Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 194 (2008) 35–50
www.elsevier.com/locate/entcs

Parallel Model Checking Large-Scale Genetic Regulatory Networks with DiVinE

J. Barnat, L. Brim, I. Černá, S. Dražan, and D. Šafránek[1,2]

Faculty of Informatics, Masaryk University, Czech Republic

Abstract

Studies of cells in silico can greatly reduce the need for expensive and prolonged laboratory experimentation. The use of model checking for the analysis of biological networks has attracted much attention recently. One of the practical limitations is the size of the model. In the paper we report on parallel model checking of genetic regulatory network using the model-checker DiVinE. The approach can check linear time properties on large networks.

Keywords: genetic regulatory networks, discrete simulation, parallel model checking

1 Introduction

Understanding of the molecular mechanisms underlying the behaviour of complex living systems is the main challenge of systems biology. In particular, investigation of how large and complex biochemical regulatory networks control the response of a living cell to its ever-changing environment is the central topic receiving recently much attention. The stress response to environmental events is induced by the interaction of several interwoven modules with complex dynamic behaviour, acting on different time scales. These networks are large and complex. For instance, a human cell contains in the order of 10,000 substances which are involved in 15,000 different types of reactions. This gives rise to a giant cellular network with complex positive and negative feedback loops.

In order to deal with the complexity of living systems, experimental methods have to be supplemented with mathematical modelling and computer-supported analysis. One of the most critical limitations in applying current approaches to modelling and analysis is their pure scalability. Large models require powerful

[1] Email: safranek@fi.muni.cz

[2] This work has been supported by the FP6 project No. NEST-043235 (EC-MOAN) and by the Academy of Sciences of CR grant No. 1ET408050503.

doi:10.1016/j.entcs.2007.12.001

computational methods, the hardware infrastructure is available (clusters, GRID, multi-core computers), but the parallel (distributed) algorithms for model analysis are still under development.

The most widely-used modelling frameworks for the analysis of the dynamics of these networks are based on ordinary differential equations [30] (ODE). The reduction of continuous models to discrete automata by a sequence of reductions, approximations, and abstractions allows formal methods for the automated verification of properties of discrete transition systems to be applied [9]. When dealing with large models from systems biology, standard model-checking techniques will not provide acceptable response times for answering user queries and parallel model-checking algorithms are required. Owing to dynamical dependences among state variables, the so-called *state-space explosion problem* arises during reduction to discrete automata. Even relatively small ODE models containing around 15 state variables lead to large automata having hundreds of thousands states. In other words, models that represent smaller parts of complex biological networks appear to computer science as *large-scale* models.

Various authors have proposed ways how to speedup model checking by either using powerful shared-memory multiprocessors (e.g. multi-core machines) or by distributing the memory requirements over several machines (e.g. on a cluster of workstations). The work on parallel verification is quite extensive, growing in recent years. There are attempts to consider both the symbolic as well as the enumerative techniques, theorem-provers as well as sat-solvers, branching as well as linear time temporal logics etc.

Model checking traditionally terms the task of verifying an implementation with respect to its specification. However, model checking could and probably should also be considered as a flexible analysis tool—as long as the object to analyse is representable as a finite-state system and the analysis can be formulated in a suitable temporal logic. In consequence, model checkers are at the heart of many modelling and analysis tools and will be in the future.

Recent studies on biologically-relevant properties identified the need for both branching-time temporal operators, able to express bi-stability properties (reachability of two different equilibrium states) and fairness operators, able to capture the oscillations of protein concentrations. While substantial work on model-checking qualitative as well as quantitative properties of biochemical networks has been already achieved, no attempts to use parallel model checkers to analyse complex networks are known.

In this paper we report on parallel model checking of genetic regulatory networks using the model-checker DiVinE [1]. We use the discretization method as proposed in [19] and implemented in *Genetic Network Analyser (GNA)* [6]. The key property of this method is simulation of ODE systems with unknown reaction rate coefficients. The distributed state space generator is built as an inherent part of the DiVinE tool which allows on-the-fly generation of the transition graph giving thus in many circumstances the possibility to analyse properties of even larger networks as opposed to the explicit representation as used e.g. in GNA. DiVinE

implements several efficient parallel LTL model-checking algorithms. This allows to check biologically interesting liveness properties on larger models than is possible with traditional sequential approach. Our primary aim is to demonstrate that parallel model checking can be satisfactorily employed for modelling and analysis of complex genetic regulatory networks.

1.1 Related Work

As the biochemistry that controls cells of living organisms is a very complicated machinery, nontrivial physical properties have to be taken into account in models. There is a large scale of mathematical approaches for modelling of biochemistry [10,22]. They vary in amount of abstraction they provide. In this paper we deal with the traditional deterministic approach based on ODE. Most of tools provided for this approach are based on direct numeric simulation of ODE solutions. Recently there appear specialised massively-parallel platforms [20,29,33] which can rapidly accelerate the numerical simulation. On the contrary, in this paper we deal with approximative discrete simulation of ODE solutions. This approach appears to be useful for analysis of genetic networks with unknown exact values of reaction rate coefficients [6,16].

The use of model checking for the analysis of biological networks has attracted much attention [9,32]. The individual approaches differ in models and model-checking tools used. Our approach is based on qualitative hybrid models as proposed by [19] and implemented in the GNA [17]. Besides GNA there are some other sequential approaches for model-checking of ODE models. The BIOCHAM workbench [12] provides an interface to symbolic model checker NuSMV and the enumerative CADP verification toolbox; the interface is based on a simple language for representing biochemical networks. The workbench provides mechanisms to reason about reachability, existence of partially described stable states, and some types of temporal behaviour. Another tool is the Robust Verification of Gene Networks (RoVerGeNe) [2]. To the best of our knowledge, none of the tools mentioned above employ the parallel approach.

