrane* is simply a boundary that confines reactions to its interior, unless these are reactions that explicitly interact with a membrane as discussed below. Graphically, a membrane may contain reagents or other membranes. Membranes are nameless, but it useful to attach *comments* to them (e.g. "cell membrane" or "virus capsid").



Reagents. A *reagent* represents a biological (or, for the matter, chemical) entity that is ready to interact with some other biological entity. Reagents typically represent protein complexes that are ready to bind to each other and to transform each other as a result. Rather than considering the countless protein structures that exists in reality, we take a fixed set of primitive reagents, enumerated later, that can be used to express a large class of interactions (the formalism is, in fact, Turing-complete). Each reagent is parameterized by a number of *binding sites*. These binding sites are named by *pure names* [4] $n_1 \ldots n_k$, that is, names that have no structure other than their identity. Graphically, a reagent encloses the future product of its activation inside a dotted line.



Binding. The binding of, e.g., a protein to a ligand, can be represented as a binding site (a pure name) *n* that is privately shared by two reagents. A *binding box* represents a region where a pure name *n* is privately shared. Unlike membranes, which have physical existence, binding boxes are more of a bookkeeping device. A binding box for *n* can graphically expand, contract, and cross other membranes and binding boxes, as long as this process does not lead to revealing *n* or to confusing it with some other *n*.

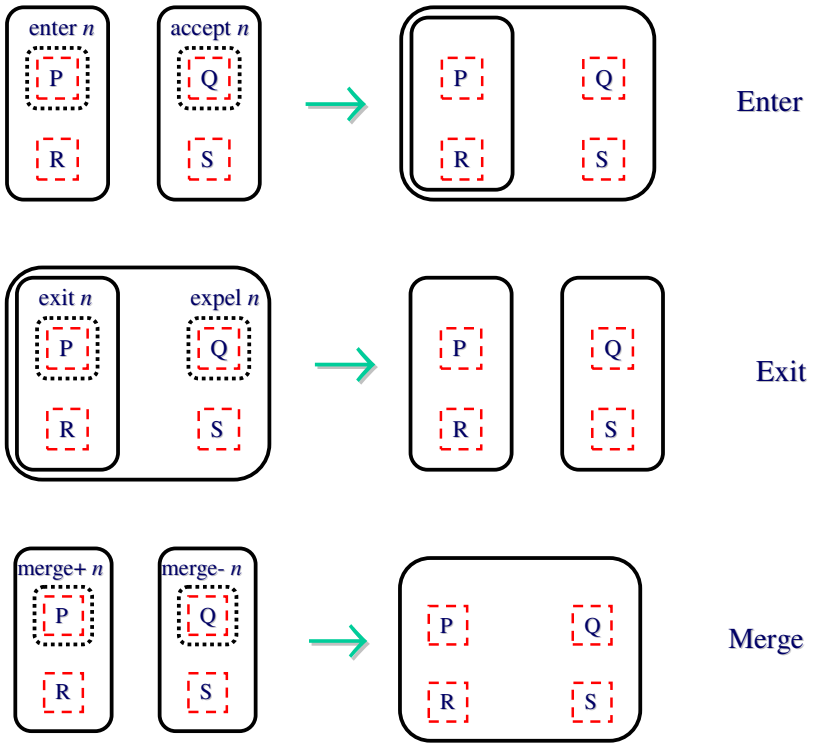
Named Subsystems. This is meta-notation for subsystems, used when expressing general interaction rules (named subsystems do not occur in specific system instances). The notation below represents a subsystem (the dashed boundary) that is named P so we can refer to it. Sometimes we need to apply a name replacement {*m*/*n*} (replacing *m* with *n*) to a still undetermined subsystem; the name replacement then sits on the boundary, until later when the subsystem is determined and the replacement can be applied.

Membrane Reactions

We start by describing reagents that affect membranes. These reagents typically represent protein complexes that sit on or across a membrane, and cause membranes to interact with each other. Graphically, these reagents are drawn inside the membrane that they actually sit on or across, so that they are transported along with the membrane.

On the left of the reaction arrow we have the situation before the interaction, and on the right we have the situation after the interaction.

