

Modeling of Multiple Valued Gene Regulatory Networks

Abhishek Garg, Luis Mendoza, Ioannis Xenarios and Giovanni DeMicheli

Abstract—In silico modeling of Gene Regulatory Networks has gained a lot of attention recently as it gives a very powerful tool to experimental biologists to gather the knowledge gained from different biological experiments and understand the dynamics of the overall system. One of the key dynamics that is often interesting is the steady states of the networks which biologically corresponds to the cellular states. In our previous paper, we gave an efficient method called GenYsis to compute these steady states in Boolean representation of Gene Regulatory Network. It has been observed that protein may be expressed at more than two level of expression. This may result in different cellular outcomes. To address this issue, we present here a multiple-level modeling methodology that allows us to be more accurate. In this paper we extend our software GenYsis to model gene regulatory networks where each node in the network may take multiple values.

I. INTRODUCTION

Modeling the qualitative behaviour of gene regulatory networks by representing gene expression as *ON* or *OFF* can capture many interesting biological properties as shown in [Mendoza and Xenarios, 2006, Mendoza, 2006, Mendoza et al., 1999, Thomas, 1991, S nchez and Thieffry, 2003]. In our previous paper we gave an efficient method called genYsis [Garg et al., 2007] for modelling regulatory networks as Boolean networks using Binary Decision Diagrams (BDDs). GenYsis was shown to scale well with large size of the gene regulatory network when represented as Boolean networks. In this paper, we extend genYsis to multiple valued networks where gene expressions are not constrained to either on or off, but can take multiple values in the range [0,1]. ‘0’ represent that the gene is completely off and ‘1’ represents the gene being on. All the levels between ‘0’ and ‘1’ represent intermediate gene expression values. The intermediate expression values may also be due to post translational modifications (like in EGFR receptor). We use 1-hot encoding and BDDs for modeling the multiple valued networks.

We show the application of our method on T-Cell [Mendoza, 2006] and *Arabidopsis thaliana* [Espinosa-Soto et al., 2004] networks. On these networks, GenYsis identifies identical steady states and gives similar perturbation experiments results as were reported in the literature.

Abhishek Garg and Giovanni DeMicheli are with the Laboratory of System Integrated, Faculty of Information and Communication Sciences, Ecole Polytechnique F d rale de Lausanne, Station 14, 1015 Lausanne, Switzerland abhishek.garg@epfl.ch, giovanni.demicheli@epfl.ch

Luis Mendoza is with Instituto de Investigaciones Biom dicas, UNAM, Mexico.lmendoza@biomedicas.unam.mx

Ioannis Xenarios is with Merck Serono, Geneva, Switzerland.ioannis.xenarios@merckserono.net

The paper is organised as follows. We first report the results in section II. In section III, we give the technical details of the method. Finally, in section IV, we introduce the method for automatic extraction of rules from multiple valued networks.

II. RESULTS

We applied genYsis on two well researched gene regulatory networks, namely T Helper and *Arabidopsis thaliana* cell network.

A. T Helper Cell Network

The vertebrate immune system is made of diverse cell populations; some of them are antigen presenting cells, natural killer cells, B and T lymphocytes. There is a sub-population of T lymphocytes, the T-helper, or Th, cells that have received much attention from the modeling point of view. Th cells can be divided into precursor Th0 cells and effector Th1 and Th2 cells, depending on the pattern of secreted molecules. Th1 and Th2 cell types play a central role in cellular immunity and humoral responses, respectively. Moreover, immune responses biased towards the Th1 phenotype result in autoimmune diseases, while enhanced Th2 responses originate allergic reactions. Various mathematical models have been proposed for the differentiation, activation and proliferation of Th-lymphocytes. Early models aimed to describe the interactions between the various immunological cell populations at a macroscopic level. Other model analyses were aiming at understanding the mechanism of the generation of antibody and T-cell receptors diversity, as well as the dynamical properties of the large networks defined by the interactions between secreted cytokines or between immunoglobulins. Recently, there have been some publications on the regulatory network that controls the differentiation of Th cells [Mendoza and Xenarios, 2006, Mendoza, 2006]. The regulatory network presented constitutes the most extensive attempt to model the regulatory network controlling the differentiation of Th lymphocytes to date, and it has been implemented both as a discrete and a continuous dynamical system. The topology of the network was derived from published experimental data. The parameters were inferred from published data for the discrete model, and a set of default values were used for the continuous model. Despite the very different approaches of the discrete and continuous versions, they reach the same qualitative results.

