BMC Bioinformatics

Software

Open Access

A service-oriented architecture for integrating the modeling and formal verification of genetic regulatory networks

Pedro T Monteiro[1,2], Estelle Dumas[1], Bruno Besson[1], Radu Mateescu[1], Michel Page[1,3], Ana T Freitas[2] and Hidde de Jong*[1]

Address: [1]INRIA Grenoble - Rhône-Alpes, 655 Avenue de l'Europe, Montbonnot, 38334 St Ismier Cedex, France, [2]INESC-ID/IST, Rua Alves Redol 9, 1000-029 Lisbon, Portugal and [3]Institut d'Administration des Entreprises, Université Pierre Mendès France, Grenoble, France

Email: Pedro T Monteiro - Pedro.Monteiro@inria.fr; Estelle Dumas - Estelle.Dumas@inria.fr; Bruno Besson - Bruno.Besson@inria.fr; Radu Mateescu - Radu.Mateescu@inria.fr; Michel Page - Michel.Page@iae-grenoble.fr; Ana T Freitas - atf@inesc-id.pt; Hidde de Jong* - Hidde.de-Jong@inria.fr

* Corresponding author

Published: 30 December 2009

BMC Bioinformatics 2009, **10**:450 doi:10.1186/1471-2105-10-450

Received: 17 July 2009
Accepted: 30 December 2009

This article is available from: http://www.biomedcentral.com/1471-2105/10/450

© 2009 Monteiro et al; licensee BioMed Central Ltd.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/2.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

Background: The study of biological networks has led to the development of increasingly large and detailed models. Computer tools are essential for the simulation of the dynamical behavior of the networks from the model. However, as the size of the models grows, it becomes infeasible to manually verify the predictions against experimental data or identify interesting features in a large number of simulation traces. Formal verification based on temporal logic and model checking provides promising methods to automate and scale the analysis of the models. However, a framework that tightly integrates modeling and simulation tools with model checkers is currently missing, on both the conceptual and the implementational level.

Results: We have developed a generic and modular web service, based on a service-oriented architecture, for integrating the modeling and formal verification of genetic regulatory networks. The architecture has been implemented in the context of the qualitative modeling and simulation tool GNA and the model checkers NUSMV and CADP. GNA has been extended with a verification module for the specification and checking of biological properties. The verification module also allows the display and visual inspection of the verification results.

Conclusions: The practical use of the proposed web service is illustrated by means of a scenario involving the analysis of a qualitative model of the carbon starvation response in *E. coli*. The service-oriented architecture allows modelers to define the model and proceed with the specification and formal verification of the biological properties by means of a unified graphical user interface. This guarantees a transparent access to formal verification technology for modelers of genetic regulatory networks.

Background

The study of genetic regulatory networks, as well as other biological networks, has led to the development of increasingly large and detailed models [1]. The models consist of dozens or even hundreds of variables describing the molecular species involved in a variety of intracellular processes [2-7]. Computer tools are essential for the simulation of the dynamical behavior of the networks from

the models, for instance when predicting the response of the system to an external perturbation. However, as the size of the models grows, it becomes infeasible to manually verify the predictions against experimental data or identify interesting features in dozens of simulation traces. This calls for the use of automated and scalable methods that help the modeler with the identification and verification of interesting dynamical properties of the network. The field of formal verification provides promising methods to prove or disprove specified properties of a system. These methods proceed by an exploration of all possible behaviors of the system, following two main approaches: logic inference, based on the use of axioms and proof rules [8], and model checking, based on an automatic and exhaustive search of the state space [9]. In this paper, we focus on the model checking approach. The basic idea underlying model checking is to specify dynamical properties of interest as statements in temporal logic, and to use model-checking algorithms to automatically and efficiently verify whether the properties are satisfied or not by the model [9]. In recent years, several examples of the application of model checking to the analysis of biological regulatory networks have been published in the literature (*e.g.*, [10-21]).

According to our experience, there are currently two major obstacles that prevent modelers in systems biology from drawing maximal benefit from formal verification tools. First, the formulation of biological questions in temporal logic and the interpretation of the verification results is far from obvious, especially for non-expert users who are not used to this kind of reasoning. Second, most of the existing modeling and simulation tools are not capable of applying model-checking techniques in a transparent way. In particular, they do not hide from the user the technical details of the installation of the model checker, the export in a suitable format of the model and the query, the call of the model checker, and the import of the results produced by the model checker (the true/false verdict and witnesses/counterexamples). In other words, what is missing is a framework that tightly integrates modeling and simulation tools with formal verification tools, on both the conceptual and the implementational level.

In order to address these issues, we propose a service-oriented architecture (SOA) [22] for the integrated modeling and formal verification of genetic regulatory networks, which reuses existing technology as much as possible. The architecture connects modeling and simulation clients to a formal verification server, *via* an intermediate request manager. In particular, the client can perform verification requests through the web, which the request manager dispatches to an appropriate formal verification server. When the formal verification server has answered the request, the results are sent back to the modeling and simulation client for display and further analysis in the graphical user interface of the tool. The interactions of the client with the remote web server are handled by a verification module assisting the specification of biological queries through a property editor, either by directly choosing the appropriate temporal logic operators or by using a tailored set of query patterns [23].

The architecture is generic and modular, but we develop it here in the context of one particular modeling and simulation tool (GNA [24]) and two different model checkers (NUSMV [25] and CADP [26]). A first generalization of the work presented here would be to integrate other formal verification tools into the architecture. This possibility is anticipated through the use of a plugin system, where each plugin contains all data transformations and operations specific to a particular formal verification tool. This simplifies the integration of a new tool to the creation of the corresponding plugin. A second generalization would be to extend the service-oriented architecture to other modeling and simulation tools. A variety of tools have been used in combination with model checkers, such as GINSIM [27], INA [28], BIOCHAM [29], GNA [24] or ROVERGENE [14], based on formalisms like Boolean and other logical models [30-32], Petri nets [21,33,34] or ordinary differential equations [35,36]. In order to integrate new modeling and simulation tools into the architecture, they each have to be equipped with a verification module that interacts with the request manager, sending verification requests and receiving answers and diagnostics, as well as a plugin system to define the contents of the messages.

In the next section, we describe the service-oriented architecture and its components in detail and we motivate the most important implementation choices. The practical use of the architecture is then illustrated by means of a scenario involving the analysis of a qualitative model of the carbon starvation response in *E. coli*. The model describes a network of key global regulators of the bacterium, responsible for the control of the expression of a large number of stress response genes [37,38]. We trace the different steps from the formulation of a temporal logic query to the visualization and interpretation of the verification results. The discussion summarizes our contributions and places it in the context of related work.

Implementation

In this section, we describe the overall architecture of the system (Figure 1), with a step-by-step description of its components: the modeling and simulation tool with its verification module, the request manager, and the formal verification server. These three components have been implemented in Java 1.5 and their web-service interface is based on Apache Axis 1.3 http://ws.apache.org/axis/.

