Markov Model Checking of Probabilistic Boolean Networks Representations of Genes

Marie Lluberes[1], Jaime Seguel[2] and Jaime Ramírez-Vick[3]
[1, 2] Electrical and Computer Engineering Department, University of Puerto Rico, Mayagüez, Puerto Rico
[3] General Engineering Department, University of Puerto Rico, Mayagüez, Puerto Rico

Abstract - *Our goal is to develop an algorithm for the automated study of the dynamics of Probabilistic Boolean Network (PBN) representation of genes. Model checking is an automated method for the verification of properties on systems. Continuous Stochastic Logic (CSL), an extension of Computation Tree Logic (CTL), is a model-checking tool that can be used to specify measures for Continuous-time Markov Chains (CTMC). Thus, as PBNs can be analyzed in the context of Markov theory, the use of CSL as a method for model checking PBNs could be a powerful tool for the simulation of gene network dynamics. Particularly, we are interested in the subject of intervention. This refers to the deliberate perturbation of the network with the purpose of achieving a specific behavior. This is attained by selectively changing the parameters in a node or set of nodes so that the network behavior can be controlled.*

Keywords: Gene Regulatory Network, Probabilistic Boolean Networks, Markov-chain, intervention, model-checking algorithms.

1 Introduction

The genome encodes thousands of genes whose products enable cell survival and numerous cellular functions. The amounts and the temporal pattern in which these products appear in the cell are crucial to the processes of life. A gene regulatory network is the collection of molecular species and their interactions, which together modulate the levels of these gene products. The dynamics due to both internal and external interactions constitute the state of a system. With the aid of Computer Science and Statistics, the study of gene regulatory network dynamics has become more feasible, and several models have been developed to simulate such dynamics. The knowledge of the intrinsic mechanisms that govern the network could provide the means to control its behavior. It is because of this that the development of an automated system capable of effectively simulating the behavior of a gene regulatory network may also provide the knowledge to alter such behavior in order to achieve a particular state of the system or, on contrary, to prevent or to stop an undesirable behavior. This "guiding" of the network dynamics is referred to as intervention. The power to intervene with the network dynamics has a significant impact in diagnostics and drug design.

Biological phenomena manifest in the continuous-time domain. But, in describing such phenomena we usually employ a binary language, for instance, expressed or not expressed; on or off; up or down regulated. Studies conducted restricting genes expression to only two levels (0 or 1) suggested that information retained by these when binarized is meaningful to the extent that it is remains in a continuous domain [2]. This allows gene regulatory networks to be modeled using a Boolean paradigm. The drawback of using this formalism is that the interactions among genes are hard-wired rules. This unrealistic assumption precludes the self-organizing nature of biological systems and, therefore, mischaracterizes their dynamics. Self-organization gives the system robustness in presence of perturbations, showing spontaneous ordered collective behavior. PBNs and Boolean networks share this quality through the existence of attractors and absorbing states, which act as a form of memory for the system.

PBNs, like Boolean networks, are rule-based. But, unlike the latter, they are not inherently deterministic using multiple rules, or "predictors". This makes PBNs robust in the face of the environmental and biological uncertainty. Markov theory allows us to study the dynamic behavior of PBNs in the context of Markov Chains. They explicitly represent probabilistic relationships between genes, allowing quantification of influence between genes. Because of this, PBNs are better suited than Boolean networks for modeling such systems. Nevertheless, given the exponentially growth in the number of states a gene can be in(2^n states for n genes), answering questions on the best way to reach or avoid particular state(s) may be cumbersome if performed through exhaustion. Model-checking algorithms have the ability of automatically check if a certain condition is met under given specifications. Thus, it could answer questions as the one previously stated efficiently. This would greatly facilitate the intervention or deliberate perturbation of a network to achieve a desired response. This research studies the union between PBNs in

the context of Markov theory and model checking techniques for Continuous-time Markov chains.