2 Piecewise-linear models of genetic regulatory networks

Genetic regulation is the central process which drives the transcription of DNA into messenger RNA during protein synthesis. Proteins themselves act as *transcription factors* (activators or inhibitors) of the transcriptional reactions.

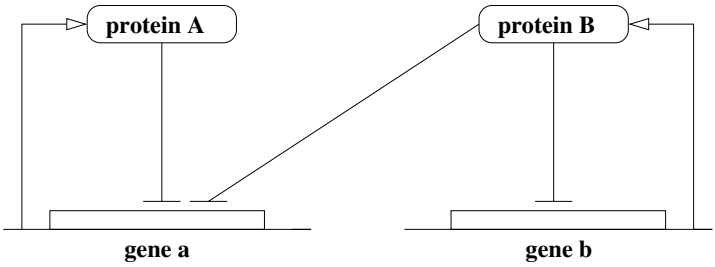
Although it is in general a very difficult task to choose the appropriate model for sufficient description of biochemical processes in a particular living organism [10], the ordinary differential equation (ODE) models are supposed to satisfactorily handle the chemical reaction mechanisms of genetic regulation in procaryotic organisms such as E. coli [6] or B. subtilis [16]. Suitable variants of ODE models can be also used for description of protein-protein reactions, metabolisms, and signal transduction processes of such elementary organisms. As the genetic regulation represents a well-studied field from the computational biology point of view, we firstly focus

on this phenomenon in order to find the potential benefits of distributed computing with respect to future analysis of complex biochemical processes.

2.1 Piecewise-linear differential equation model

Each ODE model is determined by a set of *state equations* each denoting the amount of change of concentration of the respective substrate continuously in time. As an ODE model describing an even simple biological process can be in general nonlinear [25], an abstraction of the general ODE models has been proposed by Glass [23]. In this abstraction, the underlying phase space of continuous system dynamics is uniformly partitioned into finite number of bounded subspaces (so-called *regulatory domains*) in which the system behaves linearly. More particularly, the potentially nonlinear state equations are replaced in such a simplified model with piecewise-linear (PL) equations. This simplification follows naturally from model systems representing switching networks in continuous time.

In Fig. 1(b) there is an example of a PL model (according to Mestl's extension [30] of the Glass model) of a system describing genetic regulation of two genes with mutual interactions illustrated by the simplified genetic network (a). In this sample regulatory system, the genes a and b encode the proteins A and B, respectively.



(a)

$$\frac{dx_a}{dt} = \kappa_a \, s^-(x_a, \theta_a^1) \, s^-(x_b, \theta_b^1) - \gamma_a \, x_a,$$
$$\frac{dx_b}{dt} = \kappa_b \, s^-(x_b, \theta_b^2) - \gamma_b \, x_b$$

(b)

Fig. 1. A two-gene regulatory network and its PL model

The protein A acts as a transcription factor which degrades the expression of the gene a and that way it inhibits production of its own. The protein B acts as a transcription factor which degrades the expression of both genes a and b. Hence there are two reactions in this system – synthesis of the protein A, characterised by *production rate* κ_a and *degradation rate* γ_a, and synthesis of the protein B, characterised by the rates κ_b and γ_b, respectively. The former reaction is controlled (switched) by specific concentration levels of both protein substrates, while the latter

is controlled by the concentration level of protein B only. The controlling mechanism is specified by the logical combination of *step functions* which are defined for the controlling state variable x_i and its respective *threshold level* θ_i in the following way:

$$s^+(x_i,\theta_i) = \begin{cases} 1, \text{ if } x_i > \theta_i, \\ 0, \text{ if } x_i < \theta_i, \end{cases} \quad s^-(x_i,\theta_i) = 1 - s^+(x_i,\theta_i) \tag{1}$$

In the example above, all the interactions are inhibitory, therefore all the step functions which appear there are negative. Moreover, in this example only the production is controlled. The degradation of both protein substrates is considered spontaneous (always switched on). The presence of step functions allows to partition the phase space of the system into bounded regulatory domains in which the equations are linear. Boundaries of the regulatory domains, so-called *switching domains*, are defined as hyperplanes of lower dimension than the entire phase space (the variable of each missing dimension is considered to be set to the value of particular threshold concentration). If we assume that threshold levels θ_b^1 and θ_b^2 of x_b satisfy the inequality $0 < \theta_b^1 < \theta_b^2 < max_b$ then Fig. 2 shows the partitioned phase space for the example above. It is proved [24,19] that each solution trajectory that starts in a point where each substrate has lower concentration than the respective maximal concentration value, never exceeds the maximal concentration value of any substrate. Moreover, each solution which started outside the box given by maximal values of concentration, at some point enters this box and from that while it never leaves it. Therefore the partitioning of any PL model must be always finite.

