PLOS | COMPUTATIONAL BIOLOGY

REVIEW

Computational Modeling, Formal Analysis, and Tools for Systems Biology

Ezio Bartocci[1]*, **Pietro Lió**[2]

1 Faculty of Informatics, Technische Universität Wien, Vienna, Austria, **2** Computer Laboratory, University of Cambridge, Cambridge, United Kingdom

* ezio.bartocci@tuwien.ac.at

Abstract

As the amount of biological data in the public domain grows, so does the range of modeling and analysis techniques employed in systems biology. In recent years, a number of theoretical computer science developments have enabled modeling methodology to keep pace. The growing interest in systems biology in executable models and their analysis has necessitated the borrowing of terms and methods from computer science, such as formal analysis, model checking, static analysis, and runtime verification. Here, we discuss the most important and exciting computational methods and tools currently available to systems biologists. We believe that a deeper understanding of the concepts and theory highlighted in this review will produce better software practice, improved investigation of complex biological processes, and even new ideas and better feedback into computer science.

Citation: Bartocci E, Lió P (2016) Computational Modeling, Formal Analysis, and Tools for Systems Biology. PLoS Comput Biol 12(1): e1004591. doi:10.1371/journal.pcbi.1004591

Editor: Marta Zofia Kwiatkowska, University of Oxford, UNITED KINGDOM

Published: January 21, 2016

Funding: Ezio Bartocci was partially supported by the Austrian National Research Network S 11405-N23 (RiSE/SHiNE) of the Austrian Science Fund (FWF). Pietro Liò was partially supported by the FP7-305280 MIMOmics European Collaborative Project, as part of the HEALTH-2012- INNOVATION scheme. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

Introduction

Computer science is currently central to a huge range of scientific areas. In its early days, its task was simply to translate a model expressed in a mathematical language into a computer program simulating that model. The field has progressed since then, yielding new domain-specific programming languages that are able to directly model a physical process.

In both cases, the computational implementation is perceived as a necessary methodological step for systems biologists, because the simple execution of a program provides an in silico numerical evaluation of hypotheses, avoiding the use of complex analytical methods and considerably reducing the costs of expensive in vivo or in vitro experiments.

Recently, the dichotomies between mathematical and computational models have also been subject to a debate (see also [1,2]) about whether or not the difference between them arises primarily from their ability to be directly executed [1] or from the different purposes and approaches adopted by scientists [2].

The novel concepts and principles (and well-designed tools) developed within the computer science community are accompanied by a domain-specific terminology (for example, executable models, expressivity, abstraction, model checking, reachability analysis, formal verification, and static analysis) that is scarcely known in other scientific communities such as systems biology. The introduction and assimilation of these concepts in fields other than computer science may back-propagate new ideas to computer scientists.

Recent works have discussed in detail how the methods borrowed from computer science have already benefited and can further benefit various problems in biology [1–6]. With respect

to these previous research and review papers, our effort focuses on discussing how the use of newly developed tools could facilitate the understanding of the concepts, practice, and terminology acquisition in the current language of systems biologists. Furthermore, this review will take the further step of communicating the usefulness of using a temporal-logic framework for systems biologists who are looking beyond correlation toward event causality or patterns occurring in biological signals.

A nonexhaustive literature review, which is nonetheless an aid to understanding current progress in the field, is then presented.

Computational Modeling

In the last decade, the area of systems biology has benefited greatly from computational models and techniques previously adopted only in computer science to assess the correctness and safety of a program. In this context, the design of a biological model becomes equivalent to developing a computer program. Various programming languages, often biological domain-specific, provide a means of describing the instruction sequence specifying the control flow of a biological process.

The syntax of the language defines the ways the symbols may be combined to create well-formed sentences or instructions. This specification is often represented in a textual way (i.e., a process calculus, rule-based system), but in several cases (i.e., Petri nets, statecharts, etc.), a graphical representation is also available. This helps the user to visualize the process with diagrams displaying the flow of the species in the reactions or the change in the internal states. The semantics reveals the meaning of the syntactically valid instructions by describing the behavior of the model and how it should be executed by the computer. It is also possible that a model specified using a particular language syntax may be executed using different language semantics: for example, a set of chemical reactions rules can be executed using a continuous semantics (ordinary differential equations [ODEs] on molecular concentrations) or a stochastic semantics (on the number of molecules), depending on the level of approximation and/or complexity [7] that we may want to achieve. For example, COPASI [8,9] is a tool for numerical simulation and analysis of biochemical networks for both their continuous and stochastic dynamics.

In the following, we discuss the key features of the main computational modeling approaches that have fallen on fertile ground in systems biology. Fig 1 provides simple examples, inspired by case studies reported in the literature, of the modeling approaches considered.

Process Algebras

In recent years, computer scientists have intensively investigated the use of process algebras (PAs) for the modeling and the analysis of biological systems [10–14]. The expressive power of PAs (see Fig 1: first row, first column) allows formal specification, without any ambiguity about the interactions, communications, and synchronizations between a collection of concurrent processes (also called agents). The reason for the interest in PAs for systems biology is that biological systems can be considered as concurrent reactive systems, where biological species can be modeled as processes interacting with each other. Another important feature of PAs in the modeling of complex (often multiscale) biological systems is their compositionality, which offers the possibility of defining the whole system, starting from the specification of its subcomponents. Furthermore, PAs usually permit formal reasoning about equivalences between processes. The leading examples of PAs in computational systems biology include Beta-Binders/BlenX [12], SPiM [15–17], Bio-PEPA [13], sCCP [18], and BioShape [19,20]. PA specifications are usually employed as intermediate models that are then executed or translated in other

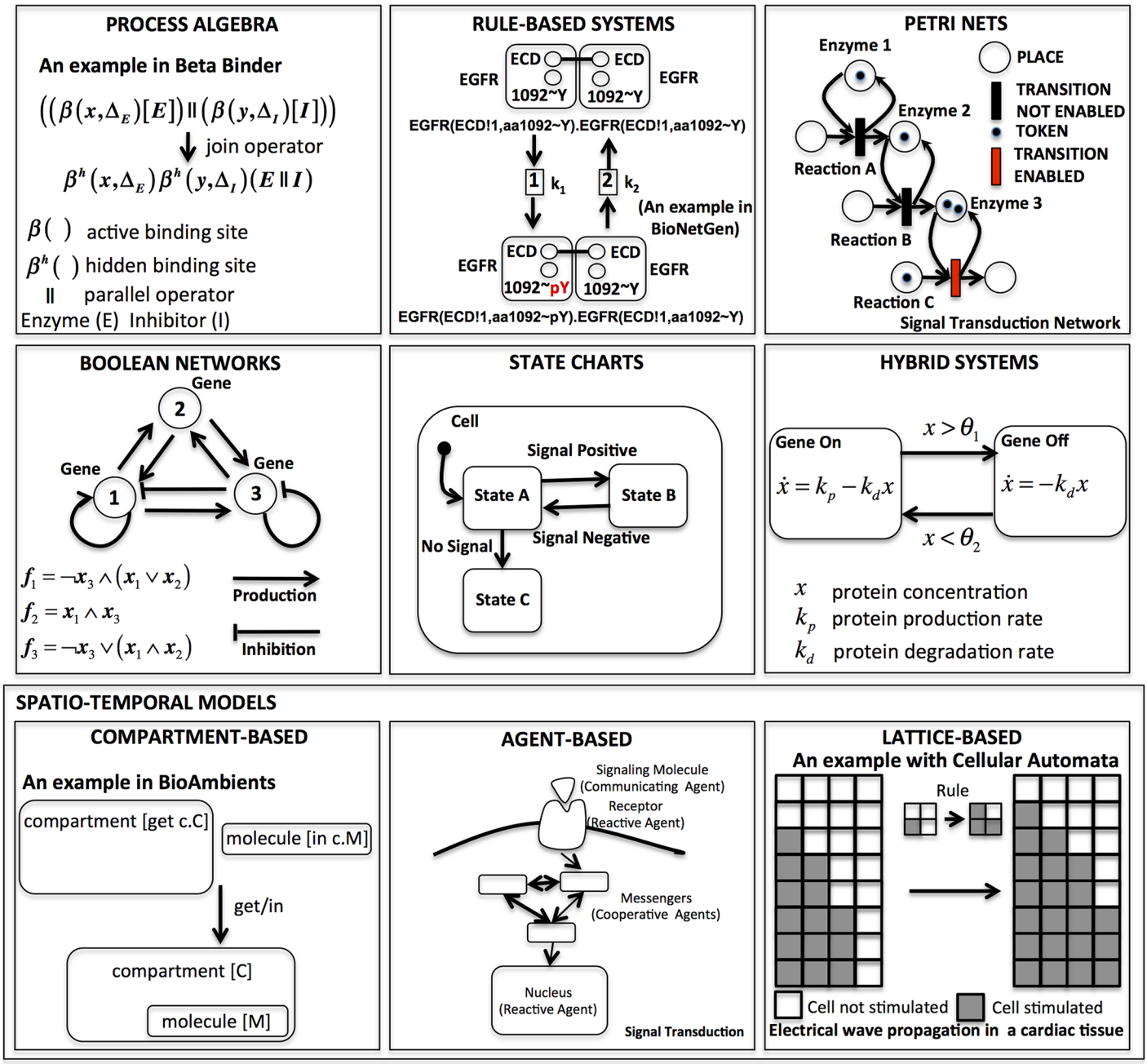

Fig 1. Most relevant examples of computational modeling approaches introduced with toy examples. Related tools are listed in Table 1. References for the examples are as follows: process algebras [12], compartment-based systems [21], rule-based systems [22], statecharts [23], hybrid systems [24], Boolean networks [25], Petri nets [26], agent-based models [27], lattice-based models [28].

doi:10.1371/journal.pcbi.1004591.g001

computational models using different semantics: continuous differential (ODEs), stochastic (Continuous Time Markov Chains), or abstract (transitions systems).

