Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 299 (2013) 85–100

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Formal Analysis of Oscillatory Behaviors in Biological Regulatory Networks: An Alternative Approach

Usman Rauf   Sheema Sameen   Antonio Cerone

UNU-IIST — International Institute for Software Technology
United Nations University, Macau SAR China

Abstract

In the realm of system biology, the study of regulatory networks leads biologists to the development of increasingly large, detailed and complex models. These complex models, replicating the dynamics of cell processes, are then analyzed using different approaches to obtain predictions. Genetic oscillations play a main role in the activity of signal transduction by maintaining the cascade of internal biochemical reactions with the extracellular environment. Molecular alterations in the performance of such behavioral rhythms can lead to severe pathological problems, e.g. cancer. Different formal approaches have been proposed to analyze Biological Regulatory Networks (BRNs) Such approaches mainly involve the use of non-functional and Binary Decision Diagrams (BDDs) based model checkers for the analysis of irregular structured BRNs, and dense time concept for the modeling of BRNs. Computational Tree Logic (CTL) based analysis of BRNs is not suitable for identifying cyclic (oscillatory) behaviors in irregular structures and the use of Linear Temporal Logic (LTL) for the analysis of multistability is not viable. Morover, the reachability problem becomes undecidable in case of dense time modeling. In order to address these issues, we use *delays* and *Minsky machines* to observe the oscillatory behavior and to overcome the limitation of LTL for the analysis of multistable states. To demonstrate our approach, we consider two different case studies: Pseudomonas aeruginosa and P53-Mdm2 feedback loop.

Keywords: Biological Regulatory Networks (BRNs), Model Checking, Binary Decision Diagrams (BDDs), Computational Tree Logic (CTL)

1 Introduction

Cells are dynamic systems of complex interacting networks in which proteins, genes and small regulatory molecules play together in a programmed manner to perform multiple tasks in an organism. Genes are the informative subunits of DNA and they decode instructions in form of proteins. When a gene is switched on, information flows from genetic to proteomic level as a complex processes of transcription and translation. Some proteins have the function of regulating the expression of genes

[1] Email: {usman; sheema; antonio}@iist.unu.edu

http://dx.doi.org/10.1016/j.entcs.2013.11.008

by turning them on or off. This process of interaction, between genes and protein regulatory elements, establishes a Biological Regulatory Network (BRNs). BRNs often contain feedback loops in order to impose a controled mechanism intended to maintain an optimal concentration of proteins in a cell [20]. This make it difficult to predict the dynamics (behaviors) of BRNs. The study of cellular dynamics is very critical for the understanding of intrinsically evolving morphological characters involved in progression towards a disease state. As in case of the p53-mdm2 feedback loop, oscillatory behavior is considered as the major cause of cancer initiation and progression.

A variety of formalisms have been proposed to analyze BRNs since 1960. De Jong identifies ten different formalisms proposed in literature, which are complemented with simulation techniques [11]. The proposed formalisms include directed graphs, Bayesian networks, stochastic equations, boolean networks and their generalizations, Ordinary/Partial Differential Equations (ODEs/PDEs), and rule-based formalisms. These formalisms can be categorized into four main modeling frameworks, which include state of the art quantitative modeling, qualitative modeling, hybrid modeling and Piecewise Linear Differential Equations (PLDEs) based modeling.

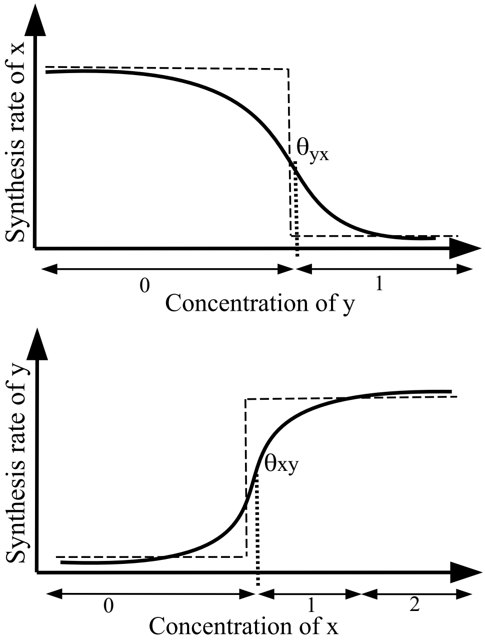


Fig. 1. Sigmoid curves showing the activation of y (below) and inhibition of x (above). The levels 0, 1, 2 represent the discrete (qualitative) concentrations and corresponding discrete abstraction of sigmoid behavior.

Qualitative approaches include boolean logic and the *Kinetic logic* introduced by René Thomas. Boolean logic based modeling approach suffers from different problems which limit their use in such analysis. For example, it only deals with two levels, i.e., 0 and 1. Whereas considering the complexity of biological entities, their modeling demands more than two levels for accurately capturing their dynamics. *Kinetic logic* is a multivalued logic, which allows us to closely approximate the sigmoid nature (c.f. Fig. 1) of biological entities.

BRNs are often represented symbolically by biologists as directed graphs or logical feedback circuits [6,18]. The behavior of these interactive graphs and circuits is then analyzed in order to infer the dynamics of the system [19]. A limitation of this modeling approach is that it does not facilitate the understanding of network dynamics as the number of intertwined biological entities increases.

Quantitative approaches based on *differential equations (ODEs and PDEs)* have also been used extensively for the analysis of BRNs. Tyson et al. [21,22] used coupled differential equations to model a complex network of protein interactions that control the activities of cyclin-dependent kinases. They introduced the notions of bifurcation and bistability in their model and their results appear aligned with experimentation. Lev et al. [12] used a quantitative approach (differential equation based modeling) to model the behavior of the p53-Mdm2 loop. They analyzed the presence of oscillations (*cyclic behavior*) in the p53-Mdm2 feedback loop (*which is important for keeping p53 function under tight control*), and observed that the major dependence of oscillation period is on the delay in the p53-dependent induction of Mdm2.

The more accurate approximation of the sigmoid behavior of biological entities is piecewise linear approximation (c.f. Fig. 2). PLDEs are used in order to model the behavior of each entity. However, this approach requires a large amount of data regarding metabolite concentrations. Moreover, for both ODEs and PLDEs approaches, some of the required biological data may not be available [9].

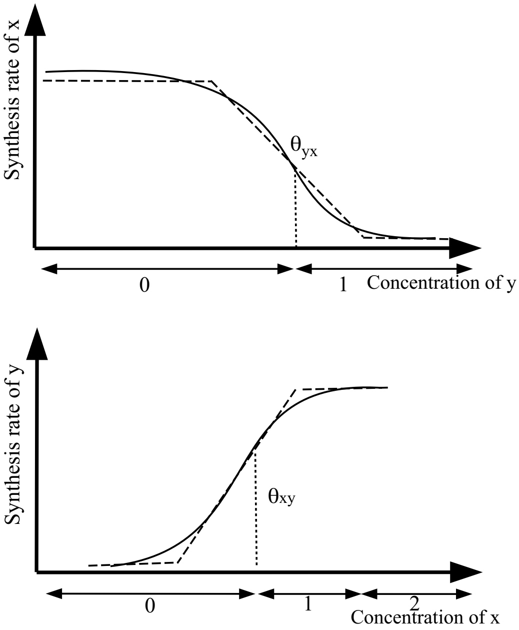


Fig. 2. Piecewise linear approximation of sigmoid curves.