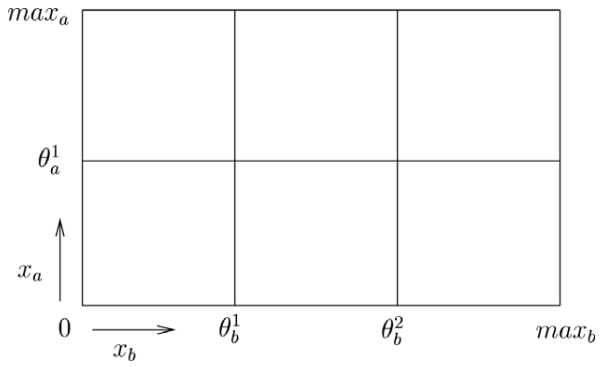


Fig. 2. A phase space for a two-dimensional PL model

Because of the linearity of regulatory domains, phase of each particular state variable in any individual point in a regulatory domain is given by the target *equilibrium* which is computed by dividing the sum of all relevant production rates active in the domain by the sum of all active degradation rates. Concerning our two-dimensional example, target equilibrium for each regulatory domain is denoted by a pair of individual reaction equilibriums (in Fig. 3).

However, from the fact that the step function is discontinuous in the point where concentration is equal to the respective threshold level it follows that the target equilibrium for any switching domain is undefined. The simulation approach of de Jong

et al. [19] deals with this discontinuity by over-approximation based on abstracting the system of differential equations into a system of differential inclusions [26].

2.2 Qualitative simulation in GNA

Given an initial condition a particular PL system cannot be solved numerically until all the reaction rate constants and activation thresholds appearing in the equations are precisely set. Unfortunately, the required information concerning the reaction parameters is usually unknown. This situation motivates development of symbolic simulation methods which allow prediction of possible solutions for equations with unknown reaction rate constants and activation thresholds [27,16]. In the simulation framework GNA (de Jong et al.), the simulation is realised symbolically for given initial conditions and inequalities among individual activation thresholds. Additional information which must be given prior to the simulation concerns approximative positions of reaction equilibriums. In particular, for each possible combination of activated reactions symbolic position of the respective equilibrium must be given. The symbolic position is specified by equilibrium inequalities relatively to the (symbolic) concentration thresholds of the participating substrates. After all this information is given, a state transition system is constructed in which states correspond to regulatory and switching domains of the partitioned phase space. Each transition of this system then simulates how the concentration of participating substrates can evolve from the source state in time. The initial state of the system is determined by the initial conditions of the simulation (the initial domain). The resulting simulation is an over-approximation of the exact solution of the original ODE system.

In the case of our example, the equilibrium inequalities for synthesis reactions of proteins A and B are set to $\theta_a^1 < \frac{\kappa_a}{\gamma_a} < max_a$ and $\theta_b^2 < \frac{\kappa_b}{\gamma_b} < max_b$, respectively. Each of the edges inside the domains in Fig. 3 then symbolises a direction vector which is followed linearly by every solution trajectory when traversing the particular regulatory domain. States of the respective simulation transition system are depicted in Fig. 4. The empty circles denote the states representing regulatory domains whereas the filled circles denote the states representing switching domains.

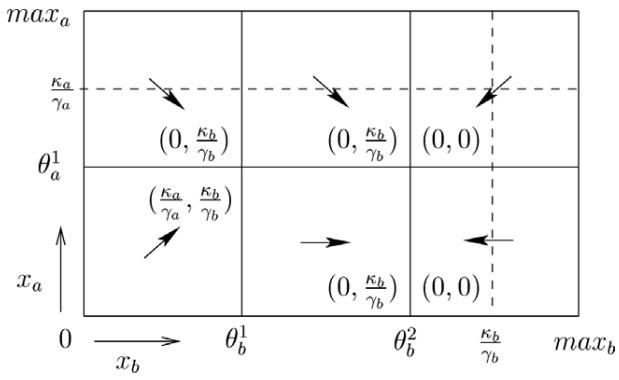


Fig. 3. A piecewise-linear phase space with equilibrium levels

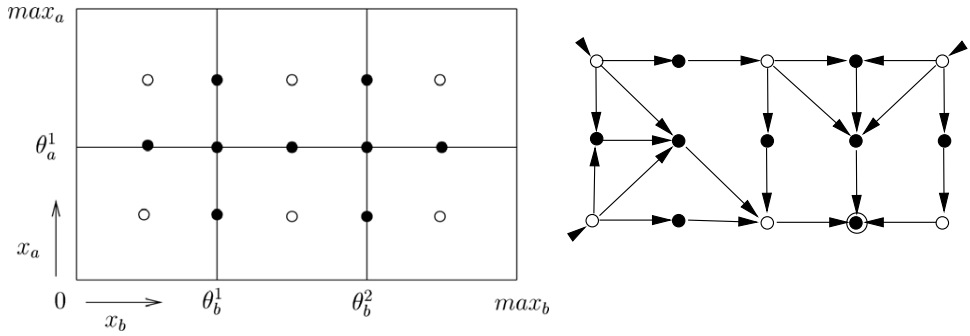


Fig. 4. States of the qualitative simulation and transitions between them

The simulation algorithm and its implementation in GNA is described precisely in [8]. Simulation of the possible system behaviour in regulatory domains is straightforward. In the dimension of each activated variable the solution trajectory tends to the respective equilibrium, while in inactive dimensions the solution tends to minimal concentration. With respect to this rule, transitions to adjacent switching domains are defined.

Transitions from switching states are computed by a more complicated algorithm. By this algorithm GNA implements the idea of over-approximation of differential equations by differential inclusions. This method is employed to overcome the discontinuities introduced by step functions. In particular, to compute successors of a given switching domain, all its neighbouring domains have to be analysed. From the set of all neighbouring domains an appropriate subset is selected according to the rules defined in [16].

The transition system simulating all possible behaviour of the example system is depicted in Fig. 4. The state which is additionally marked by a circle represents the *steady state* of the system. Steady states express a situation when the effects of active reactions are balanced. In other words, in a steady state the activated reactions in the system can unceasingly tend to an equilibrium corresponding to balanced concentration values placed nearby the approximated state. Such states are significant because they show concentration levels of the substrates for which the system can be stable.