We modeled the multi-valued Th Network (Figure 1) introduced in [Mendoza, 2006] using genYsis. This network has four genes at three levels of activation : low, medium and high. All other genes have only two levels : low and

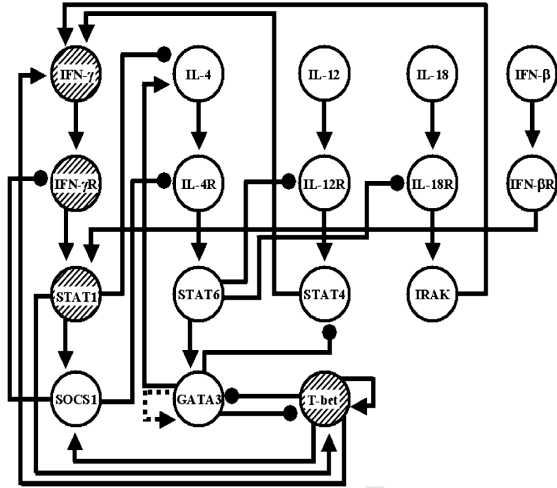


Fig. 1. Th Cell Gene Regulatory Network. Shaded Nodes have three levels of gene expression. Mendoza [2006]

high. The rules relating the expression of different genes are in the same format as Table I. We do not list the rules here due to space constraints, but they can be found in [Mendoza, 2006].

When this network was run through the GenYsis, it found four wild type (without any mutation) steady states shown in Table II. These steady states match the ones reported in [Mendoza, 2006]. We also tried two mutations, $IFN-\gamma^-$ (i.e. $IFN-\gamma$ knocked out) and $IFN-\gamma R^-$. With these mutations, similar steady states as reported in [Mendoza, 2006] were discovered. They are listed in Table II. Time taken by GenYsis to perform each of these experiment was less than 5 seconds on a 1.8 Ghz Intel Processor machine running on Linux.

The analysis performed on the Th model permitted the identification of all the stable states observed in the biological system, specifically under wild-type conditions. It is straightforward to modify the model so as to describe the situation where there are null-mutations. This capacity to simulate mutants helps both to validate the model and to help interpret some apparently contradictory phenotypes. Specifically, the model correctly helps in describing the difference in the activation patterns found between null-mutants in $IFN-\gamma$ and $IFN-\gamma R$; a difference that originally was not expected by experimental biologists because one element belongs to the same circuit and one molecule is directly downstream of the other.

Published quantitative data on the expression of the molecules represented in the Th regulatory network is currently lacking. Hence, it would be very instructive to model the Th network with different levels of granularity with respect to the levels of activation of its nodes, so as to know which stable steady states are obtained regardless of the underlying modeling approach. Moreover, it is important to

compare the result of mutants in the model, so as to validate it against experimental data.

GenYsis provides a way to biologists to perform these experiments in silico and test the validity of the network with respect to the experimental data.

B. *Arabidopsis thaliana* Network

Flowers of *Arabidopsis thaliana* are formed by four concentric whorls of organs made of four sepals (the outermost whorl), four petals, six stamens and two fused carpels (the innermost whorl). This organization of the flower can be disrupted by mutations in a series of genes. The analysis of such mutations led to the proposition of a combinatorial schema, called the “ABC model”, which has been used extensively to describe the morphology of *Arabidopsis* flowers, both in a wild type and mutant backgrounds [Coen and Meyerowitz, 1991]. The ABC model postulates the existence of three different abstract activities, namely A, B, and C, each of which is present in sets of two adjacent whorls. Whichever the nature of the molecular mechanism, the particular combination of these activities determines the identity of the organs that will develop in a particular whorl. Specifically, the sole presence of the A activity will determine the differentiation of the underlying tissue into sepals. The combination of A and B activities, however, determine the differentiation of petals. A combination of B and C leads to a production of stamens. And finally, the C activity by itself determines the development of carpels. Additionally, the ABC model postulates a mutual inhibition between activities A and C, such that when function A is absent function C substitutes it and vice versa.

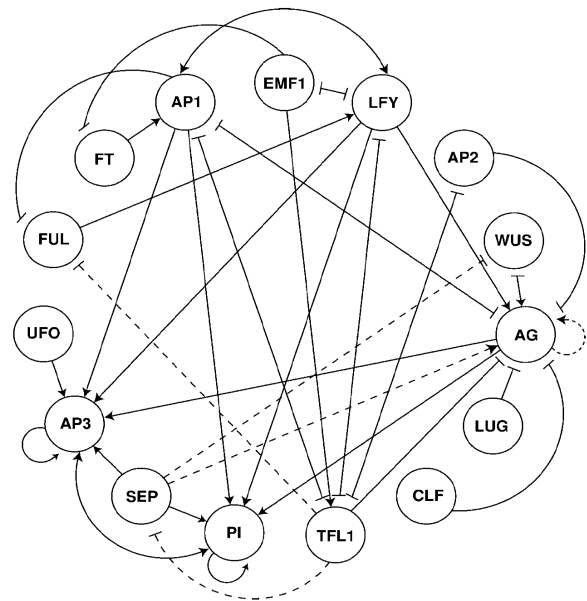


Fig. 2. *Arabidopsis thaliana* Gene Regulatory Network. Taken from Espinosa-Soto et al. [2004]

Many genes involved in *Arabidopsis thaliana* flowering and flower morphogenesis have been identified. This has permitted the cloning and analysis of expression patterns of the genes. Moreover, there is already a wealth of information related to the phenotype associated with alteration in gene expression, being null mutations or over-expression. All the known experimental information lead to the proposition of the regulatory network that controls the flower morphogenesis in *Arabidopsis* [Mendoza and Alvarez-Buylla, 1998, Mendoza et al., 1999]. This first model was later enlarged by the incorporation of new genes and interactions [Espinosa-Soto et al., 2004]. The initial versions of the flowering model used binary variables to represent the activation of genes. The more recent version of the model [Espinosa-Soto et al., 2004] used both two- and three-valued variables to represent the levels of gene activation. In this case, the model showed patterns of expression that could be compared directly with those observed not only in the mature flower, but also in the inflorescence meristems and floral organ primordia. Finally, in all versions of the *Arabidopsis* model it was possible to simulate the effect of null mutations, obtaining results that were qualitatively correct with the published experimental data.