Recently, BRNs have been described as concurrent systems, which paves the way for the application of formal verification techniques in the field of systems biology [7,17]. Bernot et al. [5] have applied formal methods to the field of system biology. Their contributions includes the formal representation of BRNs as asynchronous system of interacting biological entities in the BDD based functional model checker SMV and specification of biological properties in *Computational Tree Logic* (CTL). Finally they applied their approach to a case study: th analysis of mucus production in Pseudomonas aeruginosa.

Two major problems exist regarding the use of formal verification techniques in system biology: First, the use of CTL for the analysis of oscillatory behaviors, whereas it is not effective to characterize cycles in a Kripke structure, and the use of LTL for the analysis of multistable states *(LTL is based on path formulas and cannot express that at some instant along any execution it would be possible to extend the execution in this or that way [13,4,15])*. Second, the use of a dense time concept for the modeling of real time systems [15,8,2].

We extend the approach proposed by Rauf et al. [16], based on the explicit-state functional model checker SPIN [10], using delays and *Minsky* machines (c.f. Fig. 5)[14]. This enables us to analyze different oscillatory/multistable state behaviors that are characterized by different set of delays. We use discrete time to overcome the undecidability problem that occurs while using dense time. The rest of the paper is organized as follows: Section 2 discusses the concepts of timed model, delays and *Minsky* machine in detail and presents our extension to the approach by Rauf et al. [16]. Section 3 illustrates the validation of our work with the help of two biological case studies. Section 4 describes the limitations of the proposed methodology. Section 5 summarizes the developed contributions and places them in the context of the current use of formal verification techniques in the field of systems biology.

2 Modeling Approach

2.1 Model of a Biological Entity

In the context of the *Check and Fire* model [16], a biological entity (e.g. protein or gene) may be considered as an automaton which receives an input (level of predecessors) from interacting neighbors, changes its internal state in response to it, and produces an output depending on a unique threshold level (Θ) (c.f. Figure 1). Formally, a set of m biological entities $\mathcal{N}$ can be expressed as a set of interacting automata and each entity (A_i) may take any positive value in a range. Any entity A_i may have any possible discrete concentration levels.

$$\mathcal{N} = \{A_1, A_2..A_m\};$$

$$A_i = \{0, ..n_i\} \ where\ i \in \{1, ..m\}$$

The possible states for the regulatory network $\mathcal{A}$ are then defined as the cartesian product of all interacting biological entities:

$$\mathcal{A} = A_1 \times A_2 \times A_3 \times ... \times A_m$$

2.2 Modeling of Regulatory Interactions

An excitatory (resp. inhibitory) interaction ($A_1 \xrightarrow{+} A_2$) (resp. $\langle A_1 \xrightarrow{-} A_2 \rangle$) is active when a protein's level is equal to or above a specific threshold level Θ. We also associate a threshold (Θ_{12}) to each interaction ($A_1 \xrightarrow{\Theta_{12}} A_2$). Given an excitatory (resp. inhibitory) interaction ($A_1 \xrightarrow{\Theta_{12}} A_2$), A_1 is called the activator of A_2 if $A_1 \geq \Theta_{12}$ (resp. $A_1 < \Theta_{12}$) for the excitatory interaction (resp. inhibitory interaction). Formally, an interaction between any two biological entities (A_x *and* A_y*)* (c.f. Figure 2) is defined as follows:

$$A_x \xrightarrow{\Theta_{xy}} A_y \ \textit{where} \ \Theta_{xy} \in \{1, ..n_i\}$$

$$A_x,\, A_y \in \mathcal{N};\, x,\, y \in \{1, ..m\}$$

Classical automata can model sequencing of actions in a regulatory network. This temporal sequencing gives *"no qualitative information about delays between actions"* [4]. It is therefore not possible to observe multistable states corresponding to different delays. Contrary to the Kripke structure where each state is labeled with a set of atomic propositions, in a labeled transition system the transitions are labeled with single actions. In this research, the concept of Kripke structure is used in the modeling of biological entities and and extended by using *Minsky* machines to replicate the dynamics of *Linear Hybrid Automata* (LHA). A Linear Hybrid Automata (LHA) is a modified graph that is labeled with a finite set of clock variables, called clocks. Clocks are different from usual variables, as their access is limited: clocks may only be inspected, and reset to zero. After each transition clocks are reset to zero and then start increasing their value implicitly as time progresses. Intuitively, clocks can be considered as stopwatches that can be started and checked independently of one another. Conditions on the values of the clocks are used as enabling conditions (i.e., guards) of actions: only if the condition is fulfilled is the action enabled and capable of being taken; otherwise, the action is disabled. Conditions which depend on clock values are called clock constraints [3].

Definition 2.1 [Linear Hybrid Automata] LHA is a tuple $\mathcal{H} = \langle$S, C, *Act*, CC(C), $\hookrightarrow$, *Inv*, s_0 $\rangle$ in which;

- S is a finite set of states ;
- $s_0 \in$ S is the initial state;
- C is a finite set of clocks;
- *Act* is a finite set of actions;
- CC(C) is a finite set of constraints over clocks;
- *Inv*: S $\rightarrow CC(C)$ is an invariant-assignment function which assigns constraints (over clocks) to corresponding states;
- $\hookrightarrow \subseteq S \times CC(C) \times Act \times 2^C \times S$;

In LHA edges of the graph are labeled with tuples $(g,\ \alpha,\ D\ )$, where $g \subseteq CC(C)$, which is a set of constraints over clocks, $\alpha \in Act$ is an action to be performed if guards are satisfied and $D \subseteq C$ a set of clocks. The interpretation of $s \overset{g,\alpha,D}{\hookrightarrow} s^{'}$ is that an automaton can move from state s to state $s^{'}$ when clock constraint g holds. Besides, when moving from s to $s^{'}$, any clock in D is reset to zero and action α is performed. Function Inv assigns to each state a state invariant that specifies how long the system may stay there. For state s, $Inv(s)$ constrains the amount of time that may be spent in s. That is to say, location s should be left before invariant $Inv(s)$ becomes invalid. If this is not possible — as there is no outgoing transition enabled — no further progress is possible. The notion of time in *LHA* is dense and a clock may assume any rate in given range (0–1). The concept of dense time becomes computationally complex (*there are infinite points in range (0–1)*) as several clocks are associated with the system. To overcome this limitation, we introduced the concept of *discrete time* by using a 2-counter machine. A counter machine can have a set of control states S: $\{s_0,\ s_1,....,\ s_q\}$; and a set of constraints, which defines the rules for transition from one control state to another, whereas s_0 is the initial state of the counter.