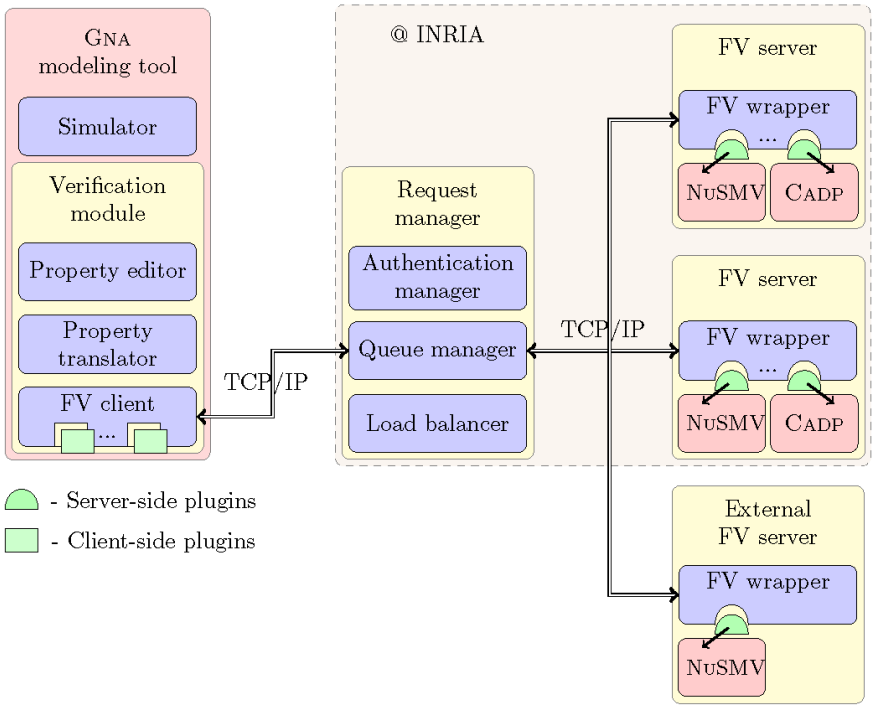

Figure 1
Service-oriented architecture. Service-oriented architecture for the integration of tools for the modeling and simulation of genetic regulatory networks with formal verification (FV) tools. In particular, the architecture has been implemented for the connection of GNA with the model checkers NUSMV and CADP. GNA is extended with a verification module responsible for the transformation of the model and properties into a format specific to a formal verification tool, and for the communication with the other components of the service-oriented architecture.

The implementation followed two main principles: a service-oriented architecture and the use of plugins. A service-oriented architecture is particularly well suited for our purpose. The formal verification service is remotely executed through the web and is implemented using standard protocols and languages like TCP/IP, SOAP and XML. A GNA user wishing to perform a verification request does not need to install a model checker or other formal verification tool locally on his or her machine. The use of plugins provides a flexible and extensible way to abstract a particular formal verification tool. It allows one to apply the tool without worrying about the details of its implementation.

Modeling and simulation tool

The service-oriented architecture is accessible for users of version 7.0 of the qualitative modeling and simulation tool Genetic Network Analyzer (GNA), available as described at the end of the paper. GNA uses a class of piecewise-linear (PL) differential equations, providing a coarse-grained picture of the dynamics of genetic regulatory networks [39]. The models associate a protein concentration variable to each of the genes in the network, and capture the switch-like character of gene regulation by means of step functions that change their value at threshold concentrations of regulatory proteins. The advantage of using PL models is that the qualitative dynamics of the high-dimensional systems are relatively simple to analyze, using inequality constraints on the parameters rather than exact numerical values [13,40]. This makes the PL models a valuable tool for the analysis of genetic regulatory networks in the absence of quantitative information on the parameter values. The graphical user interface of GNA supports the modeler in building step-by-step a PL model of the network under study (see the tutorial available from the GNA web site for details and examples).

GNA computes discrete abstractions of the continuous dynamics of the PL models, resulting in a finite-state transition system (FSTS) defined as a quintuple $\Sigma = \langle S, S_0, AP, L, T \rangle$ [9]. S is a set of states, where each state corresponds to a hyperrectangular region in the concentration space, defined by the thresholds of the concentration variables. $S_0 \subseteq S$ is the set of initial states. AP is a set of atomic propositions, related to the states by means of a labeling function L: $S \rightarrow 2^{AP}$. The labeling function determines which atomic propositions are satisfied in a particular state $s \in S$. The atomic propositions concern among other things, the

thresholds bounding the concentration variables, the signs of the derivatives of the concentration variables, and indicate if the state is a steady state. *T* represents the set of transitions between the states, where each transition corresponds to a solution trajectory entering one region from another [40]. GNA allows the user to visualize the FSTS, *i.e.* to display the corresponding state transition graph, and analyze the atomic propositions characterizing the states. For large graphs visual inspection quickly becomes infeasible and formal verification tools are needed. Previous versions of GNA supported the export of the FSTS to text files accepted by several model checkers [13,41]. Version 7.0 extends GNA with a verification module that integrates the tool into the service-oriented architecture.

Verification module

The verification module consists of three components: a pattern-based property editor, a property translator, and a formal verification client (Figure 1).

Pattern-based property editor and translator

The problem of posing relevant and interesting questions is critical in modeling in general, but even more so in the context of applying formal verification methods, due to the fact that is not easy for non-experts to formulate queries in temporal logic. The pattern-based property editor is a user interface that allows the specification of biologically-relevant properties in the form of temporal logic formulas. The specification of properties can be achieved in two distinct ways: for common biological properties through the use of a pattern system, and for more specific or complex properties through the use of a text editor of temporal logic formulas.

Patterns are high-level query templates that formulate recurring questions in the analysis of regulatory networks using a domain-specific language [42] rather than temporal logic. They were originally introduced in the formal verification field [43] and recently adapted for use in systems biology [23]. From a study of the literature on the modeling of biological regulatory networks, it was found that most of the questions asked by experts can be reduced to a set of four patterns concerning the: *occurrence/exclusion*, *consequence*, *sequence* and *invariance* of events. Notice that these patterns are classes of properties sufficiently generic to be applicable in a variety of systems biology models, and that the aim of these patterns is not to cover all possible questions the modeler can think of, but rather to simplify the formulation of the most frequent or otherwise important ones.

Figure 2 shows the pattern-based property editor of GNA. It presents the four different types of patterns as templates to be completed by the user. The completion of the templates requires the modeler to have previously defined atomic propositions, each of which describes characteristics of a state of the network, such as an increasing or decreasing protein concentration, a steady state, or a protein concentration above a certain threshold. When a pattern has been specified, it is automatically translated into Computation Tree Logic (CTL) [9]. The pattern-based property editor and translator are also available as a standalone Java application (see Availability and requirements below). An application programming interface (API) is provided, so that the patterns can be integrated into other modeling tools that wish to implement the encoding of biological properties into temporal logic formulas (CTL, CTRL and μ-calculus are currently supported).