Rule-based Systems

Rule-based modeling (see Fig 1: first row, second column) has gained a lot of attention among biologists, because its notation is very similar to the chemical reaction representation used in

systems biology to model biochemical interactions between molecular species. Consider, for example, the classical enzymatic reaction where an enzyme (E) binds a substrate (S) and produces a product (P) by releasing the enzyme (E). This can be expressed in a very compact and concise description by using two simple rules:

1. $E + S \leftrightarrows ES$
2. $ES \rightarrow E + P$

One important feature of this modeling technique is that rules, unlike equations, are independent units, so they can be easily changed or modified. Furthermore, the simple syntax of rule-based models can be stored in a file as a human-readable text and can be edited and visualized using a graph representation. This makes rule-based modeling friendly for users without specialized mathematical or computer science skills. Rule-based models can then be translated, using different semantics, to generate other computational models in order to provide a quantitative (i.e., the amount of a species in time) [22,29] prediction or a qualitative (i.e., where time is abstracted away) understanding of the system's emergent behavior. For these reasons, many rule-based modeling languages and tools, such as BIOCHAM [30,31], Kappa [32], BioNetGen [22,33], have become very popular among systems biologists in the recent years and have been intensively utilized in concrete case studies [34–36]. We refer to [3] for a more exhaustive review of rule-based modeling.

Petri Nets

A Petri net (see Fig 1: first row, third column) is a directed graph whose vertices can be divided into two disjointed sets (bipartite graph), a set of nodes called "transitions" (meaning events that may occur, i.e., reactions), graphically represented by bars, and a set of nodes called "places" (meaning the conditions for a reaction to occur, such as the presence of a molecule), graphically represented by circles. Arrows interconnect these nodes, showing the direction of flow, with this main rule: a place node can be connected only to a transition node and vice versa. The data (i.e., species) are generally represented as "tokens", signified by black marks. The tokens are consumed from the input places through the transitions and then created in the output places. A transition "fires" whenever it is enabled by the presence of some tokens in one of the places directly connected to it. A concurrent semantics specifies the evolution in time of the token distribution. This modeling framework was introduced by Carl Adam Petri in 1962 with the purpose of describing chemical processes [37], but then was also intensively employed in computer science to specify and analyze concurrent and distributed systems. It is not surprising that this intuitive and graphical modeling style is popular among computational systems biologists [26,38–40] to describe biochemical reaction systems, where the tokens are interpreted as single molecules of the species involved. The Petri net formalism, as shown also in [41], provides a natural framework in which both qualitative (given by the static structural topology of the Petri nets) and quantitative (given by the time evolution of the token distribution) analysis are tightly integrated. Important tools for Petri nets used in computational biology are Snoopy [26], MARCIE [42], GreatSPN [43,44], and Pathway Logic Assistant [45,46].

Boolean/Qualitative Networks

Boolean networks (see Fig 1: second row, first column) were first introduced by Kauffman [25] and then by Thomas [47,48]. They are often used to approximate the dynamics of genetic regulatory networks by considering genes either activated (true state) or deactivated (false state). A Boolean network is defined in terms of Boolean variables, each one updated by a Boolean function that determines the next truth value state given the inputs from a subset of those variables.

This modeling technique, even though it usually introduces a coarse approximation by neglecting intermediate states, is widely employed to analyze the robustness and stability of genetic regulatory networks. For instance, by generating initial random configurations, it is possible, by executing this model, to detect singleton attractors (also called fixed points), where the system is stable. Relevant tools for Boolean networks analysis in systems biology are GINsim [49–51], BoolNet [52], and BNS [53,54]. Qualitative networks, introduced recently in [55], extend the Boolean network, allowing its elements to assume a finite number of possible values. This feature provides biologists with more flexibility than just Boolean values and enhances the variety of behaviors that it is possible to model with this formalism. The tool for modeling and analysis of qualitative networks is Bio Model Analyzer (BMA) [56].

Statecharts

Another natural way to model the dynamics of a biological system is to specify the sequence of the states characterizing its behavior [23]. For example, when a phosphate group is added to some proteins, their functional behavior can change to a phosphorylated state, enabling other potential protein—protein interactions. A system remains in a state until the occurrence of some event (e.g., the activation or inhibition of a gene) moves its internal behavior from one state to another. This characteristic makes a biological system a multiscale reactive system in which event-driven concurrent interactions, occurring at different levels (molecular, cellular, tissue, organ, or population level) or between levels and with different timing and order, determine its emergent behavior. The statecharts notation (see Fig 1: second row, second column) is then a suitable formalism to present, in a graphical representation, the interdependence among the states of a reactive system. Several slightly different versions of these state diagrams have been proposed with different semantics.

The statecharts introduced by Harel [57] have been the most popular among biologists because they offer appropriate constructs (hierarchy of states with transitions, events, conditions, orthogonal regions, etc.) to handle the complexity of modeling biological systems. The classic statecharts notation, in fact, would require one to specify any possible combination of parameters as a distinct state, leading to an explosion of the number of states. Among the tools for statecharts, the most relevant in systems biology is IBM Rational Rhapsody [58,59].

Hybrid Systems

Hybrid systems [60] (see Fig 1: second row, third column) extend the state-based discrete representation previously mentioned with a continuous dynamics (generally ODEs) in each state (or mode). Hybrid modeling techniques [24] are gaining more and more attention in systems biology [61] for their ability to capture the behavior of biological systems that exhibit clear switching characteristics. In particular, sigmoidal switches occur everywhere in biological models: molecular (an example is the sigmoidal behavior exhibited by Hill-type kinetics), cellular, tissue, organ, and population models. Hybrid modeling is generally suitable to combine qualitative (given by the discrete state) and quantitative (given by the continuous dynamics) information [62]. In the last decade, several hybrid system identification (hybridization) methods have been proposed in the literature [63–66] to approximate complex nonlinear dynamics with piecewise-linear [67,68] or piecewise-multi-affine functions [66,69–71], making such models amenable to formal analysis [60,66–71] and improving large-scale simulation of multicellular ensembles [72–74]. It is noteworthy that widely used mathematical platforms, such as Matlab [75] and Simulink [76,77], enable the user to model and simulate hybrid systems. Other relevant tools for hybrid systems modeling in biology are Rovergene [71], BioDivine [69,78,79], Breach [80,81], dReach [82,83], and S-TaLiRo [84].

Spatio-temporal Models

Continuous state deterministic spatiotemporal systems (see Fig 1: third row) are generally formulated in terms of "reaction-diffusion" systems taking the form of semilinear parabolic partial differential equations (PDE). In the discrete state setting, compartment-based models (i.e., membrane computing), agent-based models, and lattice-based computational models (i.e., cellular automata, cellular Potts) have been employed to simulate the collective behavior of cellular structures. All of these models display a wide range of behaviors emerging from local and nonlocal interactions such as traveling waves (i.e., cardiac tissue) [85], Turing patterning [86,87], and spirals [88,89].

Compartment-based models. Biological systems are generally organized in compartments (i.e., cell membrane, cell nucleus, organelle), exchanging molecules between them according to certain rules. Compartment-based models (see Fig 1: third row, first column) are specialized to capture several biological characteristics, such as the dynamic rearrangements of the compartments (a typical behavior observed in the mitochondria) and the transport of molecules between them.

The study of the membranes separating the compartments has also initiated a new area within computer science called membrane computing, which aims to discover new bio-inspired computational paradigms, such as the P Systems [90]. However, these models are more suitable for the theory of computation than for modeling in systems biology.

Another relevant modeling framework is BioAmbients [91], a process algebra enriched with special operators able to specify merging, splitting, and communication between compartments. BAM [92,93] is a tool for executing stochastic BioAmbients. BioAmbients evolved into Brane Calculus [94], which offers a specially designed language to describe the dynamic behavior of membranes. Whereas, in BioAmbients, the ambient (i.e., compartment) plays an active role in dictating which processes may enter or exit from it, Brane Calculus offers a different perspective, in which the membranes have the control and play the role of coordinators. To the best of our knowledge, there is not yet an implementation available for it.

Agent-based models. Agent-based models [95,96] (see Fig 1: third row, second column) consider a collection of autonomous decision-making entities, called agents, which individually sense the environment and make decisions on the basis of a set of rules. Although, at the simplest level, an agent-based model consists of a system of agents and the relationships between them, it can still exhibit complex behavior patterns in terms of changes and adaptation in response to environmental challenges or to neighboring agent behaviors (for example, competition or collaboration).

Because all individuals in a population are explicitly represented, they can have unique histories and behaviors. More complex agent-based models sometimes incorporate sophisticated learning and adaptation rules based on neural networks, evolutionary algorithms, or other techniques. The single-cell-based models represent one of the most promising aspects, in which agents have many cellular functional and structural features and behavior, inching toward reality and enabling the detection of phenomena at different intermediate scales of biosystems.

Cell-based models can express important behavioral characteristics of a cell, such as the dynamics of its replication and information on each stage of its development (i.e., cell geometry, size, and mechanical properties).

A single-cell-based model should be able to understand how stage-dependent cell—cell interactions at microscopic scale will lead to cell—tissue interactions and stage heterogeneity at mesoscopic level and mechanical properties of the tissue at macroscopic level. Models could be implemented using FLAME [27,97] and REPAST [28,98], for example.

Lattice-based models. A lattice (see Fig 1: third row, third column), which defines a regular repeated graph, formed by identical n-dimensional closed grid sites and characterized by periodic or fixed boundary conditions in each direction, is particularly suited for systems description of interconnected processes at the molecular, cellular, and the tissue or organ level. These natural levels can approximately be connected to a microscopic (molecule motion and interactions), mesoscopic (cell division and motion, cell—cell interactions, cell-matrix), and macroscopic scale (tissue and organ mechanical properties), respectively.

Cellular automata [99] are discrete dynamic systems—discrete in space, time, and state. Cellular pattern formation can be seen as arising from short-range (such as adhesive forces and cell—cell signaling) and long-range (such as mechanical stress fields or diffusing chemicals) interactions. A Bethe lattice (or Cayley tree) [100] is a hierarchically ordered, cycle-free network without ends that has been applied to immunological (idiotypic) networks.