Definition 2.2 [2-Counter Minsky Machine] A 2-counter Minsky machine $\mathcal{M}$ is defined as a mathematical function C_i as follows:
C_i*:* $J(i) \rightarrow J(i)$, where $J(i) \subset \mathbb{N}$ and $i \in \{1, 2\}$, such that it evolves according to the following rules/constraints:

$$C_i(0) = 1;$$
$$C_i(|\mathrm{J(i)}| - 1) = |\mathrm{J(i)}| - 2, \text{ where } |\mathrm{J(i)}| \text{ represents the cardinality of J(i);}$$
$$C_i(k) \in \{\mathrm{k{+}1;\ k{-}1}\} \text{ where } 0 < \mathrm{k} < |\mathrm{J(i)}| - 1;$$

A *Minsky* machine with two counters, can jump to next state by having an increment/decrement. As an example, Figure 3, shows the transition of counters $(C_1(J(1))$ and $C_2(J(2)))$ from one state to another under the rules/constraints expressed above.

We integrated the concept of *2-counter Minsky machine* and delays *(time required to change the state of the system)* (c.f. Figure 4) into the *Check and Fire model* [16] to define *Regulatory Network Transition System (RNTS).*

2.3 Regulatory Network Transition System (RNTS)

A *Regulatory Network Transition System (RNTS)* is defined as a *6-tuple*:
$A_{\mathcal{M}}$ =(S, s_0, $C_i(J(i))$, CC($C_i(J(i))$), $\hookrightarrow$, δ):

- S is a finite set of states of regulatory network;
- $s_0 \in$ S is the initial state of automata;
- $C_i(J(i))$ is a 2-counter machine with values in $\mathbb{N}$;
- CC($C_i(J(i))$) is a set of constraints over the counters;

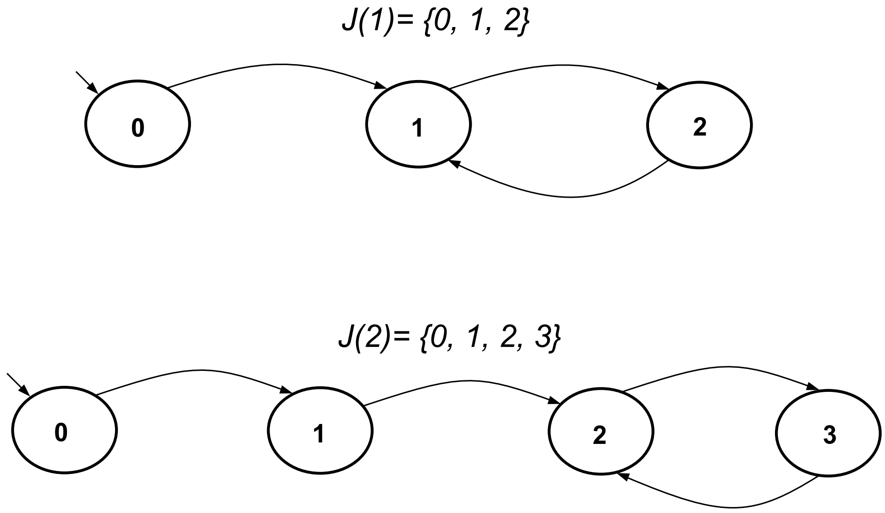


Fig. 3. Example of counters: C_1 such that $J(1) = \{0, 1, 2\}$ and $C_1(1) = 2$; C_2 such that $J(2) = \{0, 1, 2, 3\}$, $C_2(1) = 2$ and $C_2(2) = 3$.

- $\hookrightarrow$ is a finite set of transition such that: $\hookrightarrow \subseteq (S \times \mathbb{N} \times \mathbb{N})^2 \times \mathrm{CC}(C_i(J(i)))$;
- δ is a finite set of transition rules which maps $\mathrm{CC}(C_i(J(i)))$ to set of transition $\hookrightarrow$;

We associate a 2-counter *Minsky* machine with each biological entity; one counter controls the delay of activation (d_{px}) and the second counter controls the delay of inhibition (d_{nx}) (c.f. Figure 4). Constraints over counters work as guards over transitions $\hookrightarrow \subseteq (S \times \mathbb{N} \times \mathbb{N})^2 \times \mathrm{CC}(C_i(J(i)))$, a transition from one control state to another control state is only fired when all guards evaluate to true. The semantics of a transition in *RNTS* can be written as follows:

- $(s, c_1, c_2) \xrightarrow{c_1 = d_{px};\ d_{px} < d_{nx}} (s', c'_1, c_2)$, *Reset counters* where $s, s' \in S$, $c'_1 \in C_1(J(1))$ and $c_2 \in C_2(J(2))$
- $(s, c_1, c_2) \xrightarrow{c_2 = d_{nx};\ d_{nx} < d_{px}} (s'', c_1, c'_2)$, *Reset counters* where $s, s'' \in S$, $c_1 \in C_1(J(1))$ and $c'_2 \in C_2(J(2))$

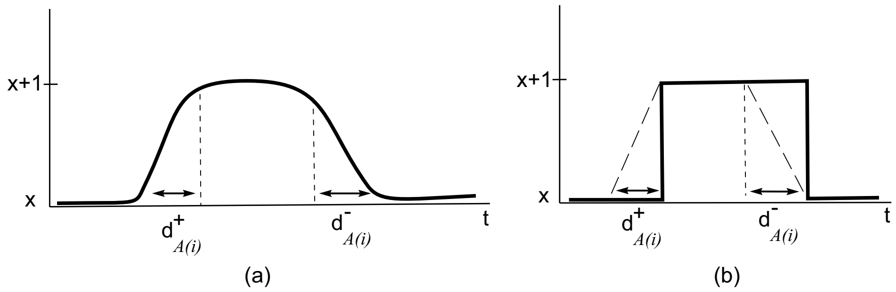


Fig. 4. (a) actual evolution of a biological entity; (b) discrete model enriched with counter machines.

In regulatory network transition system, the biological machinery (*Minsky* machine) controls (increase/decrease) the corresponding protein concentration on the basis of activation/inhibition signals. We use two types of parameters $d^+_{A_i}(x)$ and $d^-_{A_i}(x)$, to represent the time delay required to change the expression level of a biological entity A_i from x to $x+1$ and from $x+1$ to x, as shown in Figure 4. Then we add, to each biological entity A_i, a 2-counter *Minsky* machine $\mathcal{M}$ (with counters (C_1 and C_2)) whose slope at state μ is $\alpha_{A_i}(\mu)$. At a given state μ, if $\alpha_{A_i}(\mu) = +1$ (resp.$\alpha_{A_i}(\mu) = -1$), then, when c_i reaches $d^+_{A_i}(\mu_{A_i})$ (resp. $d^-_{A_i}(\mu_{A_i})$), the level of A_i becomes $\mu_{A_i}+1$ (resp. $\mu_{A_i}-1$) and finally counters c_1 and c_2 are reset. The extended framework is given in Figure 5, sketches and summarizes the SPIN based formal analysis of BRNs. The gray shaded boxes in this figure represent the main contributions of the paper, which are fundamental for the analysis of oscillatory behaviors.