More complex biological properties can be directly specified in the Computation Tree Regular Logic (CTRL) language [44]. CTRL extends CTL with regular expressions and fairness operators, which favors the expression of properties like multistability or oscillations, and endows the logic with a user-friendly syntax. The text editor allows the modeler to specify any temporal logic formula by freely combining the set of CTL and CTRL operators with propositional logic operators and the user-defined atomic propositions. The temporal logic properties can be stored for later use with the GNA model in a single project file.

Formal verification client and client-side plugins

The formal verification client is the component that enables GNA to communicate with the request manager. It thus gives the user an easy access to the formal verification technology without having to locally install a tool or worrying about how to get it to work. To perform a verification request, the modeler needs to choose which tool to use, which property to verify, *etc.* These choices may be guided by the estimation of the model size (*e.g.*, for large regulatory networks containing dozens of genes, symbolic model checking is likely to scale up better than explicit-state model checking) or by the nature of the properties to be verified (*e.g.*, linear-time or branching-time, with/without regular expressions, *etc.*). The FSTS on which the property is to be verified can be defined explicitly or implicitly. In the former case, the FSTS is completely generated by the simulation module of GNA, while in the latter case it is given by the set of initial states and a function that computes the successors of any given state. The formal verification client performs a request by sending the implicit or explicit description of the FSTS through the web and waiting for the result. The implicit definition has the advantage of considerably reducing the size of the specification of the FSTS, and thus limiting the size of the files transmitted and the response delays. This may be critical for large FSTSs. The verification result is composed of a true (false) verdict supported by a witness (counterexample). The witness or counterexample consists of a sequence of states in the FSTS, displayed in the graphical user interface of GNA.

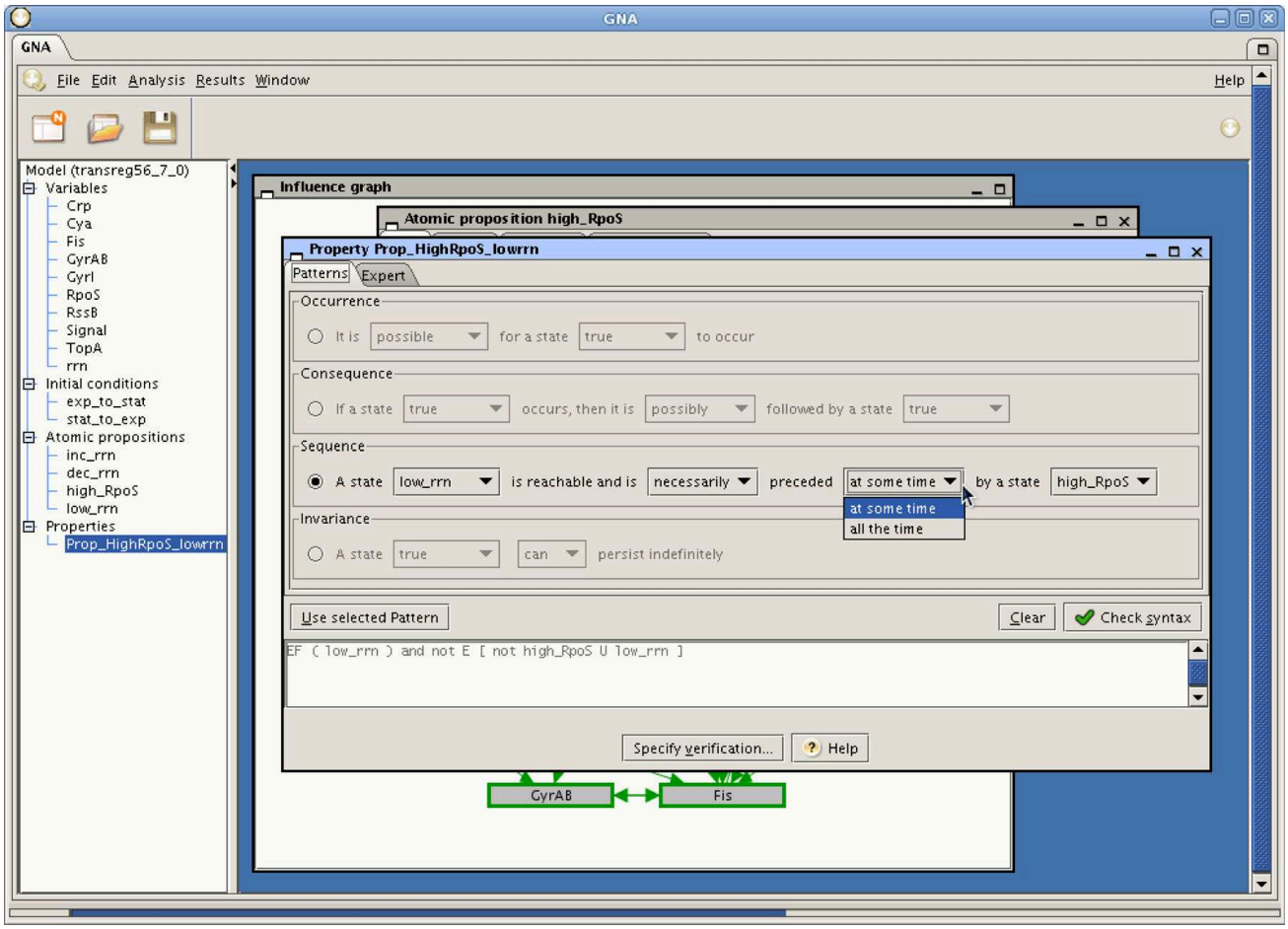

Figure 2
Pattern-based property editor. Graphical user interface for the specification of biological properties. The modeler can use a pattern-based property editor for frequently-asked questions, and a text editor for the specification of more complex biological properties (expert mode).

In order to make the verification module of the modeling tool independent of a specific formal verification tool, we have developed a plugin system. Currently, a plugin for the model checker NuSMV is available, while a beta version for CADP has been completed. All data transformations specific to a particular model checker are taken in charge by the corresponding plugin, thus leaving the service-oriented architecture free to manage generic verification requests. Each plugin has a client-side and a server-side (Figure 1). The client-side plugin has the responsibility of translating the FSTS and the property into a format accepted by the corresponding formal verification tool, while the server-side plugin is in charge of receiving the translated FSTS and property, feeding them into the formal verification tool executable, and parsing the results returned by the tool.

At the present time, the model checkers integrated in the architecture are invoked using the default parameters. More elaborate choices could be partially automated by incorporating into the plugin some knowledge of the verification method and the underlying algorithms.

Request manager

The request manager is a component of the service-oriented architecture with a public address http://java1.inrialpes.fr, acting as an intermediary service that ensures the communication between all the modeling tools and formal verification servers.

