A Calculus for Modelling, Simulating and Analysing Compartmentalized Biological Systems

Radu Mardare and Adaoha Ihekwaba

Microsoft Research-University of Trento
Centre for Computational and Systems Biology
Trento, Italy

Abstract. This paper introduces Protein Calculus, a special modeling language designed for encoding and calculating the behaviors of compartmentilized biological systems. The formalism combines, in a unified framework, two successful computational paradigms - process algebras and membrane systems. The goal of Protein Calculus is to provide a formal tool for transforming collected information from *in vivo* experiments into coded definition of the different types of proteins, complexes of proteins, and membrane-organized systems of such entities. Using this encoded information as input, our calculus computes, *in silico*, the possible behaviors of a living system.

Keywords: biological systems, modeling languages, process algebra, membrane systems
PACS: <Missing classification>

THE COMPUTATIONAL CHALLENGE IN SYSTEMS BIOLOGY

Many of the activities of the biological cells are regulated by proteins which carry signals that modify the expression of different genes at a given time, but how these signals do so is not known. The use of Systems Biology is therefore essential if we are ever to make sense of the biological complexity, as intuitive 'conceptual'models.

Systems Biology is a multi-disciplinary approach that studies the relationships and interactions of various cellular mechanisms and cellular components. It encompases the growing number of mechanistic but largely isolated insights and increasingly, high-throughput 'omics' data sets which tends to be semi-quantitative and specific to its experimental system and integrates them into a conceptual modelling framework that is holistic, quantitative and predictive.

By abstracting biological systems on the level of their behaviours, models are obtained sharing many characteristics with computational systems. Consequently, on all levels of organisation (i.e. from the atomic to full organisms) concurrent behaviours - be it event-driven, causal, time-dependent or distributed - is attained. Owing to these similarities, modelling (reverse-engineering, simulating and analysing) a biological system has been considered in a similar way to the engineering of complex artificial systems [2, 4]. This has resulted in questions such as - Is the structure of a cell much more complex than the structure of a jet plane? - to be raised. In answering the question, the challenge therefore is to construct, *in silico*, reliable 'copies' of biological systems that would allow the simulation of various experiments, which will then assist in proposing and verifying hypotheses. Recent small-scale modelling efforts in this direction have had encouraging results [14, 6, 5].

Traditionally, mathematical modelling of biological systems have always involved the use of ordinary differential equations (ODEs) for the reason that they offer a description of networks as continuous time dynamical systems as well as the added advantage of being a mature technique with many existing mathematical and computational resources available. However in the context of this work - use of ODE is limited - due to the highly non-linear behaviours exhibited by bio-systems, in addition to the lack of compositionality and systemic view of ODE-based techniques. ODEs simply model the bio-system as a whole and compute its overall behaviour, making it impossible for the description of the model to be disassembled into its constituent components. As a consequence, updating or extension of the model is not easily done when new information on the system becomes available.

For obtaining reliable models *in silico*, we expect to be able to combine small-scale models of molecular or cellular networks in large-scale models that might be used as valid images of complex biological phenomena. We also expect it to be possible to integrate new information into the models already developed. The need for small-scale modelling as a first step in designing complex models emerges from the methods of Molecular Biology - typically a molecular biologist spends years of research in studying a particular molecule. This information can be eventually used for designing a model of that molecule, but for further analysis, the ability to integrate such small models in order to

define big scenarios as the behaviours of an organ or of an organism is required. Therefore, if the behaviour of a molecule is described as a program (possibly involving a high level of indeterminism) then by running the program in a concurrent environment bigger scenarios can expect to be obtained.

Following this intuition, alternative approaches inspired by different paradigms from Computer Science have been recently proposed to complement the ODE approach while ensuring compositionality and extendability of the models. An example of such approach is process algebra.