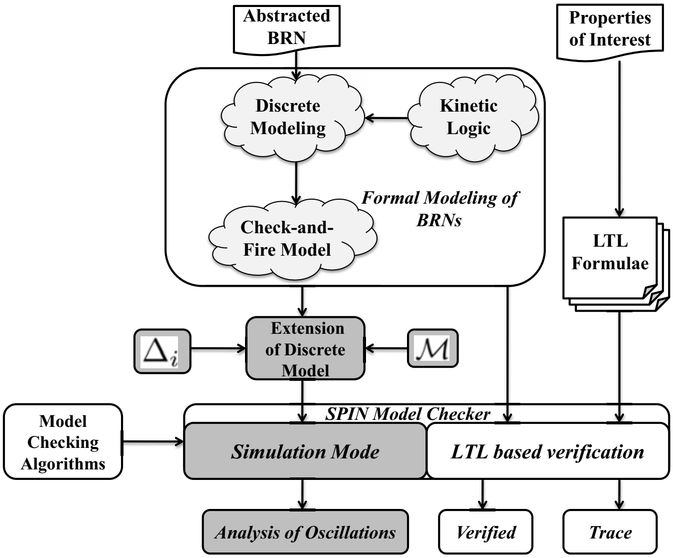


Fig. 5. Extension of discrete modeling framework enriched with Δ_i (delays) and $\mathcal{M}$ (counter machines). For more details regarding our framework readers may refer elsewhere [16].

3 Case Studies

3.1 Timed Modeling of Pseudomonas Aeruginosa

Pseudomonas aeruginosa is a pathogen, which is found in different lung syndromes such as cystic fibrosis. The main cause of the respiratory deficiency in patients of cystic fibrosis is mucus production. The regulatory network which controls the mechanism of mucus production is shown in Fig. 6. *AlgU* is the main regulator of mucus production and it favors its own production while another gene inhibits it. The regulatory network of mucus production is simplified by a regulatory graph in Fig. 6, where x represents gene *AlgU* (or its protein) and y represents the inhibitor protein of *AlgU*. To illustrate the effectiveness of our Regulatory Network Tran-

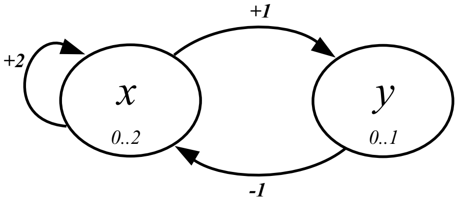


Fig. 6. BRN of Pseudomonas aeruginosa.

sition System (RNTS), we apply it to the case study of Pseudomonas Aeruginosa to verify the constraints over different behaviors (i.e. oscillations and steady stable states). The corresponding hybrid model is shown in Fig. 7. Table 1 contains details about few constraints and associated behaviors. The first constraint is satisfied when the system behaviour is oscillatory. The first part of constraints $(d_{px0} < d_{py0})$ is responsible for transition (0,0)⟶(1,0). The second part $(d_{px1} > d_{py0})$ of the first constraint switches the system as follows: (1,0)⟶(1,1). Inhibitor of gene AlgU is

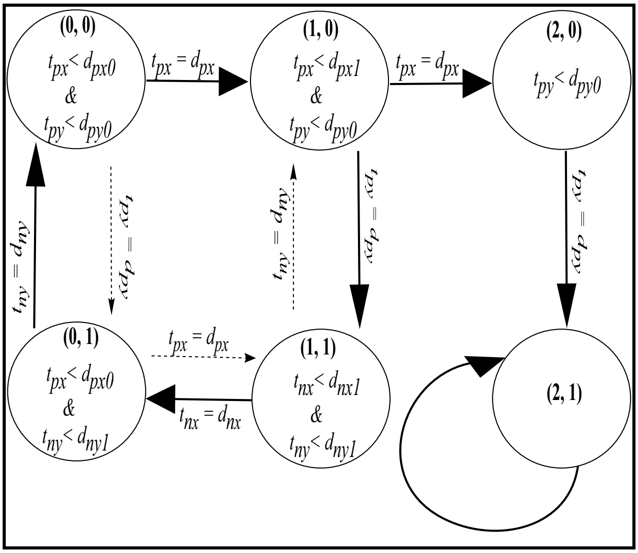


Fig. 7. Hybrid model of P. aeruginosa (bold arrows represent transitions of the discrete model); t_{px}, t_{py}(for activation)$/t_{nx}$, t_{ny} (for inhibition) are Minsky machines associated with X and Y; d_{px0}, d_{py0}/d_{nx1}, d_{ny1} are the delays required to change the concentration of genes X and Y; at a qualitative level 0–1/1–0 correspondingly; and d_{px1} is the delay required to change the concentration of gene X from level 1 to 2.

Constraint	**Satisfied**	**Corresponding behavior**
$(d_{px0} < d_{py0}) \wedge (d_{px1} > d_{py0})$	YES	Oscillation
$(d_{px0} < d_{py0}) \wedge (d_{px1} < d_{py0})$	YES	Stable steady state

Table 1
Constraints and corresponding behavior

activated in state (1,1), which tends to regulate the concentration of AlgU to the minimal level (0), and system proceeds toward the state (0,1). In state (0,1), the concentration level of gene AlgU (resource/activator of gene Y) is below its threshold (+1), which minimizes the concentration level of gene Y and system returns to its initial state (0,0). At any instance of time, condition ($d_{px1} < d_{py0}$) may cause the system to be in configuration (2,0), a state in which AlgU favors its own production, and finally the system leads toward the deadlock or disease state (2,1). Our results are in accordance with Ahmad *et al.* [1], where the authors used a Hybrid model checking tool (Hytech) to obtain the same results.

3.2 P53-Mdm2 Feedback Loop

Tumor suppressor p53 plays a crucial role in cellular damage or stress conditions by arresting cell cycle or programmed cell death. It maintains molecular integrity of the cell by controling the DNA damage that may lead to multiple malignancies. Mutations in the p53 are known as universal trait in oncogenesis for more than 50% human tumors [12]. In response to stress the p53 activity does not grow in a steady manner, but it fluctuates in oscillatory behavior to maintain specific concentration of p53 in cell. These fluctuations are regulated by the mdm2 regulatory gene, responsible for the stability of p53. The mdm2 concentration (itself) is controlled by p53 in an auto-regulatory manner (negative feedback loop). This negative feedback loop is extremely important for the molecular oncological behavior of the cells in case of DNA damage. It is reported that in case of mice, mutations in mdm2 cause death. Regardless of this, if the activation of mdm2 exceeds a normal level, it can cause abnormal suppression of p53 and thereby lead the system towards cancer progression (without damaging gene p53). In this twofold affair mdm2 not only acts as a repressor of p53 (as its transcriptional factor) but also manages p53 concentration by proteolytic degradation. On the other side, p53 performs as a promoter of mdm2 for accelerating its production. Both these genes work together in a circuit to form a strict control mechanism that regulates the p53 responses in normal and stress conditions. Although the importance of the p53-Mdm2 loop is widely recognized, the rules which govern its dynamics need further elaboration. Figure 8 shows abstracted BRN involving the p53-Mdm2 feedback loop. We applied our methodology to the BRN involving the p53-Mdm2 feedback loop and analyzed its dynamics.