Queue and authentication manager

In order to keep track of the state of all verification requests and the available formal verification servers, a queue and authentication manager has been implemented. Upon each verification request the authentication manager, together with the server-side plugin, checks for the credentials of the request. If successful, the queue manager registers the request in the queue, checks for an

available formal verification server, and hands over the request. The queue manager will continue to poll the formal verification server for a response until one of three events happens: the verification has completed, the user has aborted the verification request, or a timeout has occurred. The verification result (verdict and counterexample) is then returned to the user.

To ensure the service security, each authenticated request registered in the queue, generates an Universally Unique Identifier (UUID) that is returned to the client, so that only this client is able to retrieve the verification result. Furthermore, when the result is retrieved, both the request manager and the formal verification server that handled the request delete the model and temporal logic formula, leaving no traces of the request in the server.

Load balancer

The service-oriented architecture has been designed to support several formal verification servers. The address of every server, as well as all the model checkers and other formal verification tools types locally installed on each of the servers, are registered in the request manager. Upon a verification request, the load balancer chooses an idle formal verification server with the required tool and server-side plugin installed. When all formal verification servers are busy, the load balancer waits until one becomes idle.

Formal verification server

A formal verification server has the responsibility of verifying properties submitted by the request manager. One or several formal verification tools can be installed on a server provided that the corresponding server-side plugins are also installed on this server.

Formal verification server and server-side plugins

The formal verification server contains the web-service interface, which is responsible for receiving the requests from the request manager, the choice of the corresponding server-side plugin, and the construction of the verification result to be returned.

Each plugin specific to a formal verification tool has an authentication module which responds to the authentication requests made by the request manager. In addition, upon a verification request, the plugin pre-processes the model description and the property in order to transform them into the format accepted by the formal verification tool, and calls the latter with the appropriate parameters. When the formal verification tool finishes the verification of the request, it produces the verdict as well as the corresponding witness (or counterexample).

Since this witness has a format specific to a particular formal verification tool, it is up to the plugin to parse the results and perform the necessary data transformations to a common format that is sent back to the modeling tool. The required transformations depend on whether the request involves a FSTS of an implicit or explicit type. In the explicit case, the witness is simply a subgraph of the FSTS sent to the formal verification tool, whereas in the implicit case the state information needs to be reconstructed from the output of the tool.

Integration of new formal verification tools

An important advantage of the chosen architecture, which delegates all operations that are specific to a particular formal verification tool to plugins, is that it allows for the flexible integration of new tools. Two plugins have been developed until now: one for NUSMV (released with the GNA distribution) and one for CADP (beta version completed). Developers wanting to develop plugins for different model checkers or other formal verification tools can do so through the following main steps: the development of a client-side plugin, the development of a server-side plugin, and the installation of the server-side plugin on a server on which the new tool is running.

The client-side plugin takes a .jar file that must be placed in the *plugins* directory of GNA, allowing the modeling tool to export the FSTS to a file that can be read by the new formal verification tool. GNA dynamically recognizes the available client-side plugins, using the Java Plugin Framework technology. The development of a server-side plugin results in a Java class that needs to be copied in an appropriate directory of the wrapper on the formal verification server. The latter server must register its web service connection parameters in the request manager, so as to enable the latter to dispatch the requests to the correct formal verification server. More detailed information on the development of plugins can be obtained by contacting the authors directly.

Results

In order to illustrate the use of the web service, we present a scenario using a PL model of the network of global regulators controlling the carbon starvation response in the enterobacterium *Escherichia coli*. In order to survive, *E. coli* cells constantly have to adapt their functioning to the availability of carbon sources, essential for growth. The adaptation involves multiple levels of regulation, from metabolic fluxes and enzyme activity to gene regulation [45-47]. In this example, we focus in particular on the role of the global regulators of transcription, such as CRP, Fis, DNA supercoiling, and RpoS. These global regulators form the backbone of the network coordinating the long-term response of *E. coli* cells to starvation conditions (Figure 3). The PL model consists of 9 equations and more than 50 parameter inequalities that specify the qualitative dynamics of the system [38]. Below, we illustrate how the

specification and verification of temporal logic properties can help the analysis of the role of RpoS in the dynamics of the system.

Property specification procedure

RpoS or σ^s is a sigma factor that allows cells to adapt to and survive under harmful conditions by expressing a variety of stress response genes [48]. Due to its key role in the cell, the concentration of RpoS is tightly regulated at the transcriptional, translational, and post-translational levels. It this section, we focus on the conditions of stability of the protein. While cells grow on a carbon source, RpoS is actively degraded through the protein RssB, which binds to RpoS and targets the factor to an intracellular protease (Figure 3). However, the depletion of the carbon source inactivates RssB, thus allowing RpoS to accumulate to a high concentration. Given the important role of RpoS for the survival of the cell, one may ask whether the entry into stationary phase upon carbon starvation is always preceded by the accumulation of RpoS in the cell.

The first step in answering this question using the formal verification module of GNA consists in identifying elements of the question that refer to the state of the biological system and in stating these as atomic propositions. We represent the entry into stationary phase of the system by a low level of stable RNAs encoded by the *rrn* operons. This is motivated by the fact that stationary-phase cells do not need high levels of stable RNAs, contrary to what is required by the high translational activity in exponential phase. These characteristics are specified using the property editor, where we create an atomic proposition named *low_rrn* (Additional file 1), restricting the concentration values for the variable *rrn* to those below its (single) threshold. We also introduce an atomic proposition *high_RpoS*, representing the accumulation of RpoS to a value above its threshold *t_RpoS* (Additional file 2).

The second step is the formulation of the biological property using the pattern-based property editor and translator. We choose the *sequence* pattern to account for the temporal ordering of the two states: stationary phase and high expression of RpoS. The sequence pattern is instantiated by selecting the previously defined atomic propositions (Figure 2):

"A state | *low_rrn* | is reachable and is | *necessarily* | preceded | *at some time* | by a state | *high_RpoS*".

Once the pattern is fully instantiated, it is automatically translated into the corresponding CTL formula: EF (*low_rrn*)∧¬E (¬*high_RpoS* ½ *low_rrn*).