In multiscale lattice-based models, we can observe what happens at almost all scales, from the whole organism down to the molecular level; however, putting things together in order to obtain real understanding is much more difficult and involves scaling up and homogenization of models across multiple spatial scales and related asymptotic techniques for the analysis of multiple time scales. This problem could be overcome by using energetic considerations, such as in the cellular Potts model (also termed the Glazier Graner-Hogeweg model), which are based on the stochastic Monte Carlo method on a regular lattice [101,102]. The objects, either discrete generalized cells (unicellular organisms, clusters of cells, individual cells) or continuous fields (such as gradients of nutrients or small molecules), are associated with an energy description of processes such as cell—cell adhesion or cell—nutrient interaction. Lattice rearrangements, which simulate the evolution of the system, are driven by the energy minimization of a Hamiltonian function.

A very general and flexible framework for Potts model development is CompuCell3D [103,104], which has been used to model a variety of anatomical and pathological conditions at cell, tissue, and organ levels. This framework succeeds in combining both a rigorous energetic and mechanical treatment of the process with an intuitive and insightful biological description. There is growing interest in network ensembles approaches. Multilayer networks and, in particular, multiplex networks (in which different networks share the same nodes) could be analyzed using network entropies to evaluate and quantify the correlations between interdependent networks. For example, in biological systems, gene, protein, and metabolite networks have strong correlations and interdependencies that cannot be fully pictured in terms of single graphs [105].

Formal Analysis

The modeling languages presented in the previous section play a key role in supporting the rigorous specification of the mechanisms observed experimentally, helping scientists in the formulation of new hypotheses. Once a model is constructed, a suitable tool can parse the syntax of its specification and interpret it according to the semantics of the chosen modeling language. A model can also undergo a process of compilation that automatically translates it into a computer program simulating the biological process under investigation. The generated program can be used to predict the emergent behavior of a system with certain initial conditions. This contributes to the testing procedure and to reducing the number of costly experiments, concentrating all efforts and resources only on those that promise to reveal novel, interesting mechanisms.

Another advantage is the possibility of inheriting several methods and tools that are commonly developed and employed within the computer-aided verification community to

formally check the correctness of a program's behavior. In the context of systems biology, these methods are becoming very useful for reasoning and analyzing models, validating new experimental results, automatically checking behaviors of interest, and identifying the inputs or parameters of the system enforcing a desired behavior.

The formal verification of a program consists of proving that its execution satisfies a given specification of the possible behaviors it should display. In the following, we will first present some logic-based languages used to specify temporal behavioral properties rigorously and concisely.

These languages, first developed within computer science, are now also employed in several case studies of systems biology to model recurrent patterns in biological signals or simply the order of relevant biological events. We will then discuss three well-established formal verification techniques designed first to verify programs and now widely employed to analyze biological models: model checking [6,70,71,106–110], runtime verification [80,111–116], and static analysis [93,117–119].

Temporal Logics

Temporal logics [120–123] are very concise languages for rigorously specifying the occurrence of specific temporal behaviors. One of the most popular temporal logics is Linear Temporal Logic (LTL), introduced by Pnueli in 1977 [120] to reason about the order of events occurring during the execution of a program. The LTL syntax is given by the grammar shown in Fig 2a. The basic proposition p indicates a Boolean value that may express the relationship between a state variable of the system and a value for a particular time instant. For example, we can specify that the concentration of the specie x_1 is greater than or equal to a certain threshold r ($x_1 \geq r$) or that a specific biological event e (e.g., phosphorylation) should occur. More complex logical formulas can be obtained by combining propositions using logical operators such as or ($\vee$) and not ($\neg$). The other classical logical operators such as and ($\wedge$) and implication ($\rightarrow$) can be derived by combining the previous two, as shown in Fig 2b.

The LTL syntax also includes two temporal operators: the next ($\circ$) operator, which means that a formula φ should hold in the next step (see Fig 2c), and the until ($\mathcal{U}$) operator, which requires a formula φ_1 to hold until a formula φ_2 becomes true (see Fig 2d).

From the until operator, it is possible to derive other very suitable temporal operators: the eventually ($\diamond$) operator specifies that a formula φ will finally become true at some point (see Fig 2e), and the always ($\square$) operator states that a formula φ should remain true forever (see Fig 2f). The combination and the nesting of the basic propositions with the logical and temporal operators allow the specification of several different types of temporal behaviors.

The most common temporal patterns are:

1. Reachability properties, in which an event will finally happen. For example, we can express the property "the event of protein A production (event $A_{\uparrow}$) will finally occur" with the LTL formula $\varphi = \diamond A_{\uparrow}$. This specification does not guarantee that the same event will happen again after it has occurred.

2. Liveness properties, in which an event will always finally happen. This specification guarantees that the same event will also happen again after it has occurred. For example, the property "always the event of the degradation of protein A (event $A_{\downarrow}$) implies eventually the activation of gene B (event $B_{\uparrow}$)" [124] can be specified with the LTL formula $\varphi = \square(A_{\downarrow} \rightarrow \diamond B_{\uparrow})$.

3. Safety/invariant properties, in which a system will always satisfy a certain requirement. For example, the property "the number of the osteoclasts, the cells degrading/digesting the bone

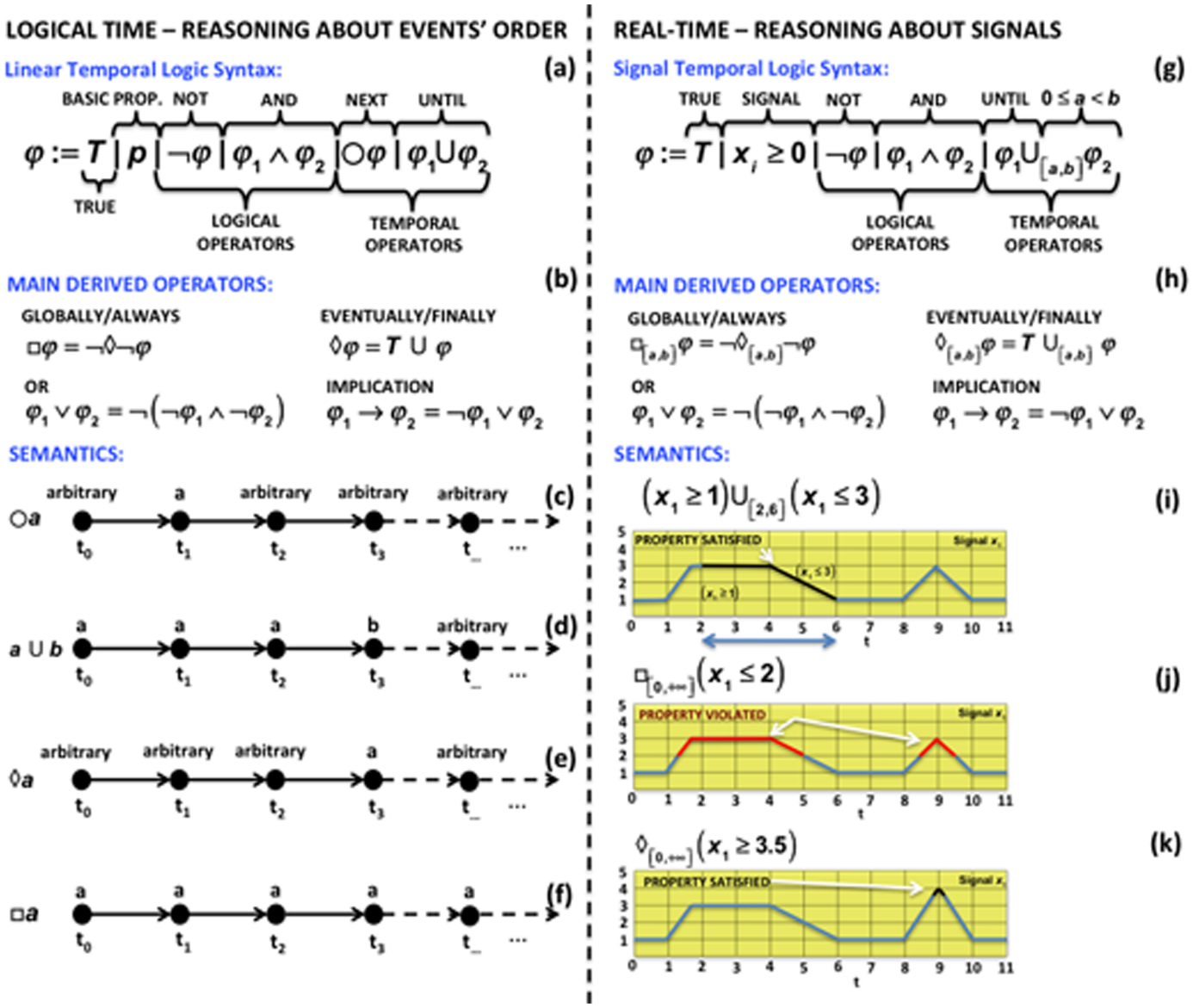

Fig 2. Examples of temporal logics. Comparison between the main features of the LTL (left) and Signal Temporal Logic (STL) (right) in terms of syntax (top), operators (middle), and semantics (bottom); the black circles represents a propositional state, and the arrows represent the next step in time.

doi:10.1371/journal.pcbi.1004591.g002

matrix, x_{oc} during bone remodeling will be always less than a particular concentration c" [107] can be specified with the LTL formula $\varphi = \Box p$ with $p = (x_{oc} \leq c)$.

4. Stability properties, special cases of liveness properties, in which eventually an invariant property will hold. For example, the property "finally the skin cell proliferation x_c will reach a stable level l" [125] can be expressed with the LTL formula $\varphi = \Diamond\Box p$ with $p = |x_c - l| \leq \varepsilon$.

5. Oscillatory properties. For example, the property "the concentration of a protein x_p is oscillating between two levels t_a, t_b with $t_a < t_b$" can be written as the LTL formula $\varphi = \Box((p_1 \rightarrow \Diamond p_2) \wedge (p_2 \rightarrow \Diamond p_1))$, with $p_1 = x_p \leq t_a$ and $p_2 = x_p \geq t_b$

LTL operates on a single path of the model execution, and a temporal property can be formulated only for one possible trajectory of the system. Other temporal logics such as computational tree logic (CTL) [126] and CTL* [127] have in their syntax special quantifiers that enable the specification of properties over all the possible trajectories or branches in time: universal quantifier ($\forall$) specifies that a nested formula should be true for all the possible trajectories, while the existential quantifier ($\exists$) requires the formula to hold in at least one of the possible trajectories.