Process algebras [1] are developed for designing and analysing complex systems of agents organised in a modular way and able to interact, collaborate, communicate, move and compose/decompose composite agents. In this paradigm, agents are understood as spatially localised and independently observable units of behaviour and computation (e.g. programs or processors running in parallel). They are successfully used in modelling distributed systems like computer networks. Recently the paradigm has been demonstrated to be particularly appropriate for modelling molecular and cellular networks [2, 14, 10, 11]. In this approach, a biological system is modelled as a network of modules (subsystems) that run in a decentralised manner, where they interact and change their relative positions which results in a global behaviour change of the systems generated. As a result, the overall model of the system can easily be obtained by mixing the models of its components. What's more, this paradigm was adapted to include quantitative and stochastic information [7, 13] - which are essential features of bio-chemical processes - and used together with stochastic simulators, built on the Gillespie method, for assisting in model validation and hypothesis forming. A stochastic simulator uses data from actual measurements and computes the expected behaviour of the system. In this manner, the high cost of running *in vivo* experiments is reduced, due to the derivation and running of *in silico* models. This method has proved to be effective for systems on small-scale levels, such as cellular systems and has been integrated in BetaWB [15], Cyto-Sim [16], BioSPI [13], which are simulators that have been successfully used in modelling and analysing signalling pathways.

Yet while in modelling biochemical soups, the straight concurrent environment is successful, it is still not sufficient for describing the complex behaviour of living systems. Biological systems involve compartmentalisation [3], which plays a major role in the functioning of living systems. They are not just unstructured, heterogeneous chemical soups, but spatially distributed hierarchies of bio-compartments organising and isolating chemical reactions and their products. On the level of cellular and molecular systems we can identify membrane-bound compartments including cells, organelles and vesicles each enclosed in a membrane, which isolates the bio-chemical components of a compartment from the external environment. These membranes organise a hierarchy of locations in the system. Then we have also molecular compartments, including stable or transient multi-molecular complexes, which insulate the component molecules from the environment. All these compartments have the role of a key regulatory mechanism for the biological system (in order to perform its function, a molecule must be present in the right location), organising it in a modular way.

The biological compartments can be extremely dynamic. The evolution of a biological system can involve movements of molecules between compartments (e.g. Golgi apparatus) and dynamic rearrangements of cellular compartments and molecular complexes (i.e. molecules may join a molecular compartment by binding to one of its members). There are also compartments that move, as a whole, with respect to the topology of the system. A typical example in this respect is phagocytosis (a cell is engulfed by another cell, e.g. macrophages 'eat' viruses or contaminated cells) or entry of a complex molecule into an organelle. Sometimes two membrane-bound compartments merge, forming one compartment. An approach known as membrane computing would be adventageous.

Membrane computing [12] was introduced as a new paradigm in computer science aiming to abstract computing models from the structure and the functioning of living cells. It focuses on the idea of dividing the space of computation by means of *"membranes"*. Accordingly, *membrane systems* are computational devices designed for modelling membrane-distributed and parallel computational scenarios. The membrane has the role of a separator of the space of computation. In between membranes there are distributed objects that can interact or move from one membrane to others following some laws (similar with the bio-chemical laws). Thus, membrane computing is a computational framework, inspired from biological reality, and designed to model compartmentalized phenomena. It has been proved useful for applications in computer science and essential in modelling biological systems [4, 8, 9] due to the role of the compartments in generating complex behaviours.

PROTEIN CALCULUS

In this report, we introduce Protein Calculus, a calculus that combines the process algebra and membrane systems paradigms in a unified framework designed for modelling compartmentalized systems of molecules. The calculus

focuses on bimolecular interactions between molecules which are modelled as objects spatially organized by a network of membranes.

With respect to the membrane, a protein can be placed inside the area enclosed by the membrane, outside it or on the membrane. A protein situated on a membrane can interact with internal or external proteins, but the proteins enclosed in the inner space delimited by a membrane cannot interact with outside proteins. The bimolecular interactions are of two types: complexation and decomplexation. The complexation is the phenomenon of binding proteins by means of their active sites. The result, a complex of proteins, which is a complex object that can be further involved in bimolecular interactions. Decomplexation however is the reverse phenomenon and consists of splitting a complex of proteins into the constituent parts. The following section gives a description of the syntax used.

Syntax

The syntax of Protein Calculus is presented by a grammar with non-terminals $\{P,C,S\}$ and is shown in Table 1 - where a desired system is represented by two disjoint alphabets, Ω and Σ. For the system, the elements of Ω is used to name proteins (i.e. each element $a \in \Omega$ appears in the syntax of a protein to identify it), whilst the elements of Σ is used for the interaction sites of the protein. With this in mind - $\alpha.a : \beta.a$ -denotes a protein "a" with two active sites α and β. Note - that the operator ":" is used to express the coexistence of different active sites.

The same operator can also be used to represent complexes of proteins. For instance, $\alpha.b : \beta.c : \gamma.c$ represents a complex formed by two proteins b and c, where, after complexation, b has only one active site α whilst c has two active sites, β and γ. By using these sites the complex can be further involved in other complexations.