2 Model Selection

2.1 Boolean Network Model

A Boolean network is a set of Boolean variables whose state is determined by other variables in the network. Formally:

A Boolean network G(V,F) is defined by a set of nodes $V = \{x_1,...,x_n\}$, *and a list of Boolean functions* $F = (f_1, ..., f_n)$. Each $x_i \in V$, $i=1,..., n$, is a binary variable representing a gene which takes value from $\{0, 1\}$. There are k_i genes assigned to gene x_i, whose value at time t determine the value at time $t+1$ of x_i by means of a Boolean function $f_i \in F$. That is, the mapping j_k: $\{1,...,n\} \rightarrow \{1,...,n\}$, $k = 1, ..., k_i$ determines the "wiring" of gene x_i and we can write [2]:

$$x_i(t+1) = f_i(x_{j_1(i)}(t), x_{j_2(i)}(t), \cdots, x_{j_{k_i}(i)}(t)). \qquad (1)$$

A network with n genes has 2^n states. Each of these states represents the pattern of expression of the individual genes. Pattern expressions are sometimes called gene activity profiles (GAPs). Some of these GAPs are attractors in the sense that the network flow eventually gets trapped in them. They represent the memory of the system. Attractors may be composed by cycles of states. Figure 1 gives an example of a Boolean network.

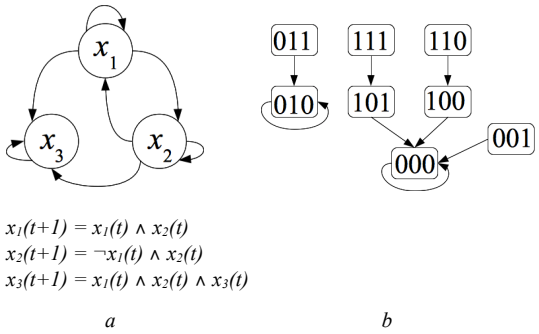


Figure 1. Example of Boolean network
a) Boolean network with three nodes
b) State transition diagram

The relationships between genes are determined from experimental data. A coefficient of determination (COD) is used in this endeavor to discover such associations. The COD measures the quality of a predictor in using an observed gene set to infer a target gene set, in the absence of observations. In order to further illustrate this, let x_i be a target gene, which we wish to predict by observing the set of genes $x_{i1}, x_{i2},..., x_{ik}$. Suppose that $f(x_{i1}, x_{i2},..., x_{ik})$ is an optimal predictor of x_i relative to some error measure ε. Let ε_{opt} be the optimal error achieved by f. Then, the COD for x_i relative to the set $x_{i1}, x_{i2}, ..., x_{ik}$ is defined as:

$$\theta = \frac{\varepsilon_i - \varepsilon_{opt}}{\varepsilon_i} \qquad (2)$$

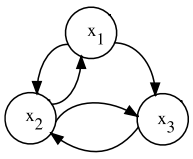
where ε_i is the error of the best (constant) estimate of x_i in the absence of any conditional variables [2].

2.2 PBN Model

The open nature of biological systems brings about a significant uncertainty into the model. One way of coping with this difficulty is to pass the uncertainty to the predictor, by synthesizing a number of good performance predictors. Each one of them contributes its own prediction proportionally to its determinative potential, which is given by the COD. More formally, given genes $V = \{x_1, ..., x_n\}$, we assign to each x_i a set $F_i = \{f_1^{(i)}, ..., f_{l(i)}^{(i)}\}$ of Boolean functions representing the "top" predictors for the target gene x_i. Thus, the PBN acquires the form of a graph $G(V, F)$ where $F = (F_1, ..., F_n)$ [4], and each F_i in F is as previously described. At each point in time or step of the network, a function $f_j^{(i)}$ is chosen with probability $c_j^{(i)}$ to predict gene x_i. Using a normalized COD [2]:

$$c_j^{(i)} = \frac{\theta_j^i}{\sum_{t=1}^{l(i)} \theta_t^i} \qquad (3)$$

where θ_j^i is the COD for gene x_i relative to the genes used as inputs to predictor $f_j^{(i)}$. Figure 2 provides an example of a PBN.



