Bio-Logics

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

Systems Biology studies the relationships and interactions between parts of biological systems (such as gene and protein networks, metabolic pathways, and the immune system) by combining knowledge from molecular biology with high-throughput techniques. The main goal is to comprehend the functionality of a biological system as a whole (with a special interest in emergent properties) and, eventually, to develop a functional model of it.

By abstracting biological systems on the level of their behaviours, we obtain models sharing many characteristics with computational systems. Thus, on all levels of organisation, from the atomic to full organisms, we are faced with concurrency (simultaneous behaviours), event-driven, causal, time-dependent and distributed behaviours. In this context, modelling (e.g., reverse-engineering, simulating and analysing) a biological system is considered in a similar way to the engineering of complex artificial systems [8,11]. The challenge is to construct, *in silico*, reliable *`copies'* of biological systems that allow us to simulate various experiments and can assist us in proposing and verifying hypotheses that might be impossible by conventional means. Recent small-scale modelling efforts in this direction have had encouraging results [30,19,13].

Mathematical models of biological processes

Traditionally, these models involve systems of ordinary differential equations (ODEs) that can successfully handle the dynamics of the systems. But these methods have significant limitations due to the lack of compositionality and systemic view. ODEs model the bio-system as a whole and compute its overall behaviour, but the description of the model cannot be disassembled into its constituent components. Thus, ODE models cannot be easily extended when new information about the system is available. For obtaining reliable models *in silico*, we expect to be able to combine small-scale models of molecular or cellular networks to create 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 eventually be used for designing a model of that molecule; however we want to be able to integrate such small models in order to define larger scenarios such as the behaviours of an organ or of an organism.

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.

Membrane computing was introduced as a new paradigm in computer science *``aiming to abstract computing ideas and models from the structure and the functioning of living cells, as well as from the way the cells are organized in tissues or higher order structures''* [28]. It comes as an extension of *formal language theory* [12], inspired by the intuition of the similarities between Brain-Mind-Computation, Mind-Natural Language-Formal Language, Brain-Neural System-Biological System [22,6]. Membrane systems (also called *P systems*) are computational devices designed for modelling distributed and parallel computational scenarios. A P system is composed of a hierarchical arrangement of membranes, similar in structure to a cell, a tissue or a neural network. A membrane has the role of a separator of the space of computation. In between membranes (in some models also *on* the membranes) there are distributed objects that can interact or move from one membrane to another following some laws (similar to bio-chemical laws). Formally, we work with multisets of symbol-objects from a given alphabet and the *evolution rules* are just rewriting rules also containing membrane-indexed information. One of the main features of the rules is that they can express communication between compartments, as well as communication between the system and its environment. Thus, membrane computing is a computational framework, inspired form biological reality and designed to model compartmentalized phenomena. It has proved useful for applications in computer science and also in modelling biological systems due to the essential role of the compartments in biology [23,21].

Process algebras [5] 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). Recently, the paradigm has been demonstrated to be particularly appropriate for modelling molecular and cellular networks [30,8]. In this approach, a biological system is modelled as a network of modules (subsystems) that run in a decentralised manner, and interact or change their relative positions, generating the global behaviour of the system. Thus, the overall model of the system can be obtained by mixing the models of its components. Further, this paradigm was adapted to include quantitative and stochastic information [29] - essential features of bio-chemical processes - and used together with stochastic simulators 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 way results of expensive or impossible *in vivo* experiments might be derived *in silico*.

Membrane computing and process algebras have opened a new paradigm in modelling biological systems. Extensive research has been done in these fields and some useful tools for simulating such models have been developed, e.g. [33]. But is this modelling paradigm sufficient for analyzing, *in silico*, living systems?

The limitations of a Turing-like test for biological modelling

In 1950 Turing proposed a test for determining whether a computer is intelligent [31]. The test uses a human interrogator, Alice, that interviews a computer and another human, Bob. Alice, Bob and the computer are placed in different rooms and their conversations are mediated by some electronic interfaces, such that Alice does not know in which room Bob is and in which room the computer is. Alice's task is to determine, by questioning the two, which is the human and which is the computer. If, in a given amount of time, she is not able to determine this, then we can say that the computer is intelligent (can mimic human intelligence).

We can consider a similar approach for testing if a computer can mimic life. In such a case the interviewer, Alice, compares a biological system with a computer simulator. She compares the values of some variables which, in the case of the bio-system, are measured, while in the case of the simulator are generated by some algorithms.