3.3 Result and Analysis

P53 prevents the cell from damage due to genotoxic variations, which are responsible for cancer. The identification of physical conditions and genomic aberrations, involved in the progression towards disease state, are very critical in the quest of unraveling carcinogenic mechanisms. The most significant cyclic behavior of the p53-Mdm2 feedback loop was analyzed by applying our timed modeling approach. Obtained results confirmed the already examined in vivo and in vitro behaviors of this mechanism (c.f. Fig. 9) [12]. The oscillatory mechanism is not only involved

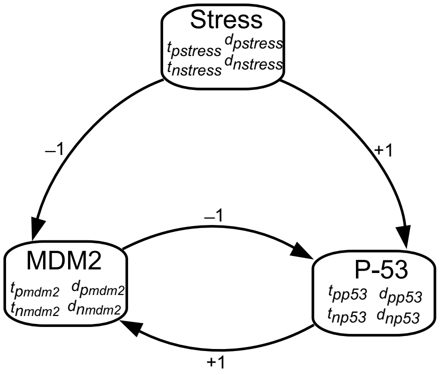


Fig. 8. BRN involving p53-Mdm2 feedback loop; $t_{pstress}$, t_{pp53}, t_{pmdm2}(for activation)/ $t_{nstress}$, t_{np53}, t_{nmdm2} (for inhibition) are Minsky machines associated with *Stress*, *P-53* and *MDM2*; d_{pp53}, d_{pmdm2} / d_{np53}, d_{nmdm2} are the delays required to change the concentration of *Stress*, *P-53* and *MDM2* from qualitative level 0–1 / 1–0 correspondingly; $d_{pstress}$ / $d_{nstress}$ are the durations for Stress signal.

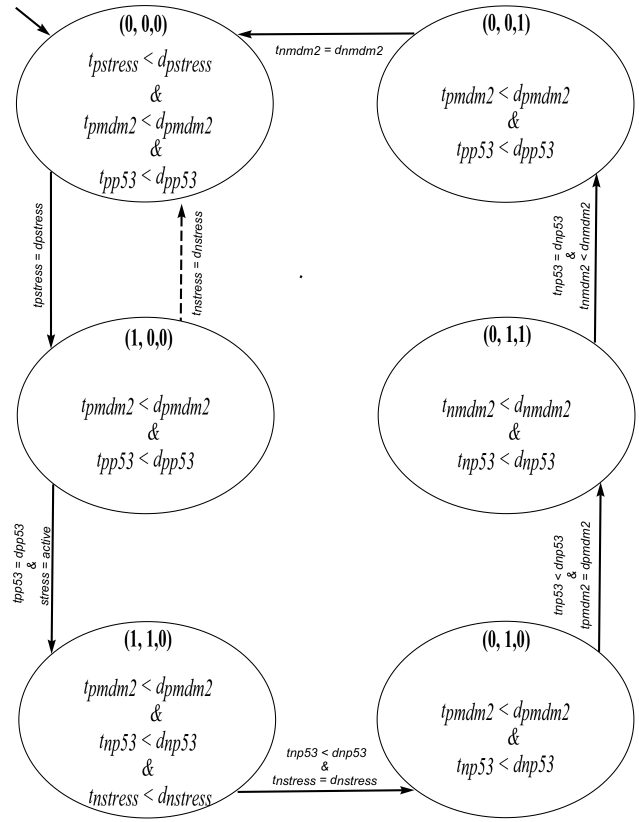


Fig. 9. Analyzed cyclic behavior using the timed modeling approach. Values appear in the following order: Stress signal, P-53, Mdm2 respectively (bold arrows represent transitions of the discrete-time model) [23].

in the behavior of switching on/off gene p53 but also maintains the overall cellular concentration of active p53 in tightly and orderly controlled fashion (under normal conditions). It was observed that in response to stress signals the body reacts by

Delays	**Period of Oscillation**
$dpp5301 = 4$, $dnp53 = 9$, $dpmdm2 = 5$, $dnmdm2 = 9$, $dpstress01 = 3$;	112 time-steps (c.f. Fig 10)
$dpp5301 = 8$, $dnp53 = 9$, $dpmdm2 = 5$, $dnmdm2 = 9$, $dpstress01 = 3$;	178 time-steps (c.f. Fig 11)
$dpp5301 = 4$, $dnp53 = 9$, $dpmdm2 = 8$, $dnmdm2 = 9$, $dpstress01 = 3$;	123 time-steps (c.f. Fig 12)

Table 2
Effect of delays on period of oscillation: *dpp5301* corresponds to the delay in change of concentration level of P53 from 0 to 1 and *dpmdm2* corresponds to the delay in change of concentration level of Mdm2 from 0 to 1.

generating repeated pulses or signals (*e.g cyclic/oscillatory behavior*) for the problem fixation until the damage is repaired effectively. The rapid induction of p53 in stress response is vital for the disease control and any delay in this process affects the period of oscillation. We observed that minor changes in delay (of *P-53* induction from 4 to 8 time steps) may result in very large period of oscillation *(Table 2: Oscillation period increased from 112 time steps to 178 time steps)*. Change in other delays ($dpmdm2$, $dnmdm2$, $dpstress$, $dnstress$) have negligible effect on the period of oscillation. To the best of our knowledge, this is the very first time that period of oscillation has been analyzed using a formal approach rather than simulation. Table 2 shows how the period of oscillation fluctuates corresponding to different changes in delays. The snapshots of the verification procedures are given below as Figure 10, 11, and 12.

4 Limitations

Apart from all the benefits, our modeling approach also has some limitations in terms of state space explosion and complexity. An explicit-state model checker explores the state space of the system by running the model. In SPIN, if a single channel or a local variable changes its value then the program (SPIN model of the BRN) is considered to be in a new state, whether the whole system (BRN) changes its state or not. Consequently, it may be difficult to overcome the state space explosion as the number of intertwined biological entities increases while dealing with the complex system. Our tesults show how the state space of a BRN increases exponentially when a discrete model is enriched with delays and counter machines (c.f. Table 3).

5 Conclusion and Future Work

We introdused a formalism based on the semantics of *Regulatory Network Transition System (RNTS)* to analyze oscillatory behaviors of complex Biological Regulatory Networks (BRNs) and to overcome the problem of undecidability in the case of dense time modeling. The formalism is used to model two different case studies:

Discrete Modeling	Timed Modeling
Pseudomonas aeruginosa	***Pseudomonas aeruginosa*** *(dpx01=4, dpx12 = 6, dnx= 2, dpy=4, dny= 5)*
States = 12	States = 5009
Transition = 14	Transition = 9654
Memory = 2.539 Mb	Memory = 3.027 Mb
Time = 0.3 sec	Time = 0.4 sec
P53-Mdm2 Feedback-Loop	***P53-Mdm2 Feedback Loop*** *(dpp5301=4, dnp53= 9, dpmdm2=8, dnmdm2= 9, dpstress01=3;)*
States = 11	States = 8311962
Transition = 13	Transition = 40633634
Memory = 1.53 Mb	Memory = 1023.984 Mb
Time = 0.46 sec	Time = 3 sec

Table 3
State space and time complexity information about discrete modeling and timed modeling.

pseudomonas aeruginosa and a BRN involving the p53-Mdm2 feedback loop. Code script and figures concerning analysis are available for download [23]. Our results show that hybrid modeling based on the concept of discrete time paves the way for researchers to use this modeling framework for the analysis of infectious diseases and to make predictions about future drug designs. Previously, simulation based analysis has been used extensively for the observation of oscillatory behaviors, which requires a lot of information regarding all unknown biological parameters (e.g. post-translational modifications), whereas these are not yet available for all organisms. The key advantage of our formal approach over simulation based analysis is that it does not require any quantitative information regarding biological parameters, and yet it provides important results with acceptable cost in terms memory and time (4–5 minutes for the analysis of P53-Mdm2 feedback loop) as well as modeling effort excluding (*98-lines* of Promela code).