$f_1^{(1)}: x_1(t+1) = \bar{x}_2(t)$ $\quad c_1^{(1)} = 1$

$f_1^{(2)}: x_2(t+1) = \bar{x}_1(t)$ $\quad c_1^{(2)} = 0.3$

$f_2^{(2)}: x_2(t+1) = x_1(t) \wedge x_3(t)$ $\quad c_2^{(2)} = 0.7$

$f_1^{(3)}: x_3(t+1) = x_1(t)$ $\quad c_1^{(3)} = 0.6$

$f_2^{(3)}: x_3(t+1) = \bar{x}_1(t) \wedge x_2(t)$ $\quad c_2^{(3)} = 0.4$

Figure 2. PBN of three nodes and its predictors

At a given instant in time, the predictors selected for each gene determine the state of the PBN. These predictors are contained on a vector of Boolean functions, where the i^{th} element of that vector contains the predictor selected at that time instant for gene x_i. This is known as a *realization* of the PBN. If there are N possible realizations, then there are N possible vector functions, $f_1, f_2, ..., f_N$, each of the form $f_k = (f_{k1}^{(1)}, f_{k2}^{(2)}..., f_{kn}^{(n)})$, for $k = 1, 2, ..., N$, $1 \leq k_i \leq l(i)$ and where $f_{ki}^{(i)} \in F_i$ ($i= 1, ..., n$). In other words, the vector function $\mathbf{f}_k$:$\{0, 1\}^n \rightarrow \{0, 1\}^n$ acts as a transition function

(mapping) representing a possible realization of the entire PBN. (See Figure 3). Thus, we have the matrix K of realizations:

$$K = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_N \end{bmatrix} \tag{4}$$

Assuming independence of the predictors, $N = \prod_{i=1}^{n} l(i)$. Each realization k can be selected with $P_k = \prod_{i=1}^{n} c_{k_i}^{(i)}$. The probability of transitioning from state $(x_1,...,x_n)$ to $(x'_1,...,x'_n)$ is given by [3]:

$$\Pr\{(x_1,\cdots,x_n) \to (x'_1,\cdots,x'_n)\} = \sum_{k=1}^{N} P_k \underbrace{\left[\prod_{i=1}^{n} \left(1 - \left|f_{ki}(x_1,\cdots,x_n) - x'_i\right|\right)\right]}_{\in\{0,1\}} \tag{5}$$

3 Perturbation And Intervention

As an open system, the genome receives inputs from the outside. Such stimuli can either activate or inhibit gene expression; therefore it is necessary for the model of such a system to reproduce this behavior. This is achieved by the inclusion of a realization in the form of a random *perturbation vector* $\gamma \in \{0, 1\}^n$. Lets assume that a gene can get independently perturbed with probability p. Then if $\gamma_i = 1$ the i^{th} gene is flipped, otherwise it is not. For simplicity, we will assume that $\Pr\{\gamma_i = 1\} = E[\gamma_i] = p$ for all $i = 1, ..., n$ (i.e., independent and identically distributed). Let $x(t) \in \{0, 1\}^n$ be the state of the network at time t. Then, the next state x' is given by:

$$x' = \begin{cases} x \oplus \gamma, & \text{with probability } 1 - (1-p)^n \\ \mathbf{f}_k(x_1,\cdots,x_n), & \text{with probability } (1-p)^n \end{cases}, \tag{6}$$

where $\oplus$ is component-wise addition modulo 2, and f_k is the transition function representing a possible realization of the entire PBN, $k = 1, 2, ..., N$ [2]. In presence of perturbation with probability p, the entrances in the state transition matrix are computed by [4]:

$$A(x,x') = \left(\sum_{k=1}^{N} P_k \underbrace{\left[\prod_{i=1}^{n} \left(1 - \left|f_{ki}(x_1,\cdots,x_n) - x'_i\right|\right)\right]}_{\in\{0,1\}} \right) \times (1-p)^n + p^{\eta(x,x')} \times (1-p)^{n-\eta(x,x')} \times 1_{[x \neq x']} \tag{7}$$

Most relevant to our research is the fact that, when performed in a deliberately way, a perturbation constitutes an intervention. We may introduce a perturbation vector for a set of selected genes for the purpose of achieving a desired state, or moving from an undesirable one, on the network. This can be done by perturbing those genes with greater impact on the global behavior, by perturbing a fewer number of genes, or by reaching the desired state as early as possible. In gene interactions, some genes used in the prediction of a target gene have more impact than others, making them more important, or of higher *influence*, thus, identifying these genes is highly relevant. Similarly, we can determine the sensitivity of a particular gene, defining it as the sum of all *influences* acting upon it. The sensitivity, in turn, defines the particular gene stability and independence. In [2, 4] a method to compute influences and sensitivities is given. One of the main benefits of determining influences and sensitivities of genes is that these allow the identification of vulnerable points in the network, or the ones most likely to affect its entire network if perturbed. Highly influential genes can control the dynamics of the network, making it possible to move to a different basin of attraction when perturbed. This kind of information may provide potential targets when an intervention is needed to obtain a desired state of the system.