But what does it mean, in these conditions, to compare? Suppose that Alice finds some differences. Is she able to indicate in which room the computer is? Observe that for the case of artificial intelligence the situation is different: even if Alice is not able to provide a formal definition of intelligence, she can recognize the absence of it! But in the case of simulating life, Alice can only test the two entities against her knowledge. Alice has to have some a priori knowledge about the biological property she will test (e.g. she knows some bio-chemical components of the living system and some laws of chemistry) and she can test if both entities behave as expected. Only if one of the two behave differently (w.r.t. her knowledge), can Alice decide which is the computer (and only if she knows that the living system has the tested property).

As underlined in [16], such a Turing test is sensitive to the knowledge of the interviewer: while for Alice the simulator is successful, for another interviewer, Eve, it may not be, since Eve knows more than Alice (and can perform a more complex test). Hence, we cannot speak about an ultimate test as we cannot have complete knowledge about a biological system.

Consequently, before proceeding with the simulations of the mathematical models we developed for biological systems, we need to certify our models as appropriate for checking the properties we are interested in. This means that we have to prove, formally, that we have sufficient information available for fulfilling our task. The modelling efforts have to be synchronised with the development of model validation techniques: a reliable scientific model has to be validated or falsified by additional analysis.

Logics and model verification techniques

A model of a computational system can be validated or falsified by prediction and retrodiction. *Model-checking techniques* are powerful tools used for, in addition to stochastic simulations, validating/falsifying models, assisting in hypothesis forming and, eventually, predicting the behaviours of biological systems. These techniques are designed to verify if a model satisfies a given specification and are widely used for applications in Computer Science. They are supported by *model checkers* - software tools that take as input a model and a query (regarding a property of the model) and calculate, using an efficient state exploration engine, if the property holds for every possible run of the model. Here is the first difference with respect to stochastic simulators: a simulator produces only one computational trace for each run, so any analysis relies on this possible run; consequently, it might be misleading as it can fail to expose an important event if a relevant run is not generated. On the other hand, a model checker makes an exhaustive search in the state space of the model searching for properties on all possible runs.

A more powerful technique is *probabilistic model checking* [32,20], which has been specifically designed for investigating probabilistic and stochastic models that arise from biochemical reactions. The query language allows probabilistic terms and, consequently, we can ask *`what is the probability of reaching a certain state'* or *`what is the probability of reaching a state in certain given (probabilistic) preconditions'*. With respect to stochastic simulations, this method allows the analysis of the range of behaviours of the model (the best/worst cases, as well as expectations).

Model-checking techniques encounter difficulties generated by state space explosion. This problem is dependent on the logic used for specifying the properties of the models (the formal query language). If the logics are too expressive, they might be undecidable (no algorithms for performing such analysis can be constructed) or computationally expensive (even if algorithms could be constructed, the cost of computations is too high). On the other hand, logics that are too simple support very efficient algorithms for model checking, e.g. temporal logics, but they might not be sufficiently expressive to specify important properties.

Outside the complexity aspects, the spectrum of problems raised by biological modelling opens up the prospect of studying new interesting biologically-inspired logics. For biological systems important properties refer to time-dependent evolution and causality, to the structure of the modules (subsystems) of the system, to simultaneous behaviours of subsystems, communication-like interaction between modules, involve membranes and compartments and focus on the knowledge of the agents that observe the systems, as highlighted in the previous section.

Temporal logics [14] are the most used logics in model checking, efficient algorithms and tools having already been developed for them. In computer science these are widely used in hardware and software engineering. These techniques were also proposed, with encouraging results, for the case of validating models of biological systems [25,27]. Temporal logics are devised with operators for expressing and quantifying on (future/past) states or traces of behaviour and computation (`*somewhere in the past*', `*at the next state*', `*eventually in the future*', `*until*', etc.) specifying complex properties, such as `*the protein always eventually degrades*', `*reaction X is not activated until promoter x occurs*', etc.

In addition, quantitative and stochastic or probabilistic information has been integrated [1], generating queries like `*what is the probability that the protein will degrade within t hours?*', `*what is the expected number of complexation reactions before relocation occurs?*', `*what is the probability of reaching the state s?*', or `*what is the probability of having, in the state s, a concentration of the substance x higher than c?*', etc.