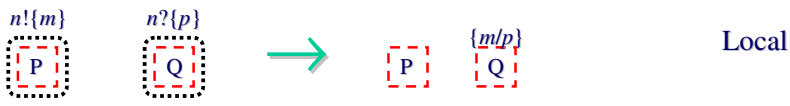
The first reaction describes a membrane that enters another contiguous membrane, through the interaction of two specific reagents, *enter* and *accept*, that have a common binding site *n*. Here P and Q represent the residuals of the interacting reagents (which could be void), while R and S represent whatever else is initially contained in the membranes. The following two reactions describe the effects of reagents that cause membranes to exit each other (*exit* and *expel*) or to merge (*merge+* and *merge-*), each based on a common interaction site *n*.



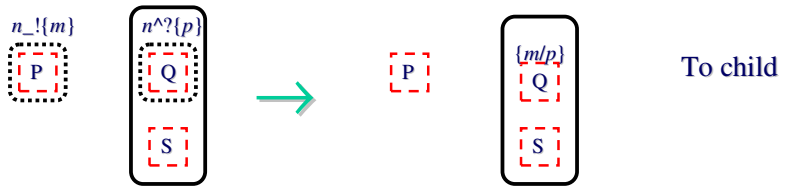
Site Reactions

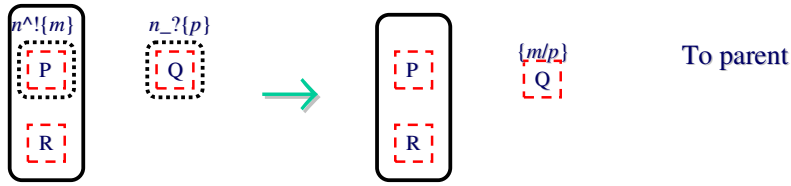
The next group of reactions do not affect membranes (although membranes may be involved), but only affect reagents. In these reactions, reagents interact on a binding site *n*, and can also exchange tokens *m*. These tokens can represent further binding sites, or other entities that get passed along in reactions (e.g., electrons or small molecules).

The first site reaction represents a pure chemical reaction: two molecules interact and produce two other molecules, within the confines of some common solution (the two molecules must be inside the same membrane, if any). The two complementary molecules are indicated by *n*! and *n*?. The common name *n* means that they can interact, and the !,? pair determines the direction of the interaction. In full, *n*!{*m*}(P) means that this is a molecule that, when interacting, provides a token *m* to the other molecule, and transforms itself into P. Instead, *n*?{*p*}(Q) means that this other molecule receives some token *m*, and transforms itself into Q{*m*/*p*}. Here *p* is really a formal input parameter, and Q{*m*/*p*} is Q where the formal *p* is replaced by the actual *m*.

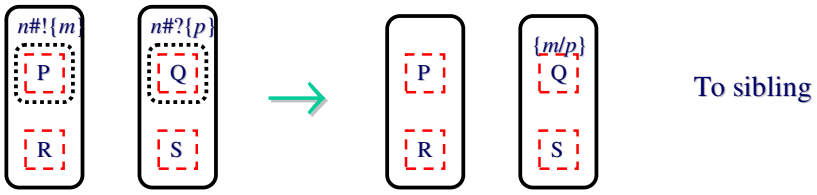


The next two reactions are similar, but the interaction between reagents happens across a membrane. The exchanged token *m* flows either down through a membrane (indicated by '_') or up through a membrane (indicated by '^').



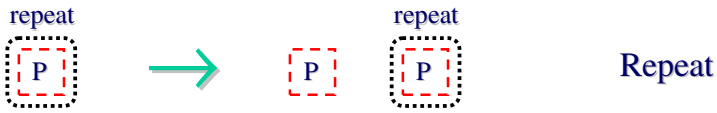


Finally, we have a reaction where the token *m* flows through two sibling membranes (indicated by '#').

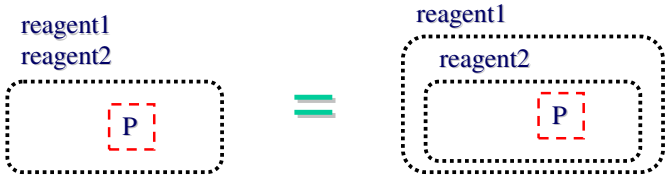


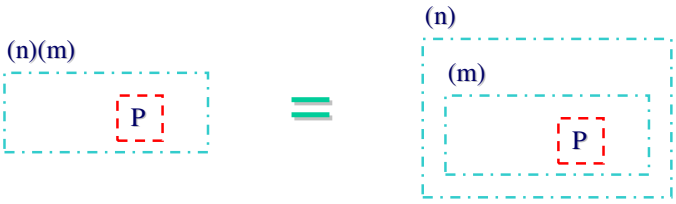
Repeat Reaction and Some Abbreviations

A "repeat" reagent creates new copies of a given reagent or subsystem. This models, abstractly, unbounded resources and processes.