All the aforementioned temporal logics consider only the temporal order of the events and not the actual time at which they really occur. For example, it is not possible to specify that a formula should hold after two units of time and before three and a half units of time. Even if we decide to discretize the time, recording all the events at each time step, the syntax of these logics is not equipped to deal directly with the specification of real-time intervals.

Real-time temporal logics [128–131] overcome these limits by using a continuous time semantics and embedding a time interval in the until temporal operator.

The Signal Temporal Logic (STL) [131,132] is an example of a real-time temporal logic suitable for many biological case studies [115,116,133–135]. STL extends LTL with the continuous time semantics and with predicates over real variables (see Fig 2g and 2h).

Fig 2i shows how the until operator of the STL syntax is enriched with the possibility of specifying a continuous time interval $[a, b]$ within which the first formula φ_1 should hold until φ_2 holds. STL operates on a continuous piecewise representation of a sampled signal. As illustrated in Fig 2j and 2k, the STL semantics uses interpolation to determine the point between two samples where the formula will start to hold or to be violated.

STL has two possible semantics: a qualitative semantics returning a yes/no answer to the question of whether the system satisfies or violates the specification, and a quantitative semantics also providing a measure of robustness [111,113] of how much the system violates or satisfies the specification. Negative robustness implies property violation, while positive robustness implies property satisfaction. As we discuss later in this section, this value can be used to guide the parameter synthesis of a biological model with unknown parameters.

Model Checking

Model checking is an automatic formal verification technique able to check the emergence of a particular behavior in a biological model. This technique operates over a discrete time model with a finite number of states, called a Kripke structure [136], where the execution of a model triggers a sequence of events determining the truth value of the propositions of a temporal logic formula. A Kripke structure is a special labeled graph in which the nodes represent the reachable states generated by executing the biological model, and the edges represent the state transitions. A labeling function maps each node to the set of propositions that hold in the corresponding reachable state. A transition relation specifies the set of possible successors for each state.

Each node always has a successor or a loop transition starting and ending in the same state, representing nonterminating computations where the evaluation of the atomic propositions does not change (also called fixed point). Suitable user-friendly tools can translate the biological models specified with one of the formalisms presented in the previous section into a Kripke structure representation that can be analyzed with very efficient model checkers such as NuSMV [137,138] or CADP [139]. The main drawback of this technique is that the number of states of a model usually grows exponentially in the number of its parameters, giving rise to the state explosion problem. In order to tackle this, the majority of model checkers do not explicitly represent the states, but represent sets of states symbolically [140,141]. For example, the states

and the transition relation of a Kripke structure can be encoded as a binary decision diagram (BDD) [140], a very compact acyclic graph data structure used to represent a Boolean function as well as sets or relations in general. The logical operations required by model checking are then interpreted as operations over sets and implemented by polynomial-time graph manipulation algorithms [140] directly on this representation of sets. The works of Bryant [140] on BDDs and of McMillan [141] on symbolic model checking provide more details on the symbolic approach.

Model checking techniques have also been extended to many other computational models that can be regarded as Kripke structures, such as continuous- and discrete-time Markov chains (CTMC and DTMC) (by adding probabilities) [6], Petri nets [41], hybrid systems (by adding continuous dynamics) [142], and spatial lattice-based models (using the quad-tree representation) [88]. In the case of CTMC and DTMC, the analysis may benefit from using probabilistic model checkers such as PRISM [6,143,144], which provide a real number in the interval of [0,1] corresponding to the probability that the system model will satisfy the property of interest. The algorithm used to calculate this probability can return either the exact solution [123], if it operates directly on the structure of the Markov chains, or an approximated solution, when it measures statistically [145] the probability of satisfying a property for a set of samples, generated using a Monte Carlo simulation of the system model. This statistical approach can be applied not only to the classical DTMC and CTMC models, but also to stochastic hybrid systems [146], in which the continuous dynamics are calculated by integration and the discrete transitions are chosen nondeterministically by following a probability distribution.

Runtime Verification/Monitoring

Another way to overcome the state explosion problem of model checking is to focus the analysis on a single execution trace instead of performing an exhaustive verification. Runtime verification is a lightweight yet powerful verification technique that aims to check whether the current execution of a program (i.e., the time series of the concentration of the protein expression during a gene regulatory network simulation) satisfies or violates a property of interest.

The emergent property is still specified in terms of a LTL formula or one of its extensions, but in this case only a single behavior is evaluated. Monitoring does not require a system model but only a set of observable, discrete, or continuous signals that can be collected during a wet-lab experiment or generated by numerical simulation. As previously mentioned, in the last decade, LTL has been extended to specify properties of real-valued variables defined over dense real time. A pioneering example of LTL with predicates expressed in terms of constraints over reals is LTL($\mathbb{R}$), presented in [147,148], and then implemented to monitor numerical simulations of biological models in BIOCHAM [108].

In addition, STL [131] extends LTL with a continuous time semantics and with predicates over real variables and is implemented in the Breach [80] and S-TaLiRo [84,149] tools. In these tools, the evaluation of the STL formula robustness for a particular trajectory through monitoring is used in combination with sensitivity-based analysis techniques [107,114,115,135] or stochastic-based optimization techniques [116,149] to steer the simulation of a biological model toward the parameter regions in which it would display the property of interest.

Static Analysis

As the term “static" indicates, this analysis is performed on the static description of the model without actually executing it. The principles of static analysis originated in the field of compiler optimization. Nowadays, this approach is widely employed in software verification, its key role being to detect potentially vulnerable code in safety-critical applications.

While model checking generally needs to explore all the states originated by executing the semantics of the model, static analysis operates on the syntactic level of the specification or by using abstract interpretation [150] over finite approximations of the possible model executions [151]. Static analysis can reveal important information regarding the model specification (e.g., the control structure, the flow of species concentrations, the interactions among species), without performing all of the underlying concrete calculations. In the last decade [117], static analysis has also become a useful technique to analyze biological models [93,118,119]. In [93], the authors successfully employ this technique to analyze biological pathways. Given a formal model in BioAmbients of the LDL degradation pathway, the authors compute a fine over-approximation of the possibly infinite reaction sequences that the model specifies. This approximation is "safe", meaning that all the reaction sequences that do not appear in the analysis are not possible.

Static analysis is crucial for dealing with the complexity of real systems (see also [118,119] for other important examples in systems biology), in which model checking all the reaction sequences will fail, owing to the state explosion problem. However, it is not as precise as executing the model: it is not possible to guarantee that all the reachable states in the over-approximation are also reachable in the original model's behavior.

In some cases (see, for example, the Rovergene tool [71]), static analysis and model checking techniques are combined. The first constructs an abstract domain using suitable abstractions. The second provides a logical framework to search in the abstract domain if a set of states is not reachable. This guarantees that they will never occur in the original model's behavior. Examples of this analytic approach can be found in several case studies: genetic networks [71], loss of cardiac cell excitability [66], and bone remodeling [107]. Static analysis has also been used in [152] to relate different semantics and formalisms used for describing reaction systems.

Tools

We now use the concepts previously discussed as a guide to choosing among the several tools available.

Fig 3 and Table 1, though not purporting to be complete, present a selection of software closely related to the topics discussed in this review. In Fig 3, we classify the listed tools by the computational modeling language, the supported semantics of execution, and the formal analysis that can be performed, based on the literature. We also specify if the tool supports a mechanism to tune the model's parameters, guided by the formal analysis.

While each modeling language was developed to solve a real problem, different modeling languages may map into the same program. Knowledge of the syntax is needed in order to carry out static analysis. The executable program, on the other hand, is no longer syntax-dependent. Quantitative analysis (i.e., simulation), which considers the time dimension, is then performed on the output produced by the program.

In the second column of Table 1 (main case studies), references to some applications are presented. The large variety of tools will accommodate the current rich interdisciplinary and multidisciplinary systems biology scenarios. Scientists with different backgrounds may have different initial preferences and later move in various directions, generating the conditions for extensive exchange of ideas and methodological innovations.

Conclusions and Vision

The growing availability of large amounts of data (i.e., big data) will allow models to be tested very finely. Spatial data could be collected in three dimensions (thanks, perhaps, to microscope imaging advances), capturing the formation of patterns, niches, molecular associations, and multiscale features. The time dimension could range from molecular events (for example,

Tool	Modeling Language											Execution			Formal Analysis			
	Agent-based	Boolean/Qualitative Networks	Biochemical Networks	Compartment-based	Hybrid Systems	Lattice-based	Markov Chains	Petri Nets	Process Algebra	Rule-based	State Charts	Continuous Semantics	Discrete/Boolean Semantics	Stochastic Semantics	Model Checking	Parameter Synthesis	Runtime Verification / Monitoring	Static Analysis
BAM [79]				✓					✓					✓				✓
BETAWB [11]									✓			✓	✓	✓				✓
BIOCHAM [29]										✓		✓	✓	✓	✓	✓	✓	✓
BIODIVINE [60, 136]			✓		✓		✓					✓	✓	✓	✓	✓		✓
BIONETGEN [21] + BIOLAB [137]										✓		✓		✓	✓			
BIO-PEPA WB [12]									✓			✓		✓	✓			✓
BOOLNET [45]		✓											✓	✓				
BMA [48]		✓											✓		✓			
BNS [46]		✓											✓		✓			
BREACH [68]					✓							✓				✓	✓	
COMPUCELL3D [87]	✓					✓								✓				
COPASI [8]			✓									✓		✓		✓		
dREACH [69]					✓							✓			✓	✓		
FLAME [26]	✓													✓				
GINSIM [138]		✓											✓		✓			
GREATSPN [40]								✓				✓	✓	✓	✓			
IBM RATIONAL RHAPSODY [50]											✓		✓					
KASIM [30] + KASA										✓		✓		✓				✓
MATHWORKS SIMULINK [67]					✓							✓						
PATHWAY LOGIC [41]								✓					✓		✓			
PRISM [6]							✓							✓	✓	✓		
ROVERGENE [62]					✓								✓		✓	✓		✓
SNOOPY [25] + MARCIE [39]								✓				✓	✓	✓	✓			
SPIM [14]									✓					✓				
S-TALIRO [70]					✓							✓		✓		✓	✓	
REPAST [27]	✓													✓				

Fig 3. Summary of the features for the selected tools. Tools are classified by the supported computational modeling language, their execution semantics, and the formal analysis that can be performed, based on the literature.

doi:10.1371/journal.pcbi.1004591.g003

DNA mutations or epigenetic changes) to organism development, circadian, species evolution, and other meaningful periodicities.