Acknowledgement

This work has been supported by Macao Science and Technology Development Fund, File No. 07/2009/A3, in the context of the EAE project.

References

[1] Ahmad, J., G. Bernot, J.-P. Comet, D. Lime and O. Roux, *Hybrid modelling and dynamical analysis of gene regulatory networks with delays*, ComPlexUs **3** (2007), pp. 231–251.

[2] Alur, R. and T. A. Henzinger, *Real-time logics: complexity and expressiveness*, in: *IEEE Symposium on Logic in Computer Science (LICS)*, 1990, pp. 390–401.

[3] Ben-Ari, M., "Principles of the Spin Model Checker," Springer, 2008, 1 edition.

[4] Bérard, B., M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci and Ph. Schnoebelen, "Systems and Software Verification. Model-Checking Techniques and Tools," Springer, 2001.

[5] Bernot, G., J. paul 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 (2004), pp. 339–347.

[6] Chaouiya, C., E. Remy, P. Ruet and D. Thieffry, *Qualitative modelling of genetic networks: From logical regulatory graphs to standard Petri nets*, Applications and Theory of Petri Nets, Lecture Notes in Computer Science (2004), pp. 137–156.

[7] Fisher, J. and T. A. Henzinger, *Executable cell biology*, Nature Biotechnology **25(11)** (2007), pp. 1239–1250.

[8] Furia, C. A., D. Mandrioli, A. Morzenti and M. Rossi, *Modeling time in computing: A taxonomy and a comparative survey*, ACM Comput. Surv. **42** (2010).

[9] Gebert, J., N. Radde and G.-W. Weber, *Modeling gene regulatory networks with piecewise linear differential equations*, European Journal of Operational Research **181** (2007), pp. 1148–1165.

[10] Holzmann, G. J., *The model checker SPIN*, IEEE Transactions on Software Engineering **23** (1997), pp. 279–295.

[11] Jong, H. D., *Modeling and simulation of genetic regulatory systems: A literature review*, Journal of Computational Biology **9** (2002), pp. 67–103.

[12] Lev, R. Maya, L. A. Segel, U. Alon, A. J. Levine and M. Oren, *Generation of oscillations by the p53-Mdm2 feedback loop: a theoretical and experimental study.*, Proc Natl Acad Sci U S A **97** (2000), pp. 11250–11255.

[13] Mateescu, R., P. T. Monteiro, E. Dumas and H. Jong, *Computation tree regular logic for genetic regulatory networks*, in: *Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis*, ATVA '08 (2008), pp. 48–63.

[14] Minsky, M. L., *Recursive unsolvability of post's problem of "tag" and other topics in theory of turing machines*, Annals of Mathematics **74** (1961), pp. 437–455.

[15] Monteiro, P., "Towards an integrative approach for the modeling and formal verification of biological regulatory networks," Ph.D. thesis, Instituto Superior Tecnico - Technical University of Lisbon, Portugal Universita Claude Bernard Lyon 1, France (2010).

[16] Rauf, U., U. Siddique, J. Ahmad and U. Niazi, *Formal modeling and analysis of biological regulatory networks using spin*, in: *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2011, pp. 304 –308.

[17] Regev, A. and E. Shapiro, *Cells as computation*, Nature **419, 343** (2002).

[18] Remy, E. and P. Ruet, *From minimal signed circuits to the dynamics of boolean regulatory networks.*, Bioinformatics (Oxford, England) **24** (2008), pp. i220–226.

[19] Siebert, H. and A. Bockmayr, *Temporal constraints in the logical analysis of regulatory networks*, Theoretical Computer Science **391** (2008), pp. 258–275.

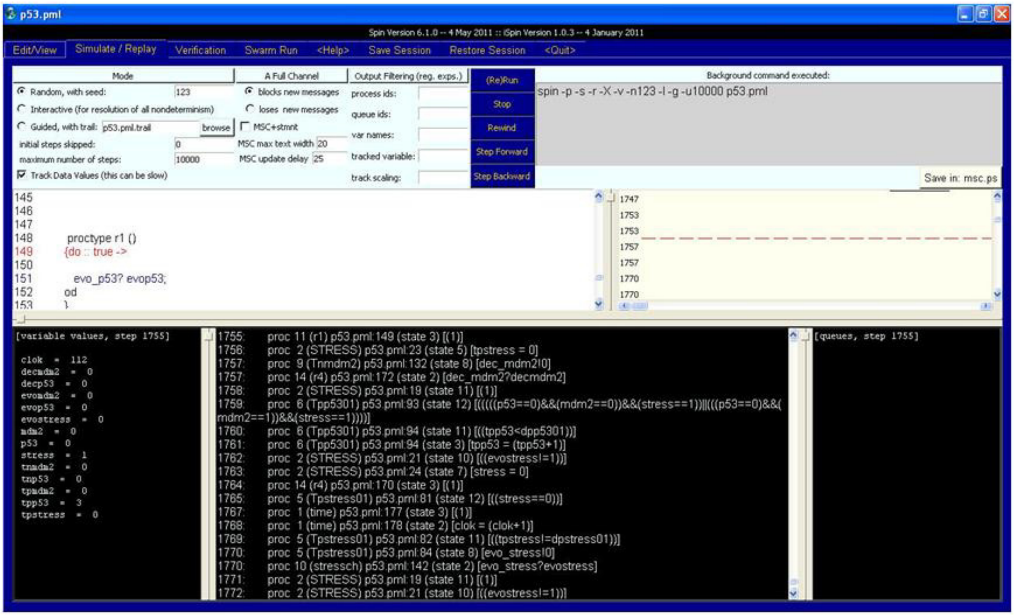
[20] Thomas, R. and R. D'Ari, *Biological feedback*, CRC Press, Boca Raton, FL (1990).

[21] Tyson, J. J., K. Chen and B. Novak, *Network dynamics and cell physiology*, Nat Rev Mol Cell Biol **2** (2001), pp. 908–916.

[22] Tyson, J. J., A. Csikasz-Nagy and B. Novak, *The dynamics of cell cycle regulation*, Bioessays **24** (2002), pp. 1095–1109.

[23] Usman Rauf, S. S. and A. Cerone, *Formal analysis of oscillatory behaviors in biological regulatory networks: An alternative approach*, ProMeLa code and figures are availble at: http://iist.unu.edu/publication/formal-analysis-oscillatory-behaviors-biological-regulatory-networks-alternative-approac .