Property verification procedure

After the specification of the property, one passes to the verification stage. For this step, the verification request

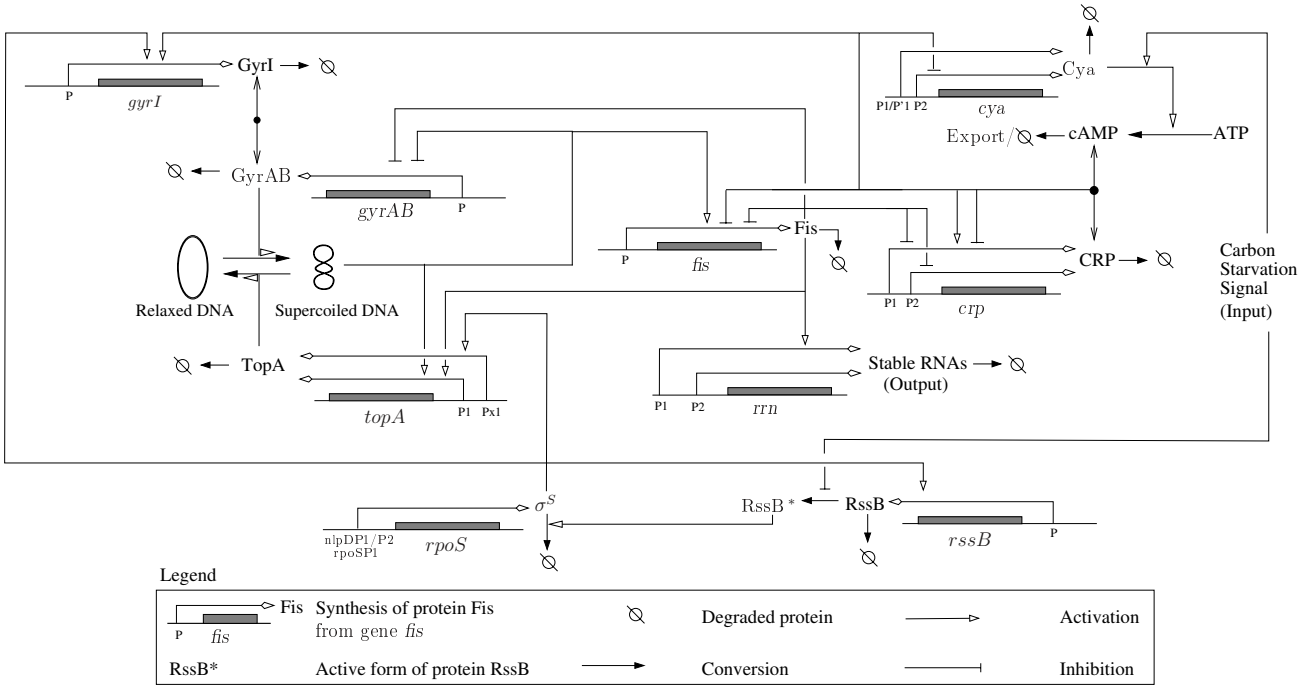

Figure 3
Carbon starvation response network in E. coli. Network of key genes, proteins and regulatory interactions involved in the carbon starvation response network in *E. coli* [37,38].

must be configured in the verification window (Additional file 3). First, we choose the name and version of the model checker plugin to be used (version 1.0 of the NuSMV implicit plugin was used in the example). Second, we specify the initial conditions. The resulting implicit Fsts represents the transition from exponential phase to stationary phase, starting from initial conditions corresponding to carbon depletion. We then run the verification request, which is treated by the service-oriented architecture as described in the previous section.

In response to the query the model checker returns false after 4 seconds. It means that the entry into stationary phase is not always preceded by the accumulation of RpoS in the cell. The counterexample is presented to the user as shown in the left panel of Figure 4. It consists of a subgraph of the initial Fsts, starting from the specified initial state and ending in a state where the property fails. By selecting a path in this subgraph, Gna allows the qualitative changes in the concentration of all the variables to be displayed (right panel of Figure 4). Looking at the evolution of the variables we immediately observe that there is (at least) one sequence of states leading to a low expression level of the *rrn* operons without having previously passed through a state with a high concentration of RpoS. This illustrates the negative verification result, and witnesses that the downregulation of the stable RNAs does not require the previous accumulation of RpoS.

Another verification example

Continuing with the previous analysis, one may want to look into the role of RpoS in the control of DNA supercoiling during growth-phase transitions. The DNA supercoiling level is regulated by the gyrase GyrAB, which supercoils the DNA structure, and by the topoisomerase TopA, which relaxes it.

In order to know whether *topA* is expressed in response to the carbon source availability, we create an atomic proposition named *low_topA* representing the low expression of *topA*, and we choose the following *invariance* pattern to check if the absence of *topA* expression persists indefinitely:

"A state|*low_topA*|can|persist indefinitely".

The corresponding translation of this pattern is the following Ctl formula: `EG` (*low_topA*).

Following the previously described verification procedure, the formal verification server returns false after 3 seconds, and the counterexample shows that expression of *topA* is stimulated at the entry into stationary phase, under the influence of RpoS. Indeed, following carbon starvation, the protein RssB is inactivated, which leads to the accumulation of RpoS at high levels. RpoS in turn activates the *topA* promoter. Complex properties like the existence of oscillations can also be verified. If the property holds, the verification module will present the corresponding lasso-shaped witness (Additional file 4) for visual inspection.

Discussion

In this paper, we have proposed a generic and modular service-oriented architecture to integrate the modeling of genetic regulatory networks with existing formal verification tools. Currently, the service-oriented architecture connects the Gna modeling tool, extended with a formal verification module, with the NuSMV and Cadp model checkers. We have given a detailed description of the existing components and motivated our implementation decisions. Additionally, we have illustrated the use of this architecture with the analysis of the complex network of global regulators involved in the carbon starvation response in *E. coli*. Gna is freely available for non-profit academic research, while the main component of the formal verification module, the pattern-based property editor and translator, is also available separately (see Availability and requirements below).

Formal verification methods have historically been used for the verification of hardware and software systems. Some of the existing model checkers, such as PRISM [49] and NuSMV [25], have recently been applied to the verification of biological systems. PRISM verifies properties specified in Continuous time Stochastic Logic (CSL) and has been used to perform quantitative analysis [17] of the ERK intracellular signaling pathway model [50]. NuSMV has been used for the analysis of biological models like the carbon starvation response in *E. coli* [13], the cell-cycle control in *C. crescentus* [20], the mucus production in *P. aeruginosa* [15], and the mammalian cell-cycle control [18]. In most cases, the biological models are built using modeling tools that are not connected to model checkers. Some modeling tools like GINsim [27] and previous versions of Gna [41] are capable of exporting the model in an implicit or explicit format accepted by the model checker and the entire analysis is carried out in the model-checking environment, without any feedback to the modeling tool. An exception is the modeling tool BIOcham [29], which integrates the model checker NuSMV and allows for a more flexible iterative modeling and verification approach.