The development of new, efficient tools will motivate others to generate new computational models or to improve the existing ones. This will increase the community of scientists sharing their knowledge through standardized computational models reproducing numerically the behavior of the biological process under investigation. With computational modeling acquiring better capacity to describe biological systems and processes at a level useful for

Table 1. Summary of the main case studies in systems biology for the listed tools.

Tool	Main case studies
BAM [92]	LDL degradation pathway [93]
BetaWB [12]	The MAPK biochemical pathway [12], cell-cycle [11]
BIOCHAM [30]	Mammalian cell cycle control [34], G protein-coupled receptor kinases [31]
BioDivine [69,78]	Genetic regulatory networks [79]
BioNetGen [22] + BioLab [33]	HMGB1 signal pathway [35] Analysis of T-cell receptor signaling pathway [33]
Bio-PEPA WB [13]	Plant circadian clock [14]
BoolNet [52]	Genetic networks [52]
BMA [56]	Biological signaling networks [55]
BNS [53]	Cell cycle sequence of fission yeast [54]
Breach [80]	Collagen proteolysis [115], Cellular iron homeostasis network [81]
CompuCell3D [103]	Vertebrate segmentation and somite formation [104]
COPASI [8]	Biochemical networks [9]
dReach [82]	Cardiac cell hybrid models [83]
FLAME [27]	Sperm behavior [97]
GINsim [49]	Diversity and plasticity of Th cell types [50]
	MAPK network on cancer cell fate decision [51]
GreatSPN [43]	Signal transduction pathways for angiogenesis [44]
IBM Rational Rhapsody [58]	T-cell activation with statecharts [59]
KaSim [32]	EGFR signaling [36]
Mathworks Simulink [76]	Heart model for pacemaker verification [77]
Pathway Logic [45]	Sporulation initiation in *B. subtilis* [46] MAPK signaling network [46] EGF stimulation network [45]
PRISM [6]	Biological signaling pathways [6,143,144], bone pathologies [107]
Rovergene [71]	Synthetic transcription cascade [71], myocyte excitability [66], bone remodeling [107]
Snoopy [26] + MARCIE [42]	Systems and synthetic biology [41]
SPiM [15]	Modeling of the EGFR network [16], MHC class I peptide optimization [17]
S-TaLiRo [84]	Modeling of the insulin-glucose regulatory system [149]
REPAST [28]	Bone remodeling [98]

doi:10.1371/journal.pcbi.1004591.t001

prediction and to suggest experiments, it will trigger a useful feed-forward process with experimental biologists.

The tools described in this paper can already accommodate different complexly structured properties of biological processes and could be used separately or in different combinations and architectures. This will enable biologists to answer complex questions. For example, temporal logics, in particular, will have a profound impact in systems biology by helping to transform cause—effect relationships into objects that can be manipulated both mathematically and computationally. In epistatic control, temporal logics can be used to model two or more causal factors as interacting mechanistically with respect to the observed phenomenon. Doing so will establish powerful connections, with reasoning based on logic and statistics and the mechanisms and processes that underlie the observed behavior.

One future interesting research direction that we envision is the extension of the current formal analysis techniques and temporal logics to the spatial domain. For example, understanding

how a spatial pattern emerges from the biochemical level acting at the cellular level (i.e., morphogenesis in developmental biology) is currently very challenging because of both the high computational complexity required by the spatiotemporal modeling and the lack of a suitable specification language to specify the spatiotemporal patterns of interest [86,87,153].

Furthermore, the rapid progress of modern technologies for healthcare has led to a new generation of devices called medical cyber-physical systems [154], in which smart and collaborative computational elements control the biological systems. Examples include pacemakers, biocompatible and implantable devices, insulin pumps, electro-anatomical mapping and intervention, robotic prosthetics, and neurostimulators. Here, the computational modeling of the biological part is indispensable to the development of efficient and safe controlling devices. Furthermore, the successful application of formal analysis techniques and tools to verify the correct and safe behavior of these systems will have an economic impact on our society by reducing warranty, liability, and certification costs. We believe that the concepts and the computational tools described here represent core elements of computational description, particularly in the framework of systems biology, and will have some relevance to both newcomers and experts.

References

1. Fisher J, Henzinger TA. Executable cell biology. Nat Biotechnol. 2007 Nov; 25(11):1239–1249. PMID: 17989686
2. Hunt CH, Ropella GEP, Park S, Engelberg J. Dichotomies between computational and mathematical models. Nature Biotechnology. 2008; 26(7):737–738. doi: 10.1038/nbt0708-737 PMID: 18612289
3. Hlavacek WS, Faeder JR, Blinov ML, Posner RG, Hucka M, Fontana W. Rules for Modeling Signal-Transduction Systems. Sci STKE. 2006; 2006(344):re6. PMID: 16849649
4. Fisher J, Harel D, Henzinger TA. Biology As Reactivity. Commun ACM. 2011 Oct; 54(10):72–82.
5. Fisher J, Piterman N. The executable pathway to biological networks. Brief Funct Genomics. 2010 Jan; 9(1):79–92. doi: 10.1093/bfgp/elp054 PMID: 20118126
6. Kwiatkowska M, Norman G, Parker D. Probabilistic Model Checking for Systems Biology. In: Symbolic Systems Biology. Jones and Bartlett; 2010. p. 31–59.
7. Gay S, Soliman S, Fages F. A graphical method for reducing and relating models in systems biology. Bioinformatics. 2010; 26(18).
8. Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, et al. COPASI—a COmplex PAthway SImulator. Bioinformatics. 2006 Dec; 22(24):3067–3074. PMID: 17032683
9. Sahle S, Gauges R, Pahle J, Simus N, Kummer U, Hoops S, et al. Simulation of Biochemical Networks using Copasi—A Complex Pathway Simulator. In: Proc. of WSC 06: the Winter Simulation Conference. IEEE; 2006. p. 1698–1706.
10. Priami C, Regev A, Shapiro E, Silverman W. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Information Processing Letters. 2001; 80 (1):25–31.
11. Dematté L, Priami C, Romanel A. The BlenX Language: A Tutorial. In: Formal Methods for Computational Systems Biology 2008: 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems. vol. 5016 of LNCS. Springer-Verlag; 2008. p. 313–365.
12. Dematté L, Priami C, Romanel A. The Beta Workbench: a computational tool to study the dynamics of biological systems. Brief Bioinform. 2008; 9(5):437–449. doi: 10.1093/bib/bbn023 PMID: 18463130
13. Ciocchetta F, Hillston J. Bio-PEPA: A framework for the modelling and analysis of biological systems. Theoretical Computer Science. 2009; 410(33–34):3065–3084.
14. Guerriero ML, Pokhilko A, Fernandez AP, Halliday KJ, Millar AJ, Hillston J. Stochastic properties of the plant circadian clock. J R Soc Interface. 2012 Apr; 9(69):744–756. doi: 10.1098/rsif.2011.0378 PMID: 21880617
15. Phillips A, Cardelli L. Efficient, Correct Simulation of Biological Processes in the Stochastic Pi-calculus. In: Proc. of CMSB 2007: The 6th Conference on Computational Methods in Systems Biology. vol. 4695 of LNCS. Springer; 2007. p. 184–199.

16. Wang DY, Cardelli L, Phillips A, Piterman N, Fisher J. Computational modeling of the EGFR network elucidates control mechanisms regulating signal dynamics. BMC Syst Biol. 2009; 3:118. doi: 10.1186/1752-0509-3-118 PMID: 20028552

17. Dalchau N, Phillips A, Goldstein LD, Howarth M, Cardelli L, Emmott S, et al. A peptide filtering relation quantifies MHC class I peptide optimization. PLoS Comput Biol. 2011 Oct; 7(10):e1002144. doi: 10.1371/journal.pcbi.1002144 PMID: 22022238

18. Bortolussi L, Policriti A. Modeling Biological Systems in Stochastic Concurrent Constraint Programming. Constraints. 2008; 13(1–2):66–90.

19. Bartocci E, Corradini F, Di Berardini MR, Merelli E, Tesei L. Shape Calculus. A Spatial Mobile Calculus for 3D Shapes. Scientific Annals of Computer Science. 2010; 20(1):2010.

20. Bartocci E, Cacciagrano DR, Di Berardini MR, Merelli E, Tesei L. Timed Operational Semantics and Well-Formedness of Shape Calculus. Scientific Annals of Computer Science. 2010; 20(33):2010.

21. Regev A, Panina EM, Silverman W, Cardelli L, Shapiro E. BioAmbients: an abstraction for biological compartments. Theoretical Computer Science. 2004; 325(1):141–167.

22. Faeder JR, Blinov ML, Hlavacek WS. Rule-Based Modeling of Biochemical Systems with BioNetGen. Methods in Molecular Biology. 2009; 500:113–167. doi: 10.1007/978-1-59745-525-1_5 PMID: 19399430

23. Fisher J, Harel D. On Statecharts for Biology. In: Symbolic Systems Biology: Theory and Methods. Jones & Bartlett Publishers; 2010.

24. Bortolussi L, Policriti A. Hybrid Systems and Biology. In: Formal Methods for Computational Systems Biology. vol. 5016 of LNCS. Springer; 2008. p. 424–448.

25. Kauffman S. Metabolic stability and epigenesis in randomly constructed genetic nets. J Theor Biology. 1969 Mar; 22:437–467.