Space complexity of the symbolic simulation algorithm grows exponentially with the number of state variables in the modelled system. Therefore for models of real biological systems it is impossible to construct the entire state space representing symbolically all the possible behaviour of the system. GNA allows to limit the state space constructed only to the states reachable from a given initial condition. However, in a large system having around hundred of variables it still can be impossible to handle the part of the state space which is of interest. As the resulting explicitly represented state space is passed from GNA directly to model checking tools NuSMV and CADP for further analysis, the exponential blow up significantly complicates the analysis of large models.

3 GeNeSim

DiVinE Tool (`http://anna.fi.muni.cz/divine`) is a parallel, distributed-memory enumerative model-checking tool for verification of concurrent systems. The tool employs aggregate power of network-interconnected workstations to verify systems whose verification is beyond capabilities of sequential tools. System properties can be specified either directly in Linear Temporal Logic (LTL) or alternatively as processes describing undesired behaviour of systems under consideration (negative claim automata). In fact, the tool can check properties expressible in Linear Time Mu-calculus (Büchi automata).

From the algorithmic point of view, the tool is quite unique. It implements a bunch of novel parallel algorithms for cycle detection (LTL model checking). Besides these, DiVinE Tool includes also an algorithm for distributed state space generation. In addition, it offers an extension for parallel model-checking of finite-state stochastic systems against linear time temporal properties, in particular finite-state Markov chains and Markov decision processes which provide a reasonable semantics for systems that exhibit uncertainty.

GeNeSim is build on the top of the DiVinE library that offers common functions needed to develop a parallel or distributed enumerative model checker. The only extension to the library that was necessary, was the extension of the state generator to a state generator tailored for the specific input provided by GeNeSim GUI. For the structure of GeNeSim implementation and connection to DiVinE see Figure 5.

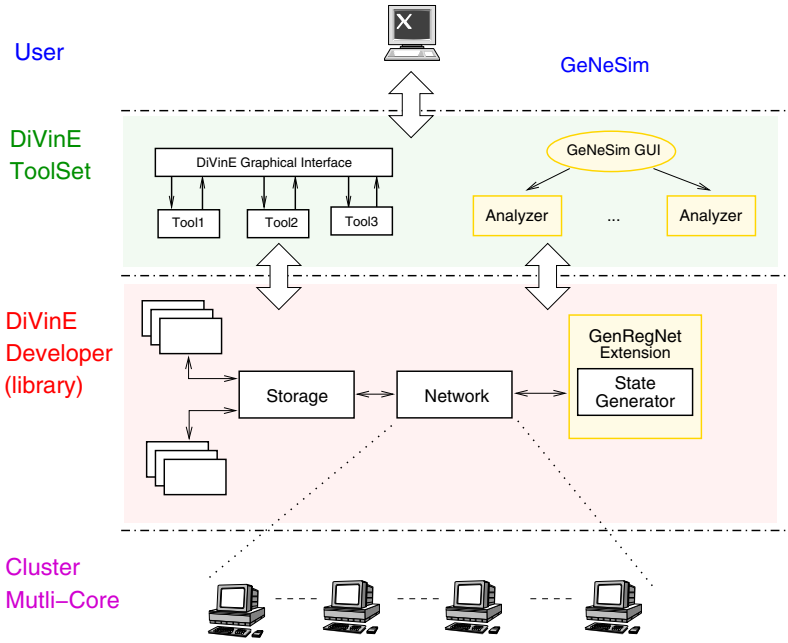


Fig. 5. How is GeNeSim embodied into DiVinE.

3.1 PL model representation and state generator

The central component of the GeNeSim extension of DiVinE is the state generator. The hierarchy of DiVinE state space representation classes is extended with new structures that represent symbolically the PL model in C++. A class diagram of the respective structures is depicted in Fig. 6. A PL model (Genesim System) is represented as a container of variables which occur in respective differential equations. Each state variable contains a set of production rates (may be empty) and a set of degradation rates. With respect to the mathematical specification of PL equations, at least one degradation rate constant must be always defined. Each rate constant is defined as a container of *regulation terms*. A regulation term represents a particular subterm of the equation which is relevant to the activation and deactivation of the respective reaction (represented by the relevant production/degradation rate constant). We support two forms of regulation terms – negative and positive. The positive term is defined as a direct product of step functions whereas the negative term is defined as a negation of a product of step functions. The term $s^-(x_a, \theta_a^1)\, s^-(x_b, \theta_b^1)$ and the term $s^-(x_b, \theta_b^2)$ which appear in the example from the previous section as regulation terms for the production constants κ_a and κ_b, respectively, are instances of positive terms. Both kinds of regulation terms allow all possible types of PL models to be encoded in GeNeSim.

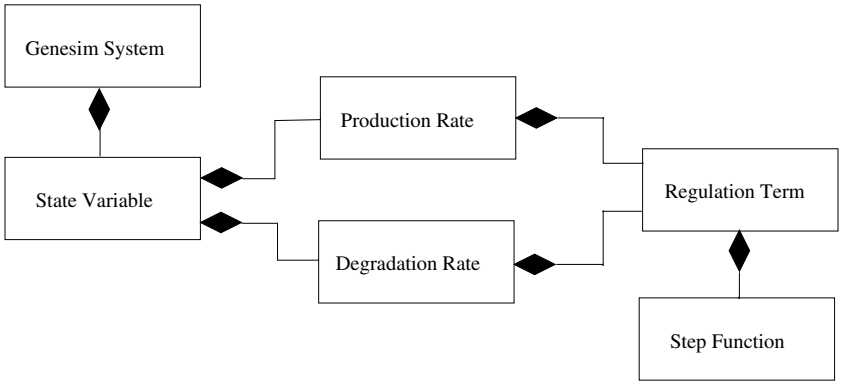


Fig. 6. GeNeSim structures representing the PL model

The GeNeSim state generator implements the DiVinE methods `get_initial()` for determining the initial state of the system and `get_succs()` that computes a container of states which are successors of a given state. By these two methods the implicit representation of the qualitative simulation is encoded. Such a representation allows integration of GeNeSim with DiVinE distributed on-the-fly state space analysis algorithms.