4 Model-Checking Algorithms

Given a PBN model of a gene regulatory network, we are interested in knowing (in an automated way) if certain state(s) are reachable under particular conditions, or specifications. This is the verification problem, to which model checking is an instance of. Because these are mathematical problems, we formulate our specifications

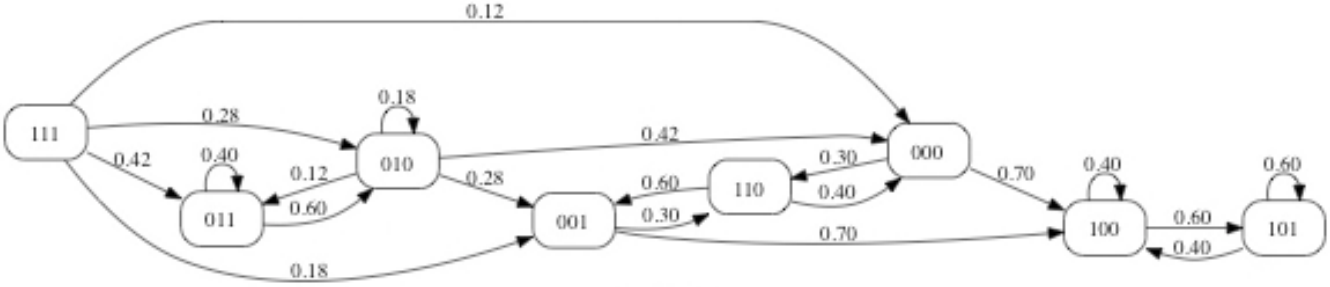


Figure 3. PBN state transition diagram

using mathematical logic. Temporal logics have been crucial in the development of model checking, because of its compact way of expressing correctness properties, and the fact that the Small Finite Model Theorem makes it decidable [5]. Its branching time logic, Computation Tree Logic (CTL) allows us to build compound formulas from the nesting of subformulas. The semantics of temporal logic formulas are defined over a finite transition system (Kripke structure).

The specification of measures of interest over systems is usually done using state-based properties (steady and transient state), due to the difficulty of specifying path-based measures. Continuous Stochastic Logic (CSL) is a probabilistic timed extension of CTL that provides means for specifying measures both state and path-based for Continuous-time Markov chains (CTMC). Numerical methods to model-check CSL over finite-state CTMC are explored in [1].

4.1 Continuous-time Markov chains

The Kripke structure to consider for CSL model checking is a CTMC, where the edges are equipped with probabilistic timing information. Let *AP* be a fixed, finite set of atomic propositions [1]:

A CTMC M *is a tuple (S,* **R***, L) with S as a finite set of states,* **R***: S x S* → ¡≥*0 as the transition matrix, and L : S* → 2^{AP} *as the labeling function*.

Each state $s \in S$ corresponds to a GAP of the PBN. **R** is the transition probability matrix of the state-transition network. Function *L* assigns to each state $s \in S$ the set *L(s)* of atomic propositions $a \in AP$ that are valid in *s*. We allow self-loops by having $\mathbf{R}(s,s) > 0$. The probability that the transition $s \rightarrow s'$ can be triggered within *t* time units is $1 - e^{-\mathbf{R}(s,s')\cdot t}$. The probability to move from a state *s* to state *s'* within *t* time units is given by [1]:

$$\mathbf{P}(s,s',t) = \mathbf{R}(s,s')\cdot(1-e^{-t}) \qquad (8)$$

The probability of moving from a nonabsorbing (with at least one transition out of it) state *s* to *s'* by a single transition is $\mathbf{P}(s,s') = \mathbf{R}(s,s')$. For an absorbing state *s*, $\mathbf{P}(s,s') = 0$ for any state *s'* [1].