26. Heiner M, Herajy M, Liu F, Rohr C, Schwarick M. Snoopy—A Unifying Petri Net Tool. In: Proc. of Petri Nets 2012: the 33rd International Conference on Application and Theory of Petri Nets. vol. 7347 of LNCS. Springer; 2012. p. 398–407.

27. Richmond P, Walker DC, Coakley S, Romano DM. High performance cellular level agent-based simulation with FLAME for the GPU. Briefings in Bioinformatics. 2010; 11(3):334–347. doi: 10.1093/bib/bbp073 PMID: 20123941

28. North MJ, Collier NT, Vos JR. Experiences creating three implementations of the REPAST agent modeling toolkit. ACM Trans Model Comput Simul. 2006 Jan; 16(1):1–25.

29. John M, Lhoussaine C, Niehren J, Versari C. Biochemical Reaction Rules with Constraints. In: Proc. of ESOP 2011: the 20th European Symposium on Programming. vol. 6602 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2011. p. 338–357.

30. Chabrier-Rivier N, Fages F, Soliman S. The Biochemical Abstract Machine BIOCHAM. In: Proc. of CMSB 2005: the 3rd International Conference on Computational Methods in Systems Biology. vol. 3082 of LNCS. Springer; 2005. p. 172–191.

31. Heitzler D, Durand G, Gallay N, Rizk A, Ahn S, Kim J, et al. Competing G protein-coupled receptor kinases balance G protein and β-arrestin signaling. Mol Syst Biol. 2012; 8:590. doi: 10.1038/msb.2012.22 PMID: 22735336

32. Danos V, Feret J, Fontana W, Harmer R, Krivine J. Rule-Based Modelling of Cellular Signalling. In: Proc. of CONCUR 2007: 18th International Conference on Concurrency Theory. vol. 4703 of LNCS. Springer Berlin Heidelberg; 2007. p. 17–41.

33. Clarke EM, Faeder JR, Langmead CJ, Harris LA, Jha SK, Legay A. Statistical Model Checking in BioLab: Applications to the Automated Analysis of T-Cell Receptor Signaling Pathway. In: Proc. of CMSB 2008: the 6th International Conference on Computational Methods in Systems Biology. vol. 5307 of LNCS. Springer; 2008. p. 231–250.

34. Chabrier-Rivier N, Chiaverini M, Danos V, Fages F, Schächter V. Modeling and querying biomolecular interaction networks. Theor Comput Sci. 2004; 325(1):25–44.

35. Gong H, Zuliani P, Komuravelli A, Faeder JR, Clarke EM. Analysis and verification of the HMGB1 signaling pathway. BMC Bioinformatics. 2010; 11 Suppl 7:S10. doi: 10.1186/1471-2105-11-S7-S10 PMID: 21106117

36. Feret J, Danos V, Krivine J, Harmer R, Fontana W. Internal coarse-graining of molecular systems. Proceedings of the National Academy of Sciences. 2009; 106(16):6453–6458.

37. Petri CA, Reisig W. Petri net. Scholarpedia. 2008; 3(4):6477.

38. Cordero F, Horváth A, Manini D, Napione L, Pierro MD, Pavan S, et al. Simplification of a complex signal transduction model using invariants and flow equivalent servers. Theor Comput Sci. 2011; 412(43):6036–6057.

39. Koch I, Junker BH, Heiner M. Application of Petri Net theory for modelling and validation of the sucrose breakdown pathway in the potato tuber. Bioinformatics. 2005; 21(7):1219–1226. PMID: 15546934

40. Blätke MA, Heiner M, Marwan W. Predicting Phenotype from Genotype through Automatically Composed Petri Nets. In: Proc. of CMSB 2012: the 10th International Conference on Computational Methods in Systems Biology. vol. 7605 of LNCS; 2012. p. 87–106.

41. Heiner M, Gilbert D, Donaldson R. Petri Nets for Systems and Synthetic Biology. In: Formal Methods for Computational Systems Biology 2008: 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems. No. 5016 in Lecture Notes in Computer Science. Springer; 2008. p. 215–264.

42. Heiner M, Rohr C, Schwarick M. MARCIE—Model Checking and Reachability Analysis Done Efficiently. In: Application and Theory of Petri Nets and Concurrency. vol. 7927 of LNCS. Springer Berlin Heidelberg; 2013. p. 389–399.

43. Baarir S, Beccuti M, Cerotti D, De Pierro M, Donatelli S, Franceschinis G. The GreatSPN tool: recent enhancements. SIGMETRICS Perform Eval Rev. 2009; 36(4):4–9.

44. Napione L, Manini D, Cordero F, Horváth A, Picco A, Pierro M, et al. On the Use of Stochastic Petri Nets in the Analysis of Signal Transduction Pathways for Angiogenesis Process. In: Proc. of CMSB 2009: the 7th Conference on Computational Methods in Systems Biology. vol. 5688 of LNCS; 2009. p. 281–295.

45. Talcott C, Dill DL. The Pathway Logic Assistant. In: Proceedings of the Workshop Computational Methods in Systems Biology (CMSB); 2005. p. 228–239.

46. Tiwari A, Talcott C, Knapp M, Lincoln P, Laderoute K. Analyzing Pathways using SAT-based Approaches. In: Proc. of AB 2007: the 2nd Intl. Conf. on Algebraic Biology. vol. 4545 of LNCS. Springer; 2007. p. 155–169.

47. Thomas R, Kaufman M. Multistationarity, the basis of cell differentiation and memory. II. Logical analysis of regulatory networks in terms of feedback circuits. Chaos. 2001; 11(1):180–195. PMID: 12779452

48. Thomas R, Kaufman M. Multistationarity, the basis of cell differentiation and memory. I. Structural conditions of multistationarity and other nontrivial behavior. Chaos. 2001 Mar; 11(1):170–179. PMID: 12779451

49. Chaouiya C, Naldi A, Thieffry D. Logical modelling of gene regulatory networks with GINsim. Methods Mol Biol. 2012; 804:463–479. doi: 10.1007/978-1-61779-361-5_23 PMID: 22144167

50. Naldi A, Carneiro J, Chaouiya C, Thieffry D. Diversity and plasticity of Th cell types predicted from regulatory network modelling. PLoS Comput Biol. 2010; 6(9):e1000912. doi: 10.1371/journal.pcbi.1000912 PMID: 20824124

51. Grieco L, Calzone L, Bernard-Pierrot I, Radvanyi F, Kahn-Perles B, Thieffry D. Integrative modelling of the influence of MAPK network on cancer cell fate decision. PLoS Comput Biol. 2013 Oct; 9(10):e1003286. doi: 10.1371/journal.pcbi.1003286 PMID: 24250280

52. Mussel C, Hopfensitz M, Kestler HA. BoolNet—an R package for generation, reconstruction and analysis of Boolean networks. Bioinformatics. 2010 May; 26(10):1378–1380. doi: 10.1093/bioinformatics/btq124 PMID: 20378558

53. Dubrova E, Teslenko M. A SAT-Based Algorithm for Finding Attractors in Synchronous Boolean Networks. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2011; 8(5):1393–1399. doi: 10.1109/TCBB.2010.20 PMID: 21778527

54. Davidich MI, Bornholdt S. Boolean Network Model Predicts Cell Cycle Sequence of Fission Yeast. PLoS ONE. 2008; 3(2):e1672. doi: 10.1371/journal.pone.0001672 PMID: 18301750

55. Schaub MA, Henzinger TA, Fisher J. Qualitative networks: a symbolic approach to analyze biological signaling networks. BMC Syst Biol. 2007; 1:4. PMID: 17408511

56. Benque D, Bourton S, Cockerton C, Cook B, Fisher J, Ishtiaq S, et al. BMA: Visual Tool for Modeling and Analyzing Biological Networks. In: Proc. of CAV 2012: the 24th International Conference on Computer Aided Verification. vol. 7358 of LNCS. Berlin, Heidelberg: Springer-Verlag; 2012. p. 686–692.

57. Harel D. Statecharts: A Visual Formalism for Complex Systems. Sci Comput Program. 1987; 8(3):231–274.

58. Harel D, Gery E. Executable object modeling with statecharts. Computer. 1997; 30(7):31–42.

59. Kam N, Cohen IR, Harel D. The Immune System as a Reactive System: Modeling T-Cell Activation With Statecharts. In: Proc. HCC 2001: Human-Centric Computing Languages and Environments. IEEE Computer Society; 2001. p. 15–22.

60. Alur R, Courcoubetis C, Henzinger TA, Ho PH. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In: Hybrid Systems. vol. 736 of LNCS. Springer; 1993. p. 209–229.

61. Bartocci E, Bortolussi L, Smolka SA. Hybrid Systems and Biology. Information and Computation. 2014; 236:1–2.

62. Fromentin J, Eveillard D, Roux O. Hybrid modeling of biological networks: mixing temporal and qualitative biological properties. BMC Syst Biol. 2010; 4:79. doi: 10.1186/1752-0509-4-79 PMID: 20525331

63. Asarin E, Dang T, Girard A. Hybridization methods for the analysis of nonlinear systems. Acta Informatica. 2007; 43(7):451–476.

64. Dang T, Maler O, Testylier R. Accurate hybridization of nonlinear systems. In: Proc. of HSCC 2010: the 13th ACM International Conference on Hybrid Systems: Computation and Control. ACM; 2010. p. 11–20.

65. Dang T, Testylier R. Hybridization domain construction using curvature estimation. In: Proc. of HSCC 2011: the 14th International Conference on Hybrid Systems: computation and control. ACM; 2011. p. 123–132.

66. Grosu R, Batt G, Fenton F, Glimm J, Le Guernic C, Smolka SA, et al. From Cardiac Cells to Genetic Regulatory Networks. In: Proc. of CAV 2011: the 14th International Conference on Computer Aided Verification. vol. 6806 of LNCS. Springer Berlin / Heidelberg; 2011. p. 396–411.

67. Ghosh R, Tomlin C. Lateral Inhibition through Delta-Notch Signaling: A Piecewise Affine Hybrid Model. In: Proc. of HSCC 2001: the 4th International Workshop on Hybrid Systems: Computation and Control. vol. 2034 of LNCS. Springer; 2001. p. 232–246.