Conclusions

In this paper we carried further the integration of the modeling and formal verification of biological networks, by proposing a service-oriented architecture that presents several advantages. First of all, the proposed connection between modeling and verification tools is completely

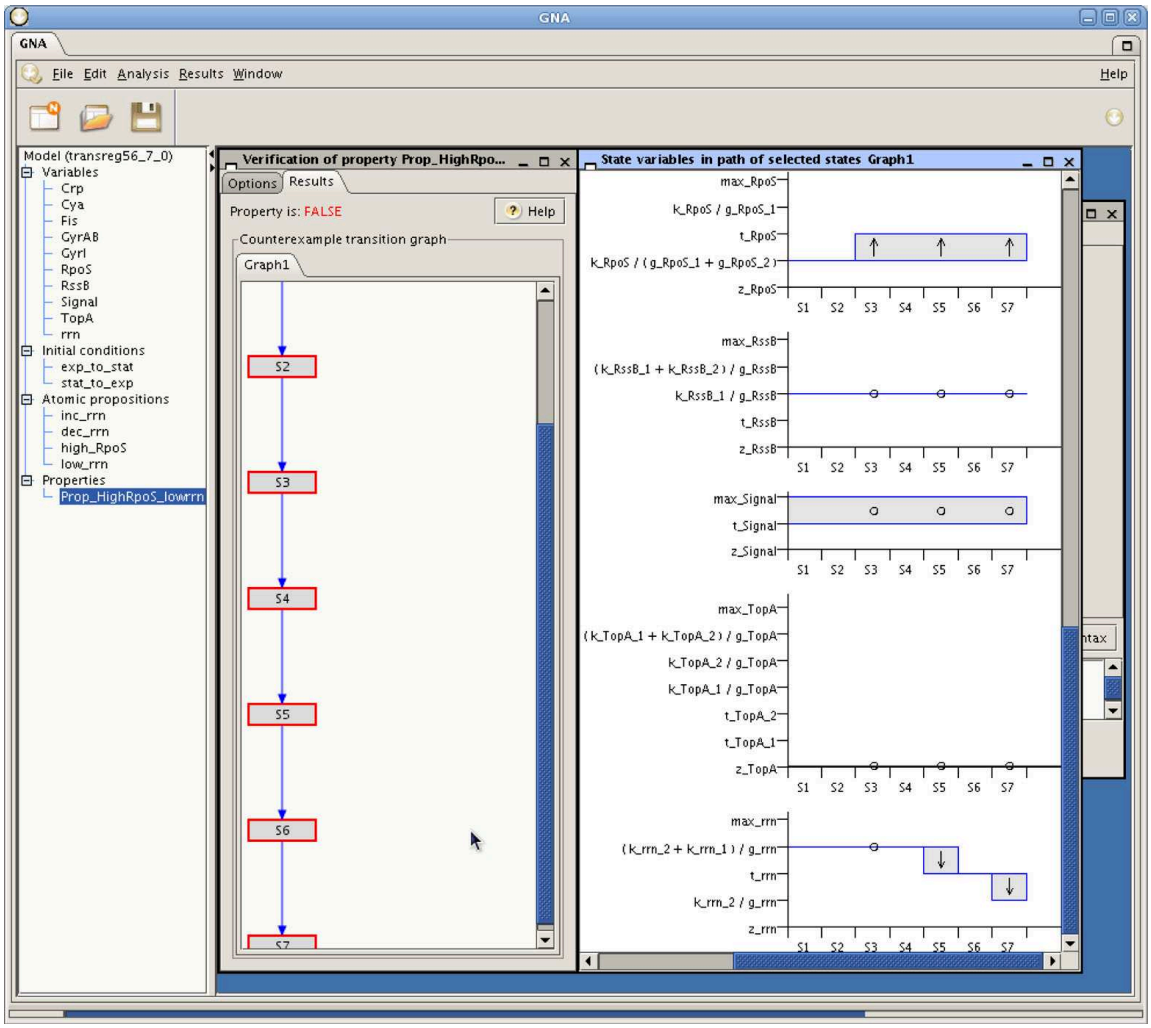

Figure 4
Verification result. Result of the verification of the biological property specified in Figure 3, consisting of a false verdict and the corresponding counterexample composed of a subgraph of the FSTS (see left panel). The qualitative evolution of the concentration variables of the selected states of the counterexample is visualized (see right panel).

transparent for the modeler and platform-independent. It requires web access but this is becoming less and less of a constraint in the current age of pervasive internet use. Second, the web-service based integration of the tools coming from different domains makes it possible to exploit the strong points of each. On the modeling side, the graphical user interfaces present the properties to be verified and the verification results in a way accessible to the modeler. For instance, the specification of biological properties by means of query patterns [23] does not require prior knowledge of any specific temporal logic. On the verification side, the latest developments of state-of-the-art model checkers can be immediately integrated. Third, the plugin system provides a modular way to add new formal verification methods without having to develop a new version of the modeling tool. The upgrade to future releases of a formal verification tool can also be performed through a simple plugin update.

The architecture has been implemented in the context of GNA, but generalizations to other modeling and simulation tools is obviously possible and facilitated by the modular structure. The integration of such tools into the architecture requires them to implement a verification module responsible for the specification of biological properties, the call of plugins for specific formal verification tools and the exchange of verification requests with the request manager. However, this implementation work is facilitated by the availability of the pattern-based property editor as a stand-alone Java application. In addition, the development of new plugins for tools based on model formalisms that can be mapped to FSTSs, explicitly or

implicitly, are conveniently designed after plugins already available for GNA.

Formal verification methods are promising tools for upscaling the analysis of genetic regulatory networks. The widespread adoption of these approaches has been hampered so far, by the difficulty for non-expert users to formulate appropriate questions in temporal logic, effectively use formal verification tools, and meaningfully interpret the results returned by the model checker. The modular infrastructure that we propose is capable of connecting modeling and formal verification tools. In combination with graphical user interfaces capable of presenting data in a form accessible to modelers, we expect this to lower the obstacles to the use of formal verification technology in biology.

Availability and requirements

Project name: Genetic Network Analyzer 7.0 (including the NUSMV plugin)

Project home page: http://ibis.inrialpes.fr/article122.html

Operating system(s): Platform independent (Windows, Linux, MacOS)

Programming language: Java 1.5

License: GNA is distributed by Genostar http://www.genostar.com/. Free license for non-commercial academic users granted upon request on the GNA home page.

Any restrictions to use by non-academics: contact Genostar at info@genostar.com for conditions.

Project name: Procrustes: Pattern-based property editor

Project home page: http://ibis.inrialpes.fr/article938.html

Operating system(s): Platform independent (Windows, Linux, MacOS)

Programming language: Java 1.5

License: LGPL

Authors' contributions

PTM designed and implemented the service-oriented architecture as well as the pattern editor and the NUSMV and CADP plugins, carried out the analysis of the *E. coli* example and drafted the manuscript. ED designed and implemented the service-oriented architecture and the CADP plugin. BB helped with the implementation of the service-oriented architecture. RM helped with the design of the pattern editor and the design and implementation of the CADP plugin. MP designed the service-oriented architecture, helped with the design and implementation of the NUSMV plugin, and helped to draft the manuscript. ATF helped with the design of the pattern editor and helped to draft the manuscript. HdJ helped with the design of the service-oriented architecture as well as the NUSMV and CADP plugins, helped with the analysis of the *E. coli* example and drafted the manuscript. All authors read and approved the final manuscript.