An example of a simulation transition system generated by GeNeSim is depicted in Fig. 7 (on the right). It represents a simulation of the PL model from the example presented in the previous section. The initial threshold inequalities have been set to $x_a > \theta_a^1$ and $x_b < \theta_b^1$. The information included in the individual states denotes the address of the domain and the so-called *direction set property*, respectively. The address of each domain is given for each dimension as the discrete

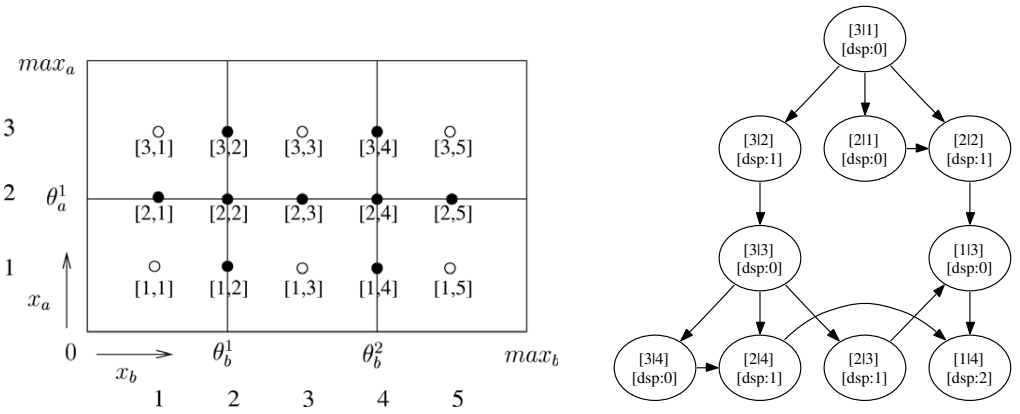


Fig. 7. Example of a simulation generated by GeNeSim

distance from the origin of the symbolic phase space (see Fig. 7). The direction set property (dsp) expresses the information concerning the potential phases in the respective domain computed by approximation. On the one hand, the dsp information is used as a key resource for generating successors of switching domains. On the other hand, computation of the dsp information requires exploration of all the potential neighbouring states. Such exploration takes an indispensable amount of time. Therefore saving of the dsp information into states accelerates generation of the simulation state space. For a particular domain D it can gather the following values:

$$dsp(D) = \begin{cases} 2, \text{ if } D \text{ represents a steady state,} \\ 1, \text{ if the set of phases is empty (only if } D \text{ is switching),} \\ 0, \text{ otherwise.} \end{cases} \quad (2)$$

The value 1 has sense only for states which represent switching domains. The empty phase set signalises that the respective switching domain must be immediately left after it is entered. It symbolises the fact that in such a domain no substrate can keep its concentration constant.

In general, to minimise the memory needed for allocation of states, we have decided to save into states only the mentioned information. All other information is computed on-the-fly whenever it is needed for state space generation, and consequently, during a particular analysis.

3.2 GUI

The GeNeSim GUI is an online web application [3] in PHP using MySQL as data storage. Models can be input manually through structured forms (Fig. 8) or imported automatically from GNA [17].

[3] `http://anna.fi.muni.cz/genesim`

GUI output is in the form of XML, GNA, or C++ data structures directly used by the GeNeSim state space generator in DiVinE (the source code of the model is included before compilation). In close future we plan to implement an XML parser into the GeNeSim state space generator to accept input in XML without the need of recompiling.

In Fig. 8 the PL model from Fig. 1 is shown. Each variable has ordered concentration thresholds (*t_a_1, t_b_1, t_b_2*), production rate (*k_a, k_b*) and degradation rate (*g_a , g_b*) parameters as well as equilibrium inequalities.

For each variable production and degradation rate parameter groups of step functions may be defined. Each group represents a product of step functions and may be either *normal* (positive term) or *negated* (negative term). The two step functions of production rate parameter *k_a* (state-variable x_a) $s^-(x_a, \theta_a^1), s^-(x_b, \theta_b^1)$ are declared as the second two lines in the first positive step group (first line is for adding new step functions). Although in this simple model the degradation rate parameters are not restricted by step functions (always present), in general they can have any number of them as well.

Before exporting a model, all equilibriums must be computed (automatically) and positioned (manually) by specifying the lower threshold of their particular domain position. The number of equilibriums can be potentially exponential to the number of production (p) and degradation (d) rate parameters $\sim 2^{p+d}$.

4 Experiments

To demonstrate for which kind of models the distributed computation may be useful, we have conducted some preliminary experiments checking two different kind of properties in two random extensions of a real model example taken from GNA and in two artificial models with uniformly interconnected genes (in circular manner). The DiVinE distributed algorithm we have employed for state space exploration is the OWCTY algorithm [13] which is an extended enumerative version of the One Way Catch Them Young Algorithm [21].