However, we know that proteins and complexes tend to float in the same compartment / soup, in making allowances for this - the operator "$\|$" is used. In other words, in a given system, when protein $\alpha.a : \beta.a$ shares the same solution with complex $\alpha.b : \beta.c : \gamma.c$, the solution $\alpha.a : \beta.a \parallel \alpha.b : \beta.c : \gamma.c$ is given.

Anywhere structure $\frac{S}{M}$ is given, this will be used to express a membrane enclosed system, where the fraction line represents a membrane and encodes the way in which the space is separated. The structure below the fraction line, M, represents the structure enclosed by the membrane, while the structure above the fraction line, S, represents the soup of molecules that floats on the membrane.

The use of the fraction line for representing the division of space has the following advantage; in that $\frac{S}{M} \parallel M'$ can be used to represent a cell having the membrane composed by S, the content M and the environment in which it evolves M'. Obviously M and M' can be further described as composed by other membranes.

TABLE 1. Syntax of Protein Calculus

$P ::= a \mid \alpha.P \mid P : P$	$a \in \Omega, \alpha \in \Sigma$	$(proteins)$
$C ::= P \mid C.\alpha_C \mid C : C$	$\alpha \in \Sigma$	$(complexes)$
$S ::= C \mid S \parallel S$		$(soups)$
$M ::= S \mid \frac{S}{M} \mid M \parallel M$		$(membrane\ systems)$

Consider, for example, the structure in Figure1. We have a membrane system composed by a main membrane that enclose other four membranes between which the fourth one encloses a fifth membrane. On the membrane and between membranes we have proteins and complexes of proteins. Using our syntax we can represent this system with the following equation

$$\frac{A \parallel U \parallel (A : B : D) \parallel B \parallel A}{\frac{V}{N} \parallel F \parallel \frac{G}{F : E} \parallel L \parallel \frac{H \parallel D}{(A : B) \parallel K} \parallel \frac{M \parallel A \parallel T}{(V : Q : R) \parallel N \parallel \frac{D}{E}}}.$$

In the above representation $A,B,D,E,F,G,H,K,L,N,Q,R,U,V$ are uninterpreted proteins that can be further defined using our syntax, and $(A : B : D)$, $(F : E)$, $(A : B)$ and $(V : Q : R)$ are complexes of proteins.

This syntax therefore, could be used to describe a membrane system in any state. But you ask what of the behaviour? The operational semantics of our calculus will be used to compute this.

Operational semantics

The behaviour of a membrane system is encoded by some basic axioms shown in Table2. "$\phi \longrightarrow \phi'$" is used to mark the evolution of the system from a state ϕ to the state ϕ' in any contextual situation, while "$\phi \Longrightarrow \phi'$" marks the evolution governed by side conditions yet to be defined in the lab. (C_1) is a complexation axiom that establishes the type of protein or complexes that binds when floating in the same environment, while (D_1) is the corresponding decomplexation. Similarly (C_1) and (C_2) dually with $(D_1), (D_2)$ are complexation/decomplexation axioms, but are used to represent conditions of bimolecular interactions case where one of the molecules is anchored on the membrane and the other is floating freely in the inner respectively outer regions. Note - that (D_1) and (D_2) have identical premises, but their products differ depending on some contextual conditions to be defined in the lab for particular systems of proteins.

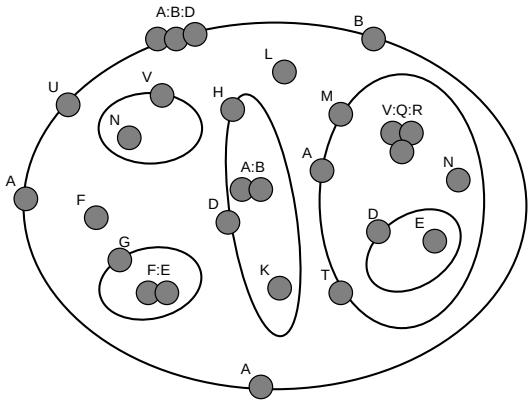

FIGURE 1. A membrane system

TABLE 2. Operational semantics

$$(C_1):\ \alpha.p \parallel \overline{\alpha}.p' \parallel s \longrightarrow p.\alpha_{p'} : p'.\overline{\alpha}_p \parallel s \qquad (D_1):\ p.\alpha_{p'} : p'.\overline{\alpha}_p \parallel s \longrightarrow \alpha.p \parallel \overline{\alpha}.p' \parallel s$$