Dynamic logics [17] are logics designed for specifying dynamic systems. They have names for ``*programs*" or ``*actions*" that a system might take, and ways to combine them. In this case we have modalities indexed on a signature that is interpreted as the set of programs that coordinate the evolution of the system. A dynamic modality, $[\pi]\varphi$, captures the weakest precondition of such a program w.r.t. a given post-specification, φ, and the accessibility relations are interpreted as transitions induced by programs. Features of these logics are relevant for specifying the evolution of living systems [26]. In this light, the programs or the actions of the system represent basic bio-chemical interactions.

Spatial logics. Temporal and dynamic logics can only specify global properties of the states of the system and their evolution over time. These are useful in verifying properties concerning molecular and bio-chemical aspects of systems. Still, many relevant properties that can be used for validating models concern the structure of the system. Biological systems involve compartmentalization, which plays a major role in the functioning of living systems [23,21,8]. These systems are not just unstructured, heterogeneous chemical soups, but spatially distributed hierarchies of bio-compartments organizing 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. 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.

Thus, many relevant properties of biological systems cannot be expressed unless referring to compartments. We want to be able to query about properties such as ``*protein x eventually disintegrates inside this cell*", ``*inside this membrane there are two vesicles*", ``*any infected cell contains protein x with a concentration higher than c*", ``*if proteins x and y eventually form a complex inside this membrane, then the reaction* $X+Y\rightarrow Z$ *is enabled*" All these properties refer to some spatial peculiarities of the system: the presence/absence of some entities in clearly specified membranes, *somewhere/everywhere* spatial quantifiers, the number of copies of some entities sharing the same location, etc. They do not refer to the whole system in a given state, but to some subsystems (well-defined areas). For this purpose we need logics enriched with ``*spatial*" operators that will allow us to express and combine properties of modules having different locations in the topology of the analysed system.

The problem of designing ``*spatial-temporal*" *logics* is not new in Computer Science. It arose from the necessity to specify important properties of computer networks relevant to problems of security. Spatial Logics [7,9] propose a special operator to mark the presence of a membrane, other operators to express the coexistence of two entities sharing the same compartment, etc. Thus, in the case of a biological system S we could assert the property $[\varphi|\psi]$ if and only if it has a main membrane that encloses two identifiable subsystems S',S" (formally S=[S'|S"]), where S' has the property φ while S" has the property ψ. Extensions of spatial logics have been proposed for studying biological systems both for the membrane computing paradigm [10] and for process algebras [24].

Epistemic/doxastic logics. Since the properties we want to encode and check against the mathematical models of living systems are sensitive to the knowledge of the observer, as underlined before, it is useful in some cases to have a formalism able to relate the knowledge of an agent observing a part of a biological system (having partial information about it) with the property that has to be checked. Such a logic can be used for proving not only if the system has a property, but also if the information available about the system suffices or not for saying that the system has the property. This formalism can prove that more information is needed in order to check a property. The peculiarities of such model verification techniques can also identify parts of the model where deeper analysis is needed, hence it can be used for organizing *in vivo* research.

Epistemic/doxastic logics [18,15] are logics that formalize the epistemic notions of *knowledge*, or *belief*, possessed by an agent or a group of agents in a computational environment. They are devised with *``epistemic'' modalities* like $K_A\varphi$ (A knows that $\varphi$) or $\#_A\varphi$ (A justifiably believes that $\varphi$) defined for any agent A. In the models of these logics each basic modality is associated to a binary *``accessibility'' relation* interpreted as an *``indistinguishability''* relation for each agent A. It expresses the agent's uncertainty about the current state: if the current state of the system is s, A thinks that any of the alternatives s' to s w.r.t. the indistinguishability relation, may be the current state. Further, different generalizations of logics combining dynamic and epistemic operators have been studied and applied in computer science [2,3,4].

Bio-Logics. By combining the epistemic approach with the spatial and temporal features, a class of new formalisms have been proposed especially designed for specifying the models of biological systems [10,24]. These logics combine the high level of expressivity inherited from the original formalisms with a low complexity, proposing decidable and completely axiomatized systems. These features are the sufficient ingredients needed for developing useful tools (model checkers, theorem provers) to be used, in addition to the simulators, for analysing the behavior of living systems.