Additional material

Additional file 1

Definition of the atomic proposition **low_rrn**. *Atomic proposition specification window, where atomic propositions are defined in terms of restrictions applied to a state (*e.g., *restrictions on concentration values, focal sets, derivatives, and other state descriptors). In this case, the value of the concentration is restricted to lie below the threshold* t_rrn.
Click here for file
[http://www.biomedcentral.com/content/supplementary/1471-2105-10-450-S1.PDF]

Additional file 2

Definition of the atomic proposition **high_RpoS**. *Atomic proposition specification window, where atomic propositions are defined in terms of restrictions applied to a state (*e.g., *restrictions on concentration values, focal sets, derivatives, and other state descriptors). In this case, the value of the concentration is restricted to lie above the threshold* t_RpoS.
Click here for file
[http://www.biomedcentral.com/content/supplementary/1471-2105-10-450-S2.PDF]

Additional file 3

Verification options window. *Configuration of a verification request by specifying the model checker plugin to be used and, if the plugin supports an implicit representation of the FSTS, the initial conditions for the qualitative simulation of the network.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1471-2105-10-450-S3.PDF]

Additional file 4

Verification result. *Results of the verification of a complex biological property, composed of a verdict (true) and the corresponding witness. The latter consists of a sequence of states containing a cycle (see left panel). The value of the concentration of the variables in the selected states is shown, presenting an oscillation of the concentration of the variable* Fis *(see right panel).*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1471-2105-10-450-S4.PDF]

Acknowledgements

We would like to thank Delphine Ropers for helping with the analysis of the *E. coli* carbon starvation model and Valentina Baldazzi for testing the GNA user interface and the architecture.

This work was partially supported by the FCT program (PhD grant SFRH/BD/32965/2006 to PTM) and the PDTC program (project PTDC/EIA/71587/2006). BB and HdJ were supported by the European commission

under project COBIOS (FP6-2005-NEST-PATH-COM/043379). RM and HdJ were supported by the European commission under project EC-MOAN (FP6-2005-NEST-PATH-COM/043235).

References

1. Jamshidi N, Palsson B: **Formulating genome-scale kinetic models in the post-genome era.** *Mol Syst Biol* 2008, **4(171):**.
2. Chen WW, Schoeberl B, Jasper PJ, Niepel M, Nielsen UB, Lauffenburger DA, Sorger PK: **Input-output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data.** *Mol Syst Biol* 2009, **5(239):**.
3. Chen KC, Calzone L, Csikasz-Nagy A, Cross FR, Novak B, Tyson JJ: **Integrative analysis of cell cycle control in budding yeast.** *Mol Biol Cell* 2004, **15(8):**3841-3862.
4. Forger DB, Peskin CS: **A detailed predictive model of the mammalian circadian clock.** *Proc Natl Acad Sci USA* 2003, **100(25):**14806-14811.
5. Klipp E, Nordlander B, Krüger R, Gennemark P, Hohmann S: **Integrative model of the response of yeast to osmotic shock.** *Nat Biotechnol* 2005, **23(8):**975-982.
6. Leloup JC, Goldbeter A: **Toward a detailed computational model for the mammalian circadian clock.** *Proc Natl Acad Sci USA* 2003, **100(12):**7051-7056.
7. Saez-Rodriguez J, Simeoni L, Lindquist JA, Hemenway R, Bommhardt U, Arndt B, Haus UU, Weismantel R, Gilles ED, Klamt S, Schraven B: **A logical model provides insights into T cell receptor signaling.** *PLoS Comput Biol* 2007, **3(8):**e163.
8. Manna Z, Pnueli A: *The temporal logic of reactive and concurrent systems: specification Volume I*. Springer-Verlag; 1992.
9. Clarke EM, Grumberg O, Peled DA: *Model Checking* Boston, MA: MIT Press; 1999.
10. Antoniotti M, Policriti A, Ugel N, Mishra B: **Model building and model checking for biochemical processes.** *Cell Biochem Biophys* 2003, **38(3):**271-286.
11. Ballarini P, Mazza T, Palmisano A, Csikasz-Nagy A: **Studying irreversible transitions in a model of cell cycle regulation.** *Proc. 3rd Intl. Workshop Practical Applications of Stochastic Modelling (PASM 2008)* 2009.
12. Barnat J, Brim L, Cerná I, Drazan S, Safranek D: **Parallel model checking large-scale genetic regulatory networks with DiVinE.** *Proc. 1st Intl. Workshop From Biology To Concurrency and Back (FBTC 2007), Volume 194 of ENTCS* 2008.
13. Batt G, Ropers D, de Jong H, Geiselmann J, Mateescu R, Page M, Schneider D: **Validation of qualitative models of genetic regulatory networks by model checking: Analysis of the nutritional stress response in *Escherichia coli*.** *Bioinformatics* 2005, **21(Suppl 1):**i19-i28.
14. Batt G, Yordanov B, Belta C, Weiss R: **Robustness analysis and tuning of synthetic gene networks.** *Bioinformatics* 2007, **23(18):**2415-2422.
15. Bernot G, Comet JP, Richard A, Guespin J: **Application of formal methods to biological regulatory networks: Extending Thomas' asynchronous logical approach with temporal logic.** *J Theor Biol* 2004, **229(3):**339-348.
16. Bosnacki D, ten Eikelder HM, Steijaert MN, de Vink EP: **Stochastic analysis of amino acid substitution in protein synthesis.** In *Proc. 6th Conf. Computational Methods in Systems Biology (CMSB 2008), Volume 5307 of LNCS* Edited by: Heiner M, Uhrmacher AM. Berlin: Springer-Verlag; 2008:367-386.
17. Calder M, Vyshemirsky V, Gilbert D, Orton R: **Analysis of signalling pathways using the PRISM model checker.** *Proc. 3th Intl. Workshop Computational Methods in Systems Biology (CMSB 2005), Edinburgh, Scotland* 2005:79-90.
18. 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.
19. Fisher J, Piterman N, Hajnal A, Henzinger TA: **Predictive modeling of signaling crosstalk during *C. elegans* vulval development.** *PLoS Comput Biol* 2007, **3(5):**e92.
20. Shen X, Collier J, Dill D, Shapiro L, Horowitz M, McAdams HH: **Architecture and inherent robustness of a bacterial cell-cycle control system.** *Proc Natl Acad Sci USA* 2008, **105(32):**11340-11345.
21. Gilbert D, Heiner M, Lehrack S: **A unifying framework for modelling and analysing biochemical pathways using Petri nets.** In *Proc. 5th Conf. Computational Methods in Systems Biology (CMSB 2007), Volume 4695 of LNBI* Edited by: Calder M, Gilmore S. Berlin: Springer-Verlag; 2007:200-216.
22. Erl T: *Service-oriented architecture* New Jersey: Prentice Hall PTR; 2004.
23. Monteiro PT, Ropers D, Mateescu R, Freitas AT, de Jong H: **Temporal logic patterns for querying dynamic models of cellular interaction networks.** *Bioinformatics* 2008, **24(16):**i227-i233.
24. de Jong H, Geiselmann J, Hernandez C, Page M: **Genetic Network Analyzer: Qualitative simulation of genetic regulatory networks.** *Bioinformatics* 2003, **19(3):**336-344.
25. Cimatti A, Clarke EM, Giunchiglia E, Giunchiglia F, Pistore M, Roveri M, Sebastiani R, Tacchella A: **NuSMV2: An OpenSource tool for symbolic model checking.** In *Proc. 14th Intl. Conf. Computer Aided Verification (CAV 2002), Volume 2404 of LNCS* Edited by: Brinksma D, Larsen K. Berlin: Springer-Verlag; 2002:359-364.
26. Garavel H, Mateescu R, Lang F, Serwe W: **CADP 2006: A toolbox for the construction and analysis of distributed processes.** In *Proc. 19th Intl. Conf. Computer Aided Verification (CAV 2007), Volume 4590 of LNCS* Edited by: Damm W, Hermanns H. Berlin: Springer-Verlag; 2007:158-163.
27. Naldi A, Berenguier D, Fauré A, Lopez F, Thieffry D, Chaouiya C: **Logical modelling of regulatory networks with GINsim 2.3.** *Biosystems* 2009, **97(2):**134-139.
28. Starke P, Roch S: **INA - The Integrated Net Analyzer.** *Humboldt University Berlin* 1999 [http://www2.informatik.hu-berlin.de/~starke/ina.html].
29. Calzone L, Fages F, Soliman S: **BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge.** *Bioinformatics* 2006, **22(14):**1805-1807.
30. Thomas R, Thieffry D, Kaufman M: **Dynamical behaviour of biological regulatory networks. I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state.** *Bull Math Biol* 1995, **57(2):**247-276.
31. Li F, Long T, Lu Y, Ouyang Q, Tang C: **The yeast cell-cycle network is robustly designed.** *Proc Natl Acad Sci USA* 2004, **101(14):**4781-4786.
32. Shmulevich I, Dougherty E, Kim S, Zhang W: **Probabilistic Boolean networks: A rule-based uncertainty model for gene regulatory networks.** *Bioinformatics* 2002, **18(2):**261-274.
33. 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.
34. Chaouiya C, Remy E, Thieffry D: **Qualitative Petri net modelling of genetic networks.** In *Transactions on Computational Systems Biology VI, Volume 4220 of LNCS* Edited by: Istrail S, Pevzner P, Waterman M. Berlin: Springer-Verlag; 2006:95-112.
35. de Jong H: **Modeling and simulation of genetic regulatory systems: A literature review.** *J Comput Bio* 2002, **9:**69-105.
36. Bolouri H, Davidson EH: **The gene regulatory network basis of the "community effect," and analysis of a sea urchin embryo example.** *Dev Biol* 2009 in press.
37. Ropers D, de Jong H, Page M, Schneider D, Geiselmann J: **Qualitative simulation of the carbon starvation response in.** *Escherichia coli* 2006, **84(2):**124-152.
38. Ropers D, Baldazzi V, de Jong H: **Model reduction using piecewise-linear approximations preserves dynamic properties of the carbon starvation response in.** *Escherichia coli* 2009 in press.
39. Glass L, Kauffman SA: **The logical analysis of continuous non-linear biochemical control networks.** *J Theor Biol* 1973, **39(1):**103-129.
40. Batt G, de Jong H, Page M, Geiselmann J: **Symbolic reachability analysis of genetic regulatory networks using discrete abstractions.** *Automatica* 2008, **44(4):**982-989.
41. Batt G, Bergamini D, de Jong H, Gavarel H, Mateescu R: **Model checking genetic regulatory networks using GNA and CADP.** In *Proc. 11th Intl. Workshop Model Checking of Software (SPIN 2004), Volume 2989 of LNCS* Edited by: Graf S, Mounier L. Berlin: Springer-Verlag; 2004:158-163.
42. Mernik M, Heering J, Sloane AM: **When and how to develop domain-specific languages.** *ACM Comput Surv* 2005, **37(4):**316-344.
43. Dwyer MB, Avrunin GS, Corbett JC: **Patterns in property specifications for finite-state verification.** *Proc. 21st Intl. Conf. Software Engineering (ICSE 1999), Los Alamitos, CA* 1999:411-420.
44. Mateescu R, Monteiro PT, Dumas E, de Jong H: **Computation tree regular logic for genetic regulatory networks.** In *Proc. 6th Intl.*