Moreover, we use some graphical abbreviations, to simplify drawings:

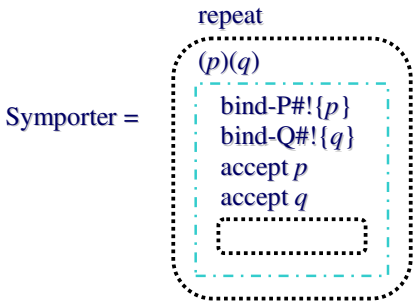




Example: Symporter

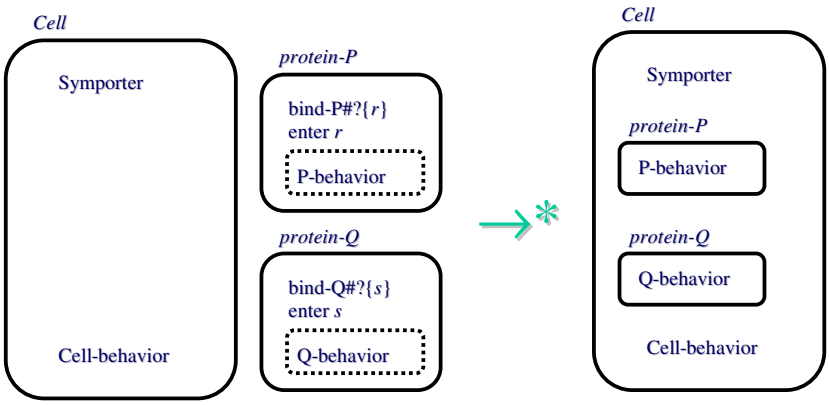
A *symporter* is a molecular channel. It binds two specific proteins, here called protein-P and protein-Q, from outside the cell in either order, and then simultaneously transports them inside the cell.

The symporter subsystem can repeat its behavior indefinitely (given sufficient energy, which is not modeled), and persists within the cell. It is first written separately, and then indicated by name in the larger system below. Two interaction sites, bind-P and bind-Q, represent the binding sites of the symporter with any instance of protein-P and protein-Q respectively. Each repeated interactions uses a fresh pair of distinct tokens *p*,*q*, which represent bindings with specific protein instances. After an instance of a protein is bound, nothing can then interfere with that binding because nothing else knows the freshly created pure names *p*,*q*.



The whole system then looks like the picture below. Initially a cell contains a symporter and whatever else, and is contiguous (that is, within the same surrounding membrane, if any) with instances of protein-P and protein-Q. Note that the proteins are themselves modeled as membranes: this is common because protein complexes can have a complicated structure.

After a sequence of reactions, during which the proteins are bound in either order, the proteins are both transported inside the cell membrane. Each reaction in the sequence is an instance of one of the reactions explained previously.



Although this protocol works under 'ordinary conditions', it is not perfect, and one can study ways in which it can be subverted. In fact, this is an important reason for modeling biological systems in all their complexity: many drugs and natural defenses work by subverting natural pathways. We need to model biological systems in order to understand them, but also to study how they can or cannot be tampered with at any level of abstraction.

References

[1] Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., Watson, J.D. (1994) Molecular Biology of the Cell. Garland Publishing.

[2] Cardelli, L., Gordon, A.D. (2000) Mobile Ambients. Theoretical Computer Science, Vol 240/1, June 2000. pp 177-213.

[3] Milner, R. (1999) Communicating and Mobile Systems: the Pi-Calculus. Cambridge University Press.

[4] Needham. R.M., Names. In S. Mullender, ed., Distributed Systems, pp 89-101. Addison-Wesley, 1989.

[5] Priami, C., Regev, A., Silverman, W., and Shapiro, E. (2001) Application of stochastic process algebras to bioinformatics of molecular processes. Information Processing Letters. 80, 25-31.

[6] Regev, A., Silverman, W., and Shapiro, E. (2001) Representation and simulation of biochemical processes using the pi-calculus process algebra**.** Proceedings of the Pacific Symposium of Biocomputing 2001 (PSB2001), 6: 459-470.

[7] Regev, A., Ph.D. Thesis, to appear.