$$(C_2):\ \frac{\alpha.p \parallel s}{\overline{\alpha}.p' \parallel m} \longrightarrow \frac{p.\alpha_{p'} : p'.\overline{\alpha}_p \parallel s}{m} \qquad (D_2):\ \frac{p.\alpha_{p'} : p'.\overline{\alpha}_p \parallel s}{m} \Longrightarrow \frac{\alpha.p \parallel s}{\overline{\alpha}.p' \parallel m}$$

$$(C_3):\ \frac{\alpha.p \parallel s}{m} \parallel \overline{\alpha}.p' \longrightarrow \frac{p.\alpha_{p'} : p'.\overline{\alpha}_p \parallel s}{m} \qquad (D_3):\ \frac{p.\alpha_{p'} : p'.\overline{\alpha}_p \parallel s}{m} \Longrightarrow \frac{\alpha.p \parallel s}{m} \parallel \overline{\alpha}.p'$$

$$(S):\ \frac{s \parallel s'}{m \parallel m'} \Longrightarrow \frac{s}{m} \parallel \frac{s'}{m'} \qquad (J):\ \frac{s}{m} \parallel \frac{s'}{m'} \Longrightarrow \frac{s \parallel s'}{m \parallel m'}$$

$$(P):\ \frac{s \parallel s'}{m} \parallel m' \Longrightarrow \frac{s}{m \parallel \frac{s'}{m'}} \qquad (E):\ \frac{s}{m \parallel \frac{s'}{m'}} \Longrightarrow \frac{s \parallel s'}{m} \parallel m'$$

(S) and (J) are the split and join axioms that model the division of a cell or merging of two membranes. (P) and (E) encodes the phagocitosys and exocitosys phenomena respectively.

CONCLUDING REMARKS

Process calculus is an invaluable technique that can be used for describing the molecular systems and their behaviours; for instance in modelling trans-membrane interactions (i.e. protein movement between cytoplasm and the nucleus) - examples of which include the translocation of the NF-kappa B proteins into the nucleus. Protein Calculus is particularly useful in transforming collected information from *in vivo* experiments into coded definition of the different types of proteins and complexes of proteins, and establishing positions of the coded objects with respect to the membranes of the system. In addition, the syntax of the calculus allows the use of the operational semantics for calculating future behaviours.

REFERENCES

1. J. A. Bergstra et al. (ed.), Handbook of Process Algebra, Elsevier, 2001
2. L. Cardelli, Abstract Machines of Systems Biology, TCSB, III-3737, 2005
3. L. Cardelli, Brane Calculi - Interactions of Biological Membranes, In Proc. CMSB'04, LNCS 3082, 2005
4. G. Ciobanu and G. Rozenberg Ed., Modelling in Molecular Biology, Springer-Verlag, Berlin, 2004
5. S. Efroni et al., Toward rigorous comprehension of biological complexity: modeling, execution, and visualization of thymic T-cell maturation, Genome Res., vol.13, 2003
6. N. Kam et al., Formal Modeling of C. elegans Development: A Scenario-Based Approach, LNCS-2602, 2003
7. P. Lecca and C. Priami, Cell cycle control in eukaryotes: a BioSpi model, In Proc. BioConcur'03, ENTCS, To appear
8. S. Marcus, Language at the crossroad of Computation and Biology, In G. Paun (ed.), Computing with Biomolecules; Theory and Experiment, Springer, 1998
9. S. Marcus, Membranes versus DNA, Fundamenta Informaticae, 49(1-3), 2002
10. R. Mardare, et. al., Model Checking Biological Systems Described Using Ambient Calculus, In Proc. CMSB'04, LNCS 3082, 2005
11. R. Mardare, C. Priami, Logical Analysis of Biological Systems, Fundamenta Informaticae, 64(1-4), 2005
12. Gh. Păun, Membrane Computing. An Introduction, Springer-Verlag, Berlin, Natural Computing Series, 2002
13. A. Phillips and L. Cardelli, A correct abstract machine for the stochastic pi-calculus, In Proc. BioConcur'04, ENTCS, To appear
14. A. Regev and E. Shapiro, Cells as Computation, Nature, vol.419, 2000
15. BetaWB simulator, http://www.cosbi.eu
16. Cyto-Sim simulator, http://www.cosbi.eu