Bibliography

[1] A. Aziz et al., Verifying continuous-time Markov chains, In Proc. CAV'96, LNCS 1102, 1996
[2] A. Baltag, A Logic for Suspicious Players: Epistemic Actions and Belief Updates in Games. Bulletin Of Economic Research vol. 54(1), 2002
[3] A. Baltag and L.S. Moss. Logics for Epistemic Programs, Synthese vol. 139(2), 2004
[4] A. Baltag and L.S. Moss and S. Solecki. The Logic of Public Announcements. Common Knowledge and Private Suspicions, CWI Tech. Rep. SEN-R9922, 1999
[5] J. A. Bergstra et al. (ed.), Handbook of Process Algebra, Elsevier, 2001
[6] C. Calude, S. Marcus, G. Paun, The Universal Grammar as a hypothetical brain, Revue Roumaine de Linguistique 24, 1979
[7] L. Caires, L. Cardelli, A Spatial Logic for Concurrency (Part I), Inf. and Comp., vol.186/2, 2003
[8] L. Cardelli, Abstract Machines of Systems Biology, TCSB, III-3737, 2005
[9] L. Cardelli, A. D. Gordon, Anytime, Anywhere: Modal Logics for Mobile Ambients, In Proc. 27th ACM Symp. on Princ. of Prog. Lang., 2000
[10] M. Cavaliere, R. Mardare, Partial Knowledge in Membrane Systems: A Logical Approach, In Proc. WMC7, LNCS 4361, 2006
[11] G. Ciobanu and G. Rozenberg Ed., Modelling in Molecular Biology, Springer-Verlag, Berlin, 2004
[12] J. Dassow and Gh. Păun, Regulated Rewriting in Formal Language Theory, Springer-Verlag, Berlin, 1989
[13] S. Efroni et al., Toward rigorous comprehension of biological complexity: modeling, execution, and visualization of thymic T-cell maturation, Genome Res., vol.13, 2003
[14] E.A. Emerson, Temporal and modal logic, Handbook of Theoretical Computer Science, MIT Press, 1990
[15] R. Fagin et al., Reasoning about Knowledge, MIT Press, 1995
[16] D. Harel, A Turing-Like Test for Biological Modeling, Nature Biotechnology, vol.23, 2005
[17] D. Harel et al., Dynamic Logic, MIT Press, 2000
[18] J. Hintikka: Knowledge and Belief, Ithaca, N.Y.: Cornell University Press, 1962

[19] N. Kam et al., Formal Modeling of C. elegans Development: A Scenario-Based Approach, LNCS-2602, 2003
[20] M. Kwiatkowska, Model Checking for Probability and Time: From Theory to Practice, In Proc. LICS’03, 2003
[21] S. Marcus, Bridging P systems and Genomics; A preliminary approach, In G. Paun et. al. (ed.), Membrane Computing, LNCS 2597, 2003
[22] S. Marcus, Language at the crossroad of Computation and Biology, In G. Paun (ed.), Computing with Biomolecules; Theory and Experiment, Springer, 1998
[23] S. Marcus, Membranes versus DNA, Fundamenta Informaticae, 49(1-3), 2002
[24] R. Mardare, Logical analysis of complex systems: Dynamic Epistemic Spatial Logics, PhD. thesis, DIT, University of Trento, Italy, 2006
[25] R. Mardare, et. al., Model Checking Biological Systems Described Using Ambient Calculus, In Proc. CMSB'04, LNCS 3082, 2005
[26] R. Mardare, C. Priami. Decidable extensions of Hennessy-Milner Logic, In Proc. FORTE'06, LNCS 4229, 2006
[27] R. Mardare, C. Priami, Logical Analysis of Biological Systems, Fundamenta Informaticae, 64(1-4), 2005
[28] Gh. Păun, Membrane Computing. An Introduction, Springer-Verlag, Berlin, Natural Computing Series, 2002
[29] C. Priami, Stochastic pi-Calculus, Comput. J., vol. 38(7), 1995
[30] A. Regev and E. Shapiro, Cells as Computation, Nature, vol.419, 2002
[31] A. Turing, Computing machinery and intelligence, Mind, vol. LIX-236, 1950
[32] M. Vardi, Automatic verification of probabilistic concurrent finite state programs, In Proc. FOCS’85, 1985
[33]Cyto-Sim simulator, available from *http://www.cosbi.eu/Rpty_Soft_CytoSim.php*