6 Appendix



● Random, with seed: 123
○ Interactive (for resolution of all nondeterminism)
○ Guided, with trail: p53.pml.trail browse
initial steps skipped: 0
maximum number of steps: 10000
☑ Track Data Values (this can be slow)

● blocks new messages
○ loses new messages
□ MSC+stmnt
MSC max text width 20
MSC update delay 25

process ids:
queue ids:
var names:
tracked variable:
track scaling:

```
spin -p -s -r -X -v -n123 -l -g -u10000 p53.pml
```

```
      proctype r1 ()
     {do :: true ->

       evo_p53? evop53;
     od
     }
```

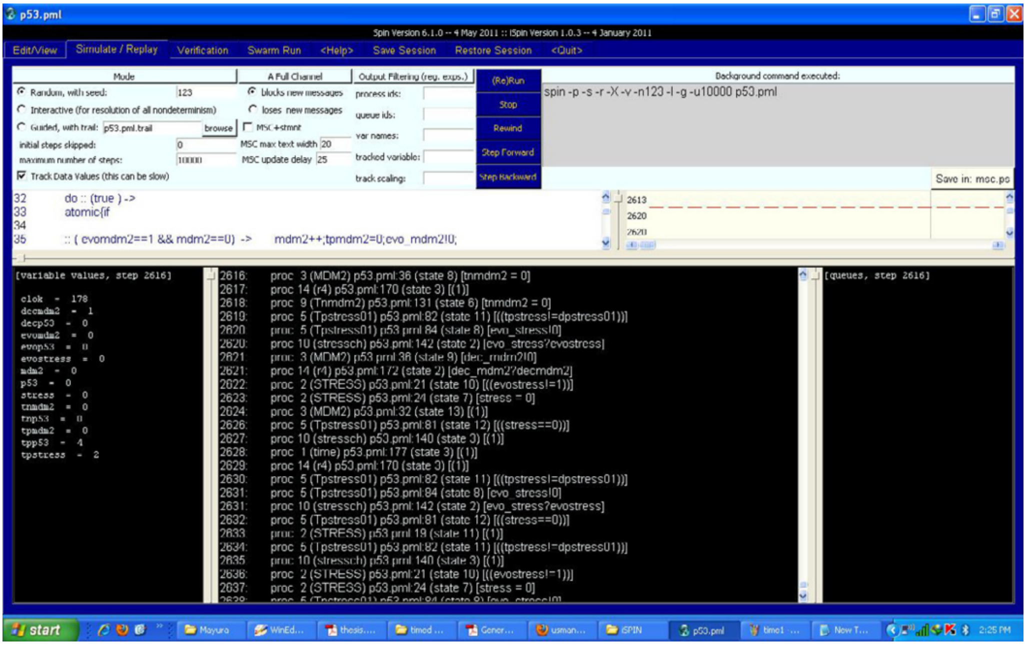
```
[variable values, step 1755]

 clok  =  112
 decmdm2  =  0
 decp53  =  0
 evomdm2  =  0
 evop53  =  0
 evostress  =  0
 mdm2  =  0
 p53  =  0
 stress  =  1
 tnmdm2  =  0
 tnp53  =  0
 tpmdm2  =  0
 tpp53  =  3
 tpstress  =  0
```

```
1755:	proc 11 (r1) p53.pml:149 (state 3) [(1)]
1756:	proc  2 (STRESS) p53.pml:23 (state 5) [tpstress = 0]
1757:	proc  9 (Tnmdm2) p53.pml:132 (state 8) [dec_mdm2!0]
1757:	proc 14 (r4) p53.pml:172 (state 2) [dec_mdm2?decmdm2]
1758:	proc  2 (STRESS) p53.pml:19 (state 11) [(1)]
1759:	proc  6 (Tpp5301) p53.pml:93 (state 12) [((((((p53==0)&&(mdm2==0))&&(stress==1))||(((p53==0)&&(
mdm2==1))&&(stress==1)))))]
1760:	proc  6 (Tpp5301) p53.pml:94 (state 11) [(((tpp53<dpp5301)))]
1761:	proc  6 (Tpp5301) p53.pml:94 (state 3) [tpp53 = (tpp53+1)]
1762:	proc  2 (STRESS) p53.pml:21 (state 10) [(((evostress!=1)))]
1763:	proc  2 (STRESS) p53.pml:24 (state 7) [stress = 0]
1764:	proc 14 (r4) p53.pml:170 (state 3) [(1)]
1765:	proc  5 (Tpstress01) p53.pml:81 (state 12) [(((stress==0)))]
1767:	proc  1 (time) p53.pml:177 (state 3) [(1)]
1768:	proc  1 (time) p53.pml:178 (state 2) [clok = (clok+1)]
1769:	proc  5 (Tpstress01) p53.pml:82 (state 11) [(((tpstress!=dpstress01)))]
1770:	proc  5 (Tpstress01) p53.pml:84 (state 8) [evo_stress!0]
1770:	proc 10 (stressch) p53.pml:142 (state 2) [evo_stress?evostress]
1771:	proc  2 (STRESS) p53.pml:19 (state 11) [(1)]
1772:	proc  2 (STRESS) p53.pml:21 (state 10) [(((evostress!=1)))]
```

```
[queues, step 1755]
```

Fig. 10. 112 time-steps [23]



● Random, with seed: 123
○ Interactive (for resolution of all nondeterminism)
○ Guided, with trail: p53.pml.trail browse
initial steps skipped: 0
maximum number of steps: 10000
☑ Track Data Values (this can be slow)

● blocks new messages
○ loses new messages
□ MSC+stmnt
MSC max text width 20
MSC update delay 25

process ids:
queue ids:
var names:
tracked variable:
track scaling:

```
spin -p -s -r -X -v -n123 -l -g -u10000 p53.pml
```

```
     do :: (true ) ->
     atomic{if

     :: ( evomdm2==1 && mdm2==0) ->      mdm2++;tpmdm2=0;evo_mdm2!0;
```