For our PBN example (Fig. 2 and 3) the Markov model would have the set of states $S = \{(000), (001), (010), (011), (100), (101), (110), (111)\}$. **R** is an 8 x 8 matrix containing the transition probabilities between states. $AP = \{x_i \in \{0,1\}, i = 1,\ldots, n\}$. $L(x_1 \ldots x_n) = \{x_{1\ldots}x_{n:}\ x_{1\ldots}x_n \in \{0,1\}^n\}$, for instance, $L(011) = (x_1x_2x_3=011)$. An initial distribution α, which can be a state or set of states, is imposed over the PBN. For this particular case, we assume an initial uniform joint distribution. This means each state has the same chance of being the initial state. Taking s_0=(111), a possible sequence of transitions, or computation, is {(111), (001), (100), (101), (100)}.

There are two major types of state probabilities for CTMC:

1. Transient-state probabilities, where the system is observed at a given time instant *t*:
   $\pi^{M}(\alpha,s',t) = \Pr_{\alpha}\{\sigma \in \mathrm{Path}^{M} \mid \sigma@t = s'\}$
2. Steady-state probabilities, where the system is observed when equilibrium has been reached:
   $\pi^{M}(\alpha,s') = \lim_{t\rightarrow\infty} \pi^{M}(\alpha,s',t)$

The two types of measures shown above are state-based. However, we are also interested in the probability on paths through the CTMC obeying particular properties. To the best of our knowledge, suitable mechanisms to measure such properties have not been considered in the literature.

It is worth noting that Binary Decision Diagrams (BDDs), a powerful tool for model checking, are not all that useful in the contexts of PBNs models. What precludes its use is the fact that each state of the PBN, or GAP, contains a string of variables representing genes. As BDDs represents possible transitions for one variable, we would need a BDD for each variable contained in the string. The output would be ramifications of several BDD. As BDDs represents Boolean functions, their values can be directly obtained from the truth table of the predictors.

4.2 Continuous Stochastic Logic

Continuous Stochastic Logic (CSL) provides means to specify state as well as path-based performance and dependability measures for CTMCs in a compact and unambiguous way. This logic is basically a probabilistic timed extension of CTL [1].

Besides the standard steady-state and transient measures, the logic allows for the specification of constraints over probabilistic measures over paths through CTMCs. For instance, we may check the probability of going from state s to state *s'* within *t* time units, avoiding or visiting some particular intermediate states. Four types of measures can be identified:

1. *Steady-state* measures: The formula $S_{\trianglelefteq p}(\Phi)$ imposes a constraint on the probability to be in some Φ state on the long run. For the PBN in the example above, $S_{\geq 0.4}(x_1 \wedge \neg x_2)$ states that there is at least a 40% probability that gene x_1 is expressed and gene x_2 is not expressed when the network reach equilibrium.
2. *Transient* measures: The combination of the probabilistic operator with the temporal operator $\Diamond^{[t,t]}$ can be used to reason about transient probabilities. More specifically, $P_{\trianglelefteq p}(\Diamond^{[t,t]} at_{s'})$ is valid in state *s* if the transient probability at time *t* to be in state *s'* satisfies the bound $\trianglelefteq p$.
3. *Path-based* measures: By the fact that P-operator allows an arbitrary path formula as the argument; much more general measures can be described. An example is the probability of reaching a certain set of states provided that all paths to these states obey certain properties.

4. *Nested* measures By nesting the P and S operators, more complex properties can be specified. These are useful to obtain a more detailed insight into the system's behavior and allow it to express probabilistic reachability that is conditioned on the system being in equilibrium.

The main benefits in using CSL for specifying constraints on measures of interest over CTMCs are[1]:

1. Since the specification is entirely formal, the interpretation is unambiguous. An important aspect of CSL is the possibility of stating performance and dependability requirements over a selective set of paths through a model, which was not possible before.
2. The possibility of nesting steady-state and transient measures provides a means to specify complex, though important measures in a compact and flexible way.