Model checking analysis of the qualitative simulation state space can be used either to check an existence of a behaviour satisfying a given property in the system or to verify that all the possible simulated behaviour satisfies a given property. In our experiments we focus on the former kind of analysis. In DiVinE, a property is encoded as a Büchi automaton. Especially, all the properties of Linear Temporal Logic (LTL) can be expressed in DiVinE. A sample of LTL properties which can be useful for analysis of qualitative simulation of regulatory networks is given in [3]. Most typical properties of interest when analysing a genetic network concern reachability of states with a given property (e.g., stability). More interesting properties are liveness properties, and even more intricated concentration oscillation properties [6].

An example of a liveness property is description of a system behaviour such that after reaching a particular concentration level of a particular substrate the respective concentration will never fall below that level. Example of such a property expressed

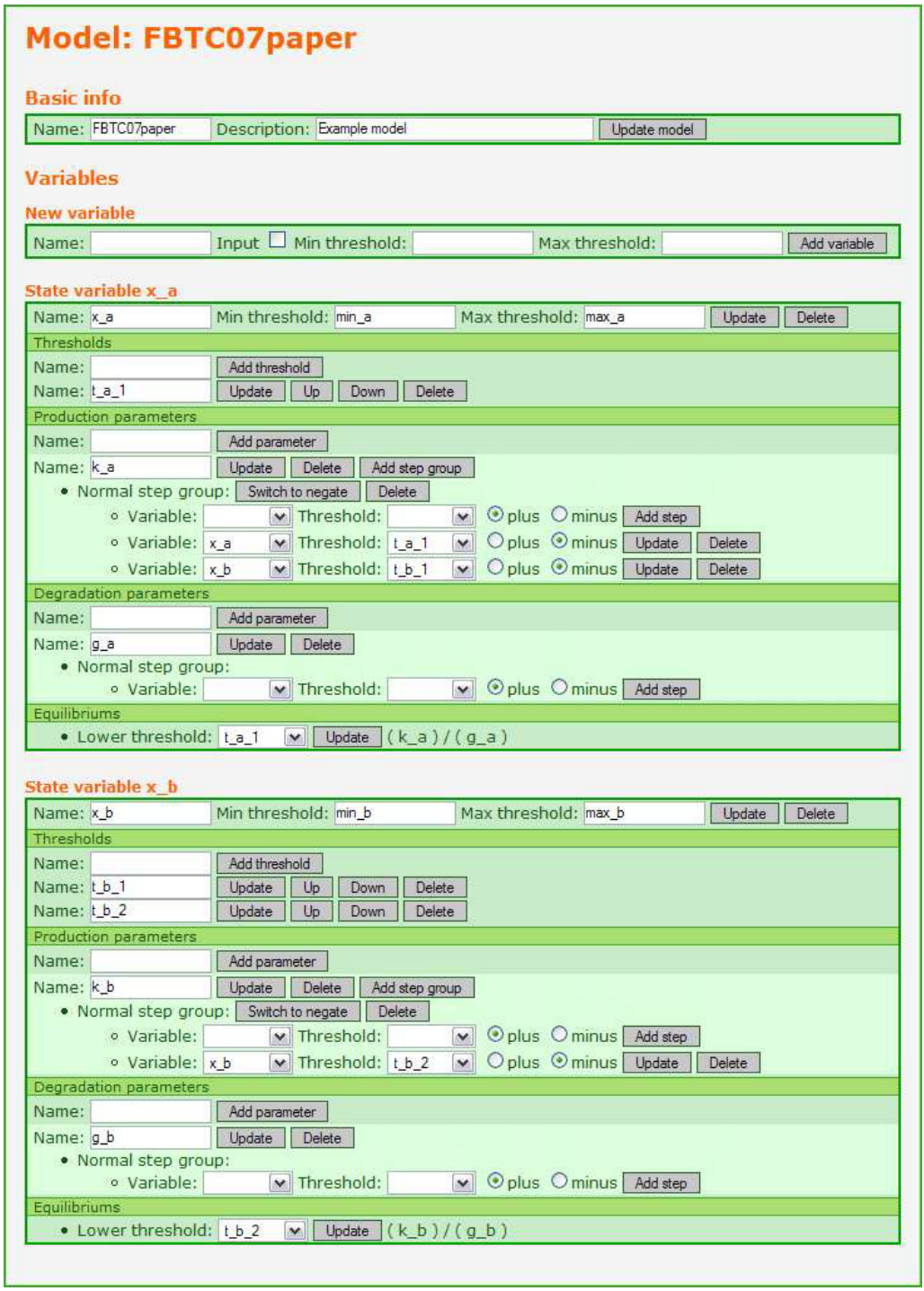


Fig. 8. GeneSim web GUI

in the form of LTL formula is $\mathrm{FG}(x_a > \theta_a^1)$. In particular, the atomic proposition of this formula states that the actual concentration level of the state variable x_a is

strictly greater than the threshold level θ_a^1.

Oscillation properties express cyclic behaviour and are of high importance when analysing and validating models of biochemical processes. Unfortunately, such kinds of properties cannot be expressed directly in LTL neither in CTL [6]. However, this property can be still expressed as a Büchi automaton and therefore we can check a model against this property using DiVinE. In Fig. 9, there is an example of a Büchi automaton describing a property of this kind. The rightmost state of the automaton is an accepting state. It states that whenever concentration of x_a exceeds the threshold θ_a^1, it must cyclically oscillate around θ_a^1.