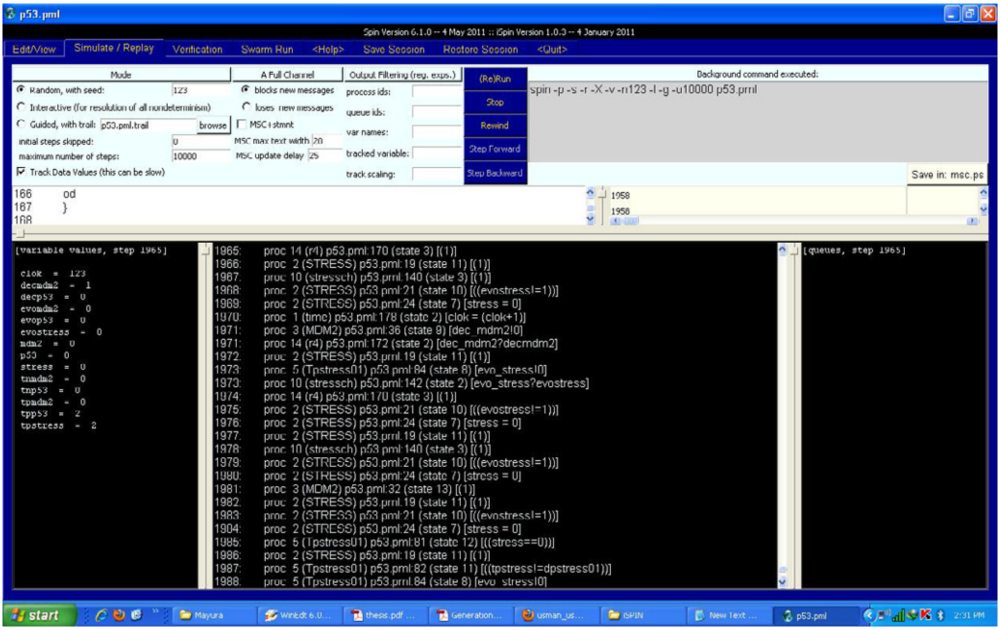
```
[variable values, step 2616]

 clok  =  178
 decmdm2  =  1
 decp53  =  0
 evomdm2  =  0
 evop53  =  0
 evostress  =  0
 mdm2  =  0
 p53  =  0
 stress  =  0
 tnmdm2  =  0
 tnp53  =  0
 tpmdm2  =  0
 tpp53  =  4
 tpstress  =  2
```

```
2616:	proc  3 (MDM2) p53.pml:36 (state 8) [tnmdm2 = 0]
2617:	proc 14 (r4) p53.pml:170 (state 3) [(1)]
2618:	proc  9 (Tnmdm2) p53.pml:131 (state 6) [tnmdm2 = 0]
2619:	proc  5 (Tpstress01) p53.pml:82 (state 11) [(((tpstress!=dpstress01)))]
2620:	proc  5 (Tpstress01) p53.pml:84 (state 8) [evo_stress!0]
2620:	proc 10 (stressch) p53.pml:142 (state 2) [evo_stress?evostress]
2621:	proc  3 (MDM2) p53.pml:36 (state 9) [dec_mdm2!0]
2621:	proc 14 (r4) p53.pml:172 (state 2) [dec_mdm2?decmdm2]
2622:	proc  2 (STRESS) p53.pml:21 (state 10) [(((evostress!=1)))]
2623:	proc  2 (STRESS) p53.pml:24 (state 7) [stress = 0]
2624:	proc  3 (MDM2) p53.pml:32 (state 13) [(1)]
2626:	proc  5 (Tpstress01) p53.pml:81 (state 12) [(((stress==0)))]
2627:	proc 10 (stressch) p53.pml:140 (state 3) [(1)]
2628:	proc  1 (time) p53.pml:177 (state 3) [(1)]
2629:	proc 14 (r4) p53.pml:170 (state 3) [(1)]
2630:	proc  5 (Tpstress01) p53.pml:82 (state 11) [(((tpstress!=dpstress01)))]
2631:	proc  5 (Tpstress01) p53.pml:84 (state 8) [evo_stress!0]
2631:	proc 10 (stressch) p53.pml:142 (state 2) [evo_stress?evostress]
2632:	proc  5 (Tpstress01) p53.pml:81 (state 12) [(((stress==0)))]
2633:	proc  2 (STRESS) p53.pml:19 (state 11) [(1)]
2634:	proc  5 (Tpstress01) p53.pml:82 (state 11) [(((tpstress!=dpstress01)))]
2635:	proc 10 (stressch) p53.pml:140 (state 3) [(1)]
2636:	proc  2 (STRESS) p53.pml:21 (state 10) [(((evostress!=1)))]
2637:	proc  2 (STRESS) p53.pml:24 (state 7) [stress = 0]
2638:	proc  5 (Tpstress01) p53.pml:84 (state 8) [evo_stress!0]
```

```
[queues, step 2616]
```

Fig. 11. 178 time-steps [23]



Mode
● Random, with seed: 123
○ Interactive (for resolution of all nondeterminism)
○ Guided, with trail: p53.pml.trail browse
initial steps skipped: 0
maximum number of steps: 10000
☑ Track Data Values (this can be slow)

A Full Channel
● blocks new messages
○ loses new messages
☐ MSC+stmnt
MSC max text width 20
MSC update delay 25

Output Filtering (reg. exps.)
process ids:
queue ids:
var names:
tracked variable:
track scaling:

Background command executed:

```
spin -p -s -r -X -v -n123 -l -g -u10000 p53.pml
```

Save in: msc.ps

```
166       od
167       }
168
```

```
1958
1958
```

```
[variable values, step 1965]

 clok  =  123
 decmdm2  =  1
 decp53  =  0
 evomdm2  =  0
 evop53  =  0
 evostress  =  0
 mdm2  =  0
 p53  =  0
 stress  =  0
 tnmdm2  =  0
 tnp53  =  0
 tpmdm2  =  0
 tpp53  =  2
 tpstress  =  2
```

```
1965:    proc 14 (r4) p53.pml:170 (state 3) [(1)]
1966:    proc  2 (STRESS) p53.pml:19 (state 11) [(1)]
1967:    proc 10 (stressch) p53.pml:140 (state 3) [(1)]
1968:    proc  2 (STRESS) p53.pml:21 (state 10) [((evostress!=1))]
1969:    proc  2 (STRESS) p53.pml:24 (state 7) [stress = 0]
1970:    proc  1 (time) p53.pml:178 (state 2) [clok = (clok+1)]
1971:    proc  3 (MDM2) p53.pml:36 (state 9) [dec_mdm2!0]
1971:    proc 14 (r4) p53.pml:172 (state 2) [dec_mdm2?decmdm2]
1972:    proc  2 (STRESS) p53.pml:19 (state 11) [(1)]
1973:    proc  5 (Tpstress01) p53.pml:84 (state 8) [evo_stress!0]
1973:    proc 10 (stressch) p53.pml:142 (state 2) [evo_stress?evostress]
1974:    proc 14 (r4) p53.pml:170 (state 3) [(1)]
1975:    proc  2 (STRESS) p53.pml:21 (state 10) [((evostress!=1))]
1976:    proc  2 (STRESS) p53.pml:24 (state 7) [stress = 0]
1977:    proc  2 (STRESS) p53.pml:19 (state 11) [(1)]
1978:    proc 10 (stressch) p53.pml:140 (state 3) [(1)]
1979:    proc  2 (STRESS) p53.pml:21 (state 10) [((evostress!=1))]
1980:    proc  2 (STRESS) p53.pml:24 (state 7) [stress = 0]
1981:    proc  3 (MDM2) p53.pml:32 (state 13) [(1)]
1982:    proc  2 (STRESS) p53.pml:19 (state 11) [(1)]
1983:    proc  2 (STRESS) p53.pml:21 (state 10) [((evostress!=1))]
1904:    proc  2 (STRESS) p53.pml:24 (state 7) [stress = 0]
1985:    proc  5 (Tpstress01) p53.pml:81 (state 12) [((stress==0))]
1986:    proc  2 (STRESS) p53.pml:19 (state 11) [(1)]
1987:    proc  5 (Tpstress01) p53.pml:82 (state 11) [((tpstress!=dpstress01))]
1988:    proc  5 (Tpstress01) p53.pml:84 (state 8) [evo_stress!0]
```

```
[queues, step 1965]
```

Fig. 12. 123 time-steps [23]