Symposium Automated Technology for Verification and Analysis (ATVA 2008), Volume 5311 of LNCS Edited by: Cha SD, Choi JY, Kim M, Lee I, Viswanathan M. Springer-Verlag; 2008:48-63. [Full version available as INRIA Research Report RR-6521]

45. Hardiman T, Lemuth K, Keller M, Reuss M, Siemann-Herzberg M: **Topology of the global regulatory network of carbon limitation in *Escherichia coli*.** *J Biotechnol* 2007, **132(4):**359-374.
46. Kremling A, Kremling S, Bettenbrock K: **Catabolite repression in *Escherichia coli* - a comparison of modelling approaches.** *FEBS J* 2009, **276(2):**594-602.
47. Gutierrez-Rios RM, Freyre-Gonzalez JA, Resendis O, Collado-Vides J, Saier M, Gosset G: **Identification of regulatory network topological units coordinating the genome-wide transcriptional response to glucose in *Escherichia coli*.** *BMC Microbiol* 2007, **7:**53.
48. Hengge-Aronis R: **Regulation of gene expression during entry into stationary phase.** In *Escherichia coli and Salmonella: Cellular and Molecular Biology* Edited by: Neidhardt F, Curtiss R III, Ingraham J, Lin E, Low K, Magasanik B, Reznikoff W, Riley M, Schaechter M, Umbarger H. Washington DC: ASM Press; 1996:1497-1512.
49. Hinton A, Kwiatkowska M, Norman G, Parker D: **PRISM: A tool for automatic verification of probabilistic systems.** In *Proc. 12th Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2006), Volume 3920 of LNCS* Edited by: Hermanns H, Palsberg J. Springer-Verlag; 2006:441-444.
50. Cho KH, Shin SY, Kim HW, Wolkenhauer O, McFerran B, Kolch W: **Mathematical modeling of the influence of RKIP on the ERK signaling pathway.** In *Proc. 1st Intl. Workshop Computational Methods in Systems Biology (CMSB 2003), Volume 2602 of LNCS* Springer-Verlag; 2003:127-141.

Publish with **BioMed Central** and every scientist can read your work free of charge

"BioMed Central will be the most significant development for disseminating the results of biomedical research in our lifetime."

Sir Paul Nurse, Cancer Research UK

Your research papers will be:

- available free of charge to the entire biomedical community
- peer reviewed and published immediately upon acceptance
- cited in PubMed and archived on PubMed Central
- yours — you keep the copyright

Submit your manuscript here:
http://www.biomedcentral.com/info/publishing_adv.asp

BioMedcentral