68. Ghosh R, Tiwari A, Tomlin C. Automated Symbolic Reachability Analysis; with Application to Delta-Notch Signaling Automata. In: Proc. of HSCC 2003: the 6th International Workshop on Hybrid Systems: Computation and Control. vol. 2623 of LNCS. Springer; 2003. p. 233–248.

69. Barnat J, Brim L, Cerná I, Drazan S, Fabriková J, Lánk J, et al. BioDiVinE: A Framework for Parallel Analysis of Biological Models. In: Proc. of COMPMOD 2009: the 2nd International Workshop on Computational Models for Cell Processes. vol. 6 of EPTCS; 2009. p. 31–45.

70. Batt G, Belta C, Weiss R. Temporal Logic Analysis of Gene Networks under Parameter Uncertainty. IEEE Trans of Automatic Control. 2008; 53:215–229.

71. Batt G, Yordanov B, Weiss R, Belta C. Robustness analysis and tuning of synthetic gene networks. Bioinformatics. 2007; 23(18):2415–2422. PMID: 17660209

72. Bartocci E, Cherry EM, Glimm J, Grosu R, Smolka SA, Fenton FH. Toward real-time simulation of cardiac dynamics. In: Proc. of CMSB 2011: the 9th International Conference on Computational Methods in Systems Biology. ACM; 2011. p. 103–112.

73. Bartocci E, Corradini F, Di Berardini MR, Entcheva E, Smolka SA, Grosu R. Modeling and simulation of cardiac tissue using hybrid I/O automata. Theoretical Computer Science. 2009; 410(33–34):3149–3165.

74. Bartocci E, Corradini F, Entcheva E, Grosu R, Smolka S. CellExcite: an efficient simulation environment for excitable cells. BMC Bioinformatics. 2008; 9(Suppl 2):S3. doi: 10.1186/1471-2105-9-S2-S3 PMID: 18387205

75. The MathWorks I. MATLAB; 2015. Natick, Massachusetts, United States.

76. The MathWorks I. Simulink; 2015. Natick, Massachusetts, United States.

77. Chen T, Diciolla M, Kwiatkowska MZ, Mereacre A. A simulink hybrid heart model for quantitative verification of cardiac pacemakers. In: Proc. of HSCC 2013: the 16th International Conference on Hybrid Systems: Computation and Control. ACM; 2013. p. 131–136.

78. Brim L, Ceska M, Drazan S, Safránek D. Exploring Parameter Space of Stochastic Biochemical Systems Using Quantitative Model Checking. In: Proc. of CAV 2013: the 25th International Conference on Computer Aided Verification 2013, Saint Petersburg, Russia, July 13–19, 2013. Proceedings. vol. 8044 of LNCS. Springer; 2013. p. 107–123.

79. Barnat J, Brim L, Krejci A, Streck A, Safránek D, Vejnar M, et al. On Parameter Synthesis by Parallel Model Checking. IEEE/ACM Trans Comput Biology Bioinform. 2012; 9(3):693–705.

80. Donzé A. Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In: Proc. of CAV 2010: the 22nd International Conference on Computer Aided Verification. vol. 6174 of LNCS. Springer Berlin; 2010. p. 167–170.

81. Mobilia N, Donzé A, Moulis JM, Fanchon E. A Model of the Cellular Iron Homeostasis Network Using Semi-Formal Methods for Parameter Space Exploration. In: Proc. of HSB 2012: First International Workshop on Hybrid Systems and Biology. vol. 92 of EPTCS; 2012. p. 42–57.

82. Kong S, Gao S, Chen W, Clarke E. dReach: δ-Reachability Analysis for Hybrid Systems. In: Proc. of TACAS 2015: the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems. vol. 9035. Springer Berlin Heidelberg; 2015. p. 200–205.

83. Liu B, Kong S, Gao S, Zuliani P, Clarke EM. Parameter Synthesis for Cardiac Cell Hybrid Models Using δ-Decisions. In: Proc. of CMSB 2014: the 12th International Conference on Computational Methods in Systems Biology. vol. 8859 of LNCS; 2014. p. 99–113.

84. Annapureddy YSR, Liu C, Fainekos GE, Sankaranarayanan S. S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems. In: Proc. of TACAS 2011: the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. vol. 6605 of LNCS. Springer; 2011. p. 254–257.

85. Murthy A, Bartocci E, Fenton FH, Glimm J, Gray R, Smolka SA, et al. Curvature analysis of cardiac excitation wavefronts. In: Proceedings of the 9th International Conference on Computational Methods in Systems Biology. CMSB '11. New York, NY, USA: ACM; 2011. p. 151–160.

86. Gol EA, Bartocci E, Belta C. A formal methods approach to pattern synthesis in reaction diffusion systems. In: Proc. of CDC 2014: the IEEE 53rd Annual Conference on Decision and Control; 2014. p. 108–113.

87. Haghighi I, Jones A, Kong Z, Bartocci E, Grosu R, Belta C. SpaTeL: A Novel Spatial-temporal Logic and Its Applications to Networked Systems. In: Proc. of HSCC '15: the 18th International Conference on Hybrid Systems: Computation and Control. ACM; 2015. p. 189–198.

88. Grosu R, Smolka SA, Corradini F, Wasilewska A, Entcheva E, Bartocci E. Learning and detecting emergent behavior in networks of cardiac myocytes. Communications of the ACM. 2009; 52(3):97–105.

89. Bartocci E, Singh R, von Stein FB, Amedome A, Caceres AJJ, Castillo J, et al. Teaching cardiac electrophysiology modeling to undergraduate students: laboratory exercises and GPU programming for the study of arrhythmias and spiral wave dynamics. Advances in Physiology Education. 2011; 35 (4):427–437. doi: 10.1152/advan.00034.2011 PMID: 22139782

90. Paun G, Rozenberg G. A guide to membrane computing. Theoretical Computer Science. 2002; 287 (1):73–100. Natural Computing.

91. Regev A, Panina E, Silverman W, Cardelli L, Shapiro E. BioAmbients: an abstraction for biological compartments. Theoretical Computer Science. 2004; 325(1):141–167. Computational Systems Biology.

92. Muganthan VA, Phillips A, Vigliotti MG. BAM: BioAmbient machine. In: Proc. of ACSD 2008: the 8th International Conference on Application of Concurrency to System Design. IEEE; 2008. p. 45–49.

93. Pilegaard H, Nielson F, Nielson HR. Pathway analysis for BioAmbients. J Log Algebr Program. 2008; 77(1–2):92–130.

94. Cardelli L. Brane Calculi. In: Proc. of CMSB 2004: the international Conference, Computational Methods in Systems Biology. vol. 3082 of LNCS. Springer; 2004. p. 257–278.

95. Merelli E, Armano G, Cannata N, Corradini F, d'Inverno M, Doms A, et al. Agents in bioinformatics, computational and systems biology. Briefings in Bioinformatics. 2007; 8(1):45–59. PMID: 16772270

96. Bartocci E, Cacciagrano D, Cannata N, Corradini F, Merelli E, Milanesi L, et al. An agent-based multilayer architecture for bioinformatics grids. NanoBioscience, IEEE Transactions on. 2007; 6(2):142–148.

97. Burkitt M, Walker DC, Romano DM, Fazeli A. Modelling sperm behaviour in a 3D environment. In: Proc. of CMSB 2011: the 9th International Conference on Computational Methods in Systems Biology; 2011. p. 141–149.

98. Paoletti N, Liò P, Merelli E, Viceconti M. Multilevel Computational Modeling and Quantitative Analysis of Bone Remodeling. IEEE/ACM Trans Comput Biology Bioinform. 2012; 9(5):1366–1378.

99. Deutsch A, Dormann S. Cellular Automaton Modeling of Biological Pattern Formation: Characterization, Applications, and Analysis. Genetic Programming and Evolvable Machines. 2007 Mar; 8(1):105–106.

100. Bethe HA. Statistical Theory of Superlattices. Proceedings of the Royal Society of London Series A, Mathematical and Physical Sciences. 1935; 150(871):552–575.

101. Zaki MH, Tahar S, Bois G. Computing cancer software models of complex tissues and disease are yielding a better understanding of cancer and suggesting potential treatments. Nature. 2012; 491: s62–s63.

102. Scianna M, P L, editor. Cellular Potts Models: Multiscale Developments and Biological Applications. Chapman Hall/CRC Press; 2013.

103. Izaguirre JA, Chaturvedi R, Huang C, Cickovski TM, Coffland J, Thomas GL, et al. COMPUCELL, a multi-model framework for simulation of morphogenesis. Bioinformatics. 2004; 20(7):1129–1137.

104. Hester SD, Belmonte JM, Gens JS, Clendenon SG, Glazier JA. A Multi-cell, Multi-scale Model of Vertebrate Segmentation and Somite Formation. PLoS Comput Biol. 2011; 7(10):e1002155. doi: 10.1371/journal.pcbi.1002155 PMID: 21998560

105. Menichetti G, Remondini D, Bianconi G. Correlations between weights and overlap in ensembles of weighted multiplex networks. Phys Rev E. 2014; 90:062817.

106. Bartocci E, Corradini F, Merelli E, Tesei L. Detecting synchronisation of biological oscillators by model checking. Theoretical Computer Science. 2010; 411(20):1999–2018.

107. Bartocci E, Liò P, Merelli E, Paoletti N. Multiple Verification in Complex Biological Systems: The Bone Remodelling Case Study. T Comp Sys Biology. 2012; 14:53–76.

108. Calzone L, Fages F, Soliman S. BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge. Bioinformatics. 2006; 22(14):1805–1807. PMID: 16672256

109. Batt G, Bergamini D, de Jong H, Garavel H, Mateescu R. Model Checking Genetic Regulatory Networks Using GNA and CADP. In: Proc. of SPIN 2004: the 11th International SPIN Workshop on Model Checking Software. vol. 2989 of LNCS; 2004. p. 158–163.

110. Rizk A, Batt G, Fages F, Soliman S. Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures. Theoretical Computer Science. 2011; 412(26):2827–2839.