Once we have obtained the model (CTMC M) of the system under consideration, and specified the constraint on the measure of interest in CSL by a formula Φ, the next step is to model check the formula. The model-checking algorithm for CTL that supports the automated validation of Φ over a given state *s* in M, is adapted to these purposes. The basic procedure is as for model checking CTL: in order to check whether state s satisfies the formula Φ, we recursively compute the set *Sat*(Φ) of states that satisfy Φ and, finally, check whether *s* is a member of that set. For the non-probabilistic state operators, this procedure is the same as for CTL [1].

For the purpose of intervention, it would be necessary to know how likely are certain states to reach a steady-state on the network of genes. This information, and with the use of the influences and sensitivities previously explained, would aid in determining the genes that represent the best candidates for reaching a desired condition. For instance, if we want to verify if a particular state reach a steady-state condition with a certain probability, a very high-level algorithm would look as follows:

Input: PBN, state *s*, measure *m*, constraint *c*
Do:
1. Determine Bottom Strongly Connected Components BSCC of PBN.
2. *If s* isn't in some BSCC
   Output "State specified doesn't reach steady state".
   Stop.
3. *Else* continue.
4. Compute transition probabilities to state *s*.
5. Use constraint *c* to compare computed probabilities with *m*.
6. *If* constraint is met with some probability *p*
   Output "The condition is met with probability *p*".
   Stop.
7. *Else*
   Output "The system doesn't meet the desired condition".

Stop.

Given the state-explosion problem that characterizes this kind of model, abstraction is crucial. Bisimulation, the technique that guarantees exact abstraction, has a slight variation called lumping. It has been observed that lumping preserves all CSL formulas [1].

5 Future Work

At the moment, we are using CSL for describing some measurements on PBNs constructed with fictitious data. So far, steady-state measurements have been checked. Next, we have to develop algorithms for the particular cases of steady and transient states, as well as for path-based measurements. Then, we will test them with PBNs built from real data. This, of course, belongs to a feedback loop where results will be used to improve the algorithms. Once we are able to verify with certainty particular conditions against real data, we will work on the process of intervention. For this, we need to check the changes on the dynamics due to particular alterations of the parameters using a vector of perturbation.

6 Conclusions

PBNs make an ideal model representation for genetic networks because the robustness that multiple predictors give them. As Kripke structures representing state transitions of a system, CSL can be used as a model-checking algorithm for CTMC, expanding the traditional state-based measures with the use of path-based probabilistic measures. PBNs can be studied in the context of Markov theory, and Markov chains have been widely used to specify system performance and dependability. Because of this, it is our belief that a model-checking algorithm for CSL can be used to study the dynamics of CTMC representations of PBN used to model genetic regulatory networks in an effective way. Avoiding the matrix-based model, such algorithm would mitigate the impact of the analysis of an exponential size network. Intervention on the network would be attainable, due the information gathered thanks to the algorithm's ability of answering questions about the transition system of the PBN.

The breadth of logic topics that this research evolves through is worth remarking. In its most primitive formulation, relationships between genes can be described with the use of logic connectives from propositional logic. Predicate logic is then used for formulating questions on the state and dynamics of the system. Finally, temporal logic is the basis of the model checking algorithms that answers these questions.

7 Acknowledgements

This research is conducted in part thanks to the support of a RISE-NIH scholarship (1R25GM088023-01A1) granted to the first author.

8 References

[1] Baier, C., Haverkort, B., Hermanns, H. and Katoen, J-P. Model-Checking Algorithms for Continuous-Time Markov Chains. IEEE Transactions on Software Engineering. 2003; 29(6):1-18.

[2] Shmulevich, I., Dougherty, E.R. and Zhang, W. From Boolean to Probabilistic Boolean Networks as Models of Genetic Regulatory Networks. IEEE. 2002; 90(10).

[3] Shmulevich, I., Dougherty, E.R., Kim, S. and Zhang, W. Probabilistic Boolean Networks: a rule-based uncertainty model for gene regulatory networks. Bioinformatics (Oxford, England). 2002; 18(2):261-74

[4] Shmulevich, I., Dougherty, E.R. and Zhang, W. Gene perturbation and intervention in probabilistic Boolean networks. Bioinformatics (Oxford, England). 2002; 18(10):1319-31.

[5] Emerson, E. Allen. The beginning of Model Checking: A personal perspective. Springer. 2008.Volume 5000; 27-45.