We have experimented exploring the state space by OWCTY to find if the simulated models provide the behaviour specified by the two properties above. We have randomly modified a four-variable model of genetic regulation in bacteriophage lambda (`thth`) (according to Thieffry and Thomas [34]) providing high mutual interaction of genes. In particular, we have considered random modifications of this model with 6 and 8 state variables. Moreover, we have generated a scalable artificial model (`circ`) with uniform circular interconnection of genes. For this model we have considered settings with 10 and 12 variables. We have run the experiments on a cluster which offers up-to 20 homogeneous workstations. The experiments have been scaled for 1, 5, 10, 15, and 20 nodes.

Behaviour satisfying the liveness property $\mathrm{FG}(x_a > \theta_a^1)$ has been found in all variants of the `thth` model. On the contrary, all variants of the `circ` model do not embody such behaviour. Summary of the most interesting experiments is presented in Table 1. There is showed the number of states which have been explored and the time needed for computation.

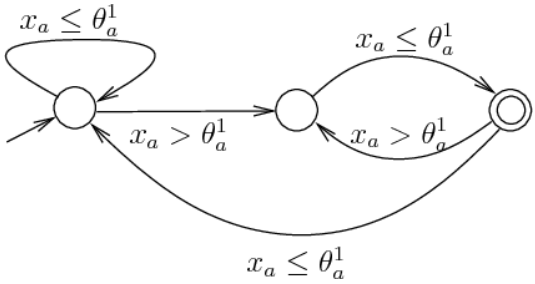


Fig. 9. A Büchi automaton expressing oscillation

Model	States	1 node	5 nodes	10 nodes	15 nodes	20 nodes
thth6	13230	28.40 s	7.07 s	5.38 s	4.68 s	4.59 s
thth8	117390	1249.88 s	309.30 s	190.73 s	163.67 s	151.34 s
circ10	152191	3972.17 s	1003.02 s	683.60 s	555.78 s	494.92 s
circ12	1878527	>72 h	>48 h	>24 h	>12 h	40694.40 s

Table 1
Experiments on checking the liveness property

The oscillation property has not been found in any experimented model. The respective results are given in Table 2.

Model	States	1 node	5 nodes	10 nodes	15 nodes	20 nodes
thth6	7996	20.21 s	5.08 s	3.41 s	2.53 s	2.49 s
thth8	69410	869.99 s	206.12 s	131.10 s	110.06 s	108.6 s
circ10	109008	1807.35 s	524.56 s	354.41 s	285.22 s	253.39 s
circ12	1267000	>24 h	>18 h	>12 h	>10 h	25836.2 s

Table 2
Experiments on checking the oscillation property

5 Conclusions

Results presented in the previous section show that the parallel approach accelerates simulation and model-checking of genetic regulatory networks. Average maximal rate of acceleration achieved by our experiments makes the parallel analysis 7.5 times faster than the sequential analysis with GNA. In particular, the parallel approach enables queries for models having up to 10 state variables to be answered in terms of minutes. Moreover, also larger models (more than 10 variables), which are not satisfactorily tractable by the explicit sequential approach, can be still analysed by the implicit parallel approach on suitably large clusters.

To summarise, our contribution is a demonstration of the use of parallel model-checking for biological systems. In particular, we provide a translation of a piecewise-linear model of a genetic regulatory network into a discrete transition system which serves as an input for the parallel model-checker DiVinE. The approach allows for parallel on-the-fly model-checking of larger networks than is possible by sequential algorithms. The preliminary experiments conducted with the tool confirm good scalability. Though we have focused on qualitative analysis, the DiVinE tool is also able to analyse some stochastic and quantitative properties as well. These extensions together with improvements of the GeNeSim implementation in speeding-up the state space exploration, and that way reaching practicable results for extremely large networks (having around 100 variables), remain for our future work.

References

[1] J. Barnat, L. Brim, I. Černá, P. Moravec, P. Ročkai, and P. Šimeček. DiVinE – A Tool for Distributed Verification (Tool Paper). In *Computer Aided Verification*, volume 4144/2006 of *LNCS*, pages 278–281. Springer, 2006.

[2] G. Batt, C. Belta, and R. Weiss. Model checking genetic regulatory networks with parameter uncertainty. In A. Bemporad, A. Bicchi, and G. Butazzo, editors, *Tenth International Workshop on Hybrid Systems: Computation and Control (HSCC'07)*, volume 4416 of *LNCS*, pages 61–75. Springer, 2007.

[3] G. Batt, C. Belta, and R. Weiss. Model checking liveness properties of genetic regulatory networks. In O. Grumberg and M. Huth, editors, *Thirteenth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07)*, volume 4424 of *LNCS*, pages 323–338. Springer, 2007.

[4] G. Batt, D. Bergamini, H. de Jong, H. Garavel, and R. Mateescu. Model checking genetic regulatory networks using GNA and CADP. In *Eleventh International SPIN Workshop on Model Checking Software (SPIN'04)*, volume 2989 of *LNCS*, pages 158–163. Springer, 2004.

[5] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page, and D. Schneider. Analysis and verification of qualitative models of genetic regulatory networks: A model-checking approach. In L. P. Kaelbling and A. Saffiotti, editors, *Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 370–375, 2005.

[6] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page, and D. Schneider. Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in escherichia coli. In *Thirteenth International Conference on Intelligent Systems for Molecular Biology*, pages 19–28, 2005.

[7] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, M. Page, and D. Schneider. Qualitative analysis and verification of hybrid models of genetic regulatory networks : Nutritional stress response in escherichia coli. In *Eighth International Workshop on Hybrid Systems : Computation and Control (HSCC'05)*, volume 3414 of *LNCS*, pages 134–150. Springer, 2005.