111. Fainekos G, Pappas G. Robust Sampling for MITL Specifications. In: Proc. of FORMATS 2007: the 5th International Conference on Formal Modeling and Analysis of Timed Systems. vol. 4763 of LNCS. Springer; 2007. p. 147–162.

112. Rizk A, Batt G, Fages F, Soliman S. On a Continuous Degree of Satisfaction of Temporal Logic Formulae with Applications to Systems Biology. In: Proceedings of the 6th International Conference on Computational Methods in Systems Biology. CMSB '08. Berlin, Heidelberg: Springer-Verlag; 2008. p. 251–268.

113. Donzé A, Maler O. Robust Satisfaction of Temporal Logic over Real-Valued Signals. In: Proc. of FORMATS 2010: the 8th International Conference on Formal Modeling and Analysis of Timed Systems. vol. 6246 of LNCS. Springer; 2010. p. 92–106.

114. Donzé A, Clermont G, Langmead CJ. Parameter Synthesis in Nonlinear Dynamical Systems: Application to Systems Biology. Journal of Computational Biology. 2010; 17(3):325–336. doi: 10.1089/cmb.2009.0172 PMID: 20377448

115. Donzé A, Fanchon E, Gattepaille LM, Maler O, Tracqui P. Robustness analysis and behavior discrimination in enzymatic reaction networks. PLoS ONE. 2011; 6(9):e24246. doi: 10.1371/journal.pone.0024246 PMID: 21980344

116. Bartocci E, Bortolussi L, Nenzi L, Sanguinetti G. System design of stochastic models using robustness of temporal properties. Theor Comput Sci. 2015; 587:3–25.

117. Nielson F, Nielson HR, Priami C, Rosa D. Static Analysis for Systems Biology. In: Proc. of the Winter International Synposium on Information and Communication Technologies. WISICT '04. Trinity College Dublin; 2004. p. 1–6.

118. Danos V, Feret J, Fontana W, Krivine J. Abstract interpretation of cellular signalling networks. In: Proc. of VMCAI'2008: the Ninth International Conference on Verification, Model Checking and Abstract Interpretation. vol. 4905 of LNCS. Springer, Berlin, Germany; 2008. p. 83–97.

119. Feret J. Reachability Analysis of Biological Signalling Pathways by Abstract Interpretation. In: Proc. of ICCMSE'2007: the International Conference of Computational Methods in Sciences and Engineering. No. 963.(2) in American Institute of Physics Conference Proceedings. American Institute of Physics; 2007. p. 619–622.

120. Pnueli A. The temporal logic of programs. Foundations of Computer Science, IEEE Annual Symposium on. 1977; 0:46–57.

121. Clarke EM, Emerson E. Design and synthesis of synchronization skeletons using branching time temporal logic. In: Proc. of Logics of Programs, Workshop. vol. 131 of LNCS. Springer Berlin; 1982. p. 52–71.

122. Hansson H, Jonsson B. A Logic for Reasoning about Time and Reliability. Formal Asp Comput. 1994; 6(5):512–535.

123. Aziz A, Sanwal K, Singhal V, Brayton R. Model-checking continuous-time Markov chains. ACM Trans Comput Logic. 2000 Jul; 1(1):162–170.

124. Batt G, Belta C, Weiss R. Model Checking Liveness Properties of Genetic Regulatory Networks. In: Proc. of TACAS 2007: the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. vol. 5016 of LNCS. Springer-Verlag; 2007. p. 323–338.

125. Cook B, Fisher J, Krepska E, Piterman N. Proving Stabilization of Biological Systems. In: Proc. of VMCAI 2011: the 12th International Conference on Verification, Model Checking, and Abstract Interpretation. vol. 6538 of LNCS. Springer; 2011. p. 134–149.

126. Clarke EM, Emerson EA, Sistla AP. Automatic Verification of Finite-state Concurrent Systems Using Temporal Logic Specifications. ACM Trans Program Lang Syst. 1986; 8(2):244–263.

127. Emerson EA, Halpern JY. "Sometimes" and "Not Never" Revisited: On Branching Versus Linear Time Temporal Logic. J ACM. 1986; 33(1):151–178.

128. Alur R, Henzinger TA. Real-time Logics: Complexity and Expressiveness. In: LICS; 1990. p. 390–401.

129. Alur R, Henzinger TA. A Really Temporal Logic. J ACM. 1994; 41(1):181–204.

130. Alur R, Feder T, Henzinger TA. The Benefits of Relaxing Punctuality. J ACM. 1996; 43(1):116–146.

131. Maler O, Nickovic D. Monitoring Temporal Properties of Continuous Signals. In: Proc. of FORMATS/FTRTFT 2004: joint International Conferences on Formal Modeling and Analysis of Timed Systmes and Formal Techniques in Real-Time and Fault-Tolerant Systems. vol. 3253 of LNCS; 2004. p. 152–166.

132. Donzé A, Maler O, Bartocci E, Nickovic D, Grosu R, Smolka SA. On Temporal Logic and Signal Processing. In: Proc. of ATVA 2012, the 10th International Symposium on Automated Technology for Verification and Analysis. vol. 7561 of Lecture Notes in Computer Science. Springer; 2012. p. 92–106.

133. Bartocci E, Bortolussi L, Sanguinetti S. Data-driven Statistical Learning of Temporal Logic Properties. In: Proc. of FORMATS 2014: the 12th International Conference on Formal Modeling and Analysis of Timed Systems. vol. 8711 of LNCS; 2014. p. 23–37.

134. Bufo S, Bartocci E, Sanguinetti G, Borelli M, Lucangelo U, Bortolussi L. Temporal Logic Based Monitoring of Assisted Ventilation in Intensive Care Patients. In: Proc. of ISoLA 2014: the 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Part II. vol. 8803 of Lecture Notes in Computer Science. Springer; 2014. p. 391–403.

135. Bartocci E, Bortolussi L, Nenzi L. A Temporal Logic Approach to Modular Design of Synthetic Biological Circuits. In: Computational Methods in Systems Biology. vol. 8130 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2013. p. 164–177.

136. Kripke S. Semantical Considerations on Modal Logic. Acta Philosophica Fennica. 1963; 16:83–94.

137. Cimatti A, Clarke EM, Giunchiglia F, Roveri M. NuSMV: A New Symbolic Model Checker. STTT. 2000; 2(4):410–425.

138. Nusser-Stein S, Beyer A, Rimann I, Adamczyk M, Piterman N, Hajnal A, et al. Cell-cycle regulation of NOTCH signaling during C. elegans vulval development. Mol Syst Biol. 2012; 8:618. doi: 10.1038/msb.2012.51 PMID: 23047528

139. Garavel H, Lang F, Mateescu R, Serwe W. CADP 2010: A Toolbox for the Construction and Analysis of Distributed Processes. In: Proc. of TACAS 2011: the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. vol. 6605. Springer; 2011. p. 372–387.

140. Bryant RE. Graph-Based Algorithms for Boolean Function Manipulation. IEEE Trans Computers. 1986; 35(8):677–691.

141. Burch JR, Clarke EM, McMillan KL, Dill DL, Hwang LJ. Symbolic Model Checking: 1020 States and Beyond. Inf Comput. 1992; 98(2):142–170.

142. Ahmad J, Bourdon J, Eveillard D, Fromentin J, Roux O, Sinoquet C. Temporal constraints of a gene regulatory network: Refining a qualitative simulation. Biosystems. 2009; 98(3):149–159. doi: 10.1016/j.biosystems.2009.05.002 PMID: 19446002

143. Calder M, Vyshemirsky V, Gilbert D, Orton RJ. Analysis of Signalling Pathways Using Continuous Time Markov Chains. T Comp Sys Biology. 2006; 4220:44–67.

144. Heath J, Kwiatkowska M, Norman G, Parker D, Tymchyshyn O. Probabilistic model checking of complex biological pathways. Theoretical Computer Science. 2008; 319(3):239–257.

145. Jha SK, Clarke EM, Langmead CJ, Legay A, Platzer A, Zuliani P. A Bayesian Approach to Model Checking Biological Systems. In: Proc. of CMSB 2009: the 7th International Conference on Computational Methods in Systems Biology. vol. 5688 of Lecture Notes in Computer Science. Springer; 2009. p. 218–234.

146. Zuliani P, Baier C, Clarke EM. Rare-event verification for stochastic hybrid systems. In: HSCC. ACM; 2012. p. 217–226.

147. Antoniotti M, Policriti A, Ugel N, Mishra B. Model building and model checking for biochemical processes. Cell Biochemistry and Biophysics. 2003; 38:271–286. doi: 10.1385/CBB:38:3:271 PMID: 12794268

148. Calzone L, Chabrier-Rivier N, Fages F, Soliman S. Machine learning biochemical networks from temporal logic properties. Transactions on Computational Systems Biology VI. 2006; p. 68–94.

149. Sankaranarayanan S, Fainekos G. Simulating Insulin Infusion Pump Risks by In-Silico Modeling of the Insulin-Glucose Regulatory System. In: Proc. of CMSB 2012: the 10th Conference on Computational Methods in Systems Biology. vol. 7605 of LNCS; 2012. p. 322–341.

150. Cousot P, Cousot R. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proc. of POPL '77: the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages. ACM; 1977. p. 238–252.

151. Paulev L, Magnin M, Roux O. Static analysis of Biological Regulatory Networks dynamics using abstract interpretation. Math Struct in Comp Science. 2012; 22:651–685.

152. Fages F, Soliman S. Abstract interpretation and types for systems biology. Theor Comput Sci. 2008; 403(1):52–70.

153. Bartocci E, Bortolussi L, Milios D., Nenzi L., Sanguinetti G. Studying Emergent Behaviours in Morphogenesis using Signal Spatio-Temporal Logic. In: Proc. of HSB 2015: the 4th International Workshop on Hybrid Systems Biology. vol. 9271 of LNCS. Springer; 2015. p. 156–172

154. Bartocci E, Gao S, Smolka SA. Proc. of ISoLA 2014: the 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Part II. In: Medical Cyber-Physical Systems—(Track Introduction). vol. 8803 of Lecture Notes in Computer Science. Springer; 2014. p. 353–355.