[8] G. Batt, D. Ropers, H. de Jong, M. Page, and J. Geiselmann. Symbolic Reachability Analysis of Genetic Regulatory Networks using Qualitative Abstractions. Technical Report RR-6136, INRIA, 2007.

[9] G. Bernot, J-P. Comet, A. Richard, and J. Guespin. Application of Formal Methods to Biological Regulatory Networks: Extending Thomas' Asynchronous Logical Approach with Temporal Logic. *Journal of Theoretical Biology*, 229(3):339–347, 2004.

[10] James M. Bower and Hamid Bolouri. *Computational modeling of genetic and biochemical networks*. Cambridge : Bradford Book, 2001.

[11] L. Calzone, N. Chabrier-Rivier, F. Fages, and S. Soliman. Machine Learning Biochemical Networks from Temporal Logic Properties. *Transactions on Computational Systems Biology VI*, 4220:68–94, 2006.

[12] L. Calzone, F. Fages, and S. Soliman. BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, 2006.

[13] I. Černá and R. Pelánek. Distributed explicit fair cycle detection (set based approach). In T. Ball and S.K. Rajamani, editors, *Model Checking Software. 10th International SPIN Workshop*, volume 2648 of *Lecture Notes in Computer Science*, pages 49 – 73. Springer Verlag, 2003.

[14] N. Chabrier and F. Fages. Symbolic Model Checking of Biochemical Networks. In N. Chabrier and F. Fages, editors, *Computational Methods in Systems Biology (CMSB'03)*, volume 2602 of *LNCS*, pages 149–162. Springer-Verlag, 2003.

[15] H. de Jong. Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[16] H. de Jong, J. Geiselmann, G. Batt, C. Hernandez, and M. Page. Qualitative simulation of the initiation of sporulation in Bacillus subtilis. *Bulletin of Mathematical Biology*, 66(2):261–300, 2004.

[17] H. de Jong, J. Geiselmann, C. Hernandez, and M. Page. Genetic network analyzer: qualitative simulation of genetic regulatory networks. *Bioinformatics*, 19(3):336–344, 2003.

[18] H. de Jong, J-L. Gouzé, C. Hernandez, M. Page, T. Sari, and Geiselmann J. "hybrid modeling and simulation of genetic regulatory networks: A qualitative approach". In Oded Maler and Amir Pnueli, editors, *Hybrid Systems: Computation and Control, 6th International Workshop, HSCC 2003*, volume 2623 of *LNCS*, pages 267–282. Springer, 2003.

[19] H. de Jong, J-L. Gouz, C. Hernandez, T. Sari, M. Page, and J. Geiselmann. Qualitative simulation of genetic regulatory networks using piecewise-linear models. Technical Report RR-4407, INRIA-Helix, 2002.

[20] S. Williams et.al. The Potential of the Cell Processor for Scientific Computing. In *Proc. of 3rd Conference on Computing Frontiers*, pages 9–20. ACM Press, 2006.

[21] K. Fisler, R. Fraer, G. Kamhi, M. Y. Vardi, and Z. Yang. Is there a best symbolic cycle-detection algorithm? In *Proc. Tools and Algorithms for the Construction and Analysis of Systems*, volume 2031 of *LNCS*, pages 420–434. Springer-Verlag, 2001.

[22] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(No. 25):2340–2381, 1977.

[23] L. Glass. Classification of biological networks by their qualitative dynamics. *Journal of Theoretical Biology*, 54(1):85–107, 1975.

[24] L. Glass. Global analysis of nonlinear kinectics. *Statistical Mechanics, Part B: Time Dependent Processes*, pages 311–349, 1977.

[25] L. Glass, C. Hill, and T. Mestl. Nonlinear dynamics of gene and neural networks. In *Nonlinear Phenomena in Biological and Physical Systems*, pages 249–262. Indian National Science Academy, 2000.

[26] Jean-Luc Gouzé and Tewfik Sari. A class of piecewise linear differential equations arising in biological models. Technical Report RR-4207, INRIA, 2001.

[27] B. Kuipers. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press, 1994.

[28] J.M. McCollum, G.D.Peterson, Ch.D. Cox, and M.L.Simpson. Accelerating Gene Regulatory Network Modeling Using Grid-Based Simulation. In *SIMULATION 2004*, volume 80, pages 231–241. SAGE, 2004.

[29] J.S. Meredith, S.R. Alam, and J.S. Vetter. Analysis of a Computational Biology Simulation Technique on Emerging Processing Architectures. In *Proc. of Parallel and Distributed Processing Symposium*, pages 1–8. IEEE, 2007.

[30] T. Mestl, E. Plahte, and S.W.Omholt. A mathematical framework for describing and analysing gene regulatory networks. *Journal of Theoretical Biology*, 176(2):291–300, 1995.

[31] L. Popova-Zeugmann, M. Heiner, and I. Koch. Time petri nets for modelling and analysis of biochemical networks. *Fundam. Inform.*, 67(1-3):149–162, 2005.

[32] M. Schaub, Th. Henzinger, and J. Fisher. Qualitative networks: a symbolic approach to analyze biological signaling networks. *BMC Systems Biology*, 1:4, 2007.

[33] C. P. Sosa, T. Milledge, J. McAllister, and G. Z. Milledge. Life Sciences Molecular Dynamics Applications on the IBM System Blue Gene Solution: Performance Overview. Technical report, IBM, 2006.

[34] D. Thieffry and R. Thomas. Dynamical Behavior of Biological Regulatory Networks. *Bulletin of Mathematical Biology*, 57(2):277–297, 1995.