In this paper we show the application of genYsis on the *Arabidopsis* network published in [Espinosa-Soto et al., 2004] with both two and three levels of expression of gene activation. The network used is as shown in Figure 2. The rules encoding the interactions of genes are in a similar format as for the Th Cell Network in the previous section. We do not show the rules in this paper due to space constraints. Interested readers can find the rules in [Espinosa-Soto et al., 2004].

GenYsis found 10 wild type steady states on the network 2. These steady states are listed in Table III and match the ones reported in [Espinosa-Soto et al., 2004]. In addition to the wild type experiment, we tried two different mutations: $AP2^-$ and $AP3^-$. The steady states reported are listed in Table III. These steady states match the cell states reported in experimental data published [Liu and Meyerowitz, 1995, DeyHoles and Sieburth, 2000] and [Bowman et al., 1989]. The same results were also reported in [Espinosa-Soto et al., 2004]. Time taken for each of these experiments was less than 5 seconds.

From these applications it is clear that genYsis is very efficient in modelling the dynamics of gene regulatory networks and can be used by biologists, to some extent, to understand the development of complex organisms.

III. BINARY DECISION DIAGRAMS

A. Introduction

A Binary Decision Diagram (BDD) [Bryant, 1986] is a directed acyclic graph consisting of a root node, intermediate decision nodes and two terminal nodes, namely **0**-terminal and **1**-terminal. BDDs can be used for representing Boolean functions. Each variable of the function is represented as a decision node of the graph. Each decision node has two outgoing edges to represent evaluation of variable to **1** and

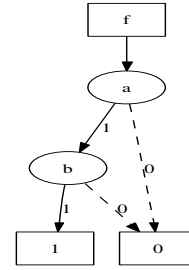


Fig. 3. BDD for the function $f = (a \wedge b)$.

0. All paths from root node to **1**-terminal gives the variable evaluations for which the function is true. A simple BDD that represents the Boolean function $f = a \text{ AND } b$ is shown in Figure 3.

Here, we use Reduced Ordered BDDs (ROBDDs), which are the compact reduced form of BDDs. For the sake of brevity whenever we say BDD in this paper, we refer to ROBDDs.

B. BDDs for multiple valued functions

BDDs as defined in section III-A can be used for Boolean function representation. To extend the same for multiple valued functions one can introduce the concept of 1-hot encoding. Using 1-hot encoding, if there are n logic levels for a variable v_i , then we can represent v_i using a binary vector x^i of length n . A bit j of vector x^i is 1 (i.e. $x_j^i = 1$) if the variable takes the logic level j . Otherwise $x_j^i = 0$. So, if a variable has three logic levels: low, medium and high, then these levels can be represented as 001, 010 and 100 respectively. This encoding scheme is called 1-hot encoding.

To use this definition of 1-hot encoding to represent the function $f = (a \wedge b)$ over three logic levels low, medium and high, we need some additional rules to describe the semantics of \wedge , \vee and $=$ for multiple valued variables. Let us assume these rules are as given in Table I. Then the Boolean function corresponding to $f = \text{medium}$ is given by:

$$f = (a = m \wedge b = m) \vee (a = l \wedge b = h) \vee (a = h \wedge b = l) \quad (1)$$

Similar functions exist for $f = \text{low}$ and $f = \text{high}$. These functions are now Boolean and can be represented using BDDs.

TABLE I
RULES FOR FUNCTION $f = (a \wedge b)$.

$f = \text{low}$	$f = \text{medium}$	$f = \text{high}$
$a = l, b = l$	$a = m, b = m$	$a = m, b = h$
$a = l, b = m$	$a = l, b = h$	$a = h, b = m$
$a = m, b = l$	$a = h, b = l$	$a = h, b = h$

C. BDDs for Gene Regulatory Networks

In this section, we show how gene regulatory networks given as a table of rules can be mapped into BDDs using

1-hot encoding. For each gene i , we define a vector of length n_i , where n_i is the number of logic levels for that particular gene. This way we can have a variable number of logic levels for different genes. The state of the gene i , at time t is given by the Boolean vector $x_i^{n_i}(t)$. We represent each rule in the table as $f_i^{j_p}$, which represents p^{th} rule for j^{th} logic level of i^{th} gene. To represent the evolution in time, we define the state of a gene at time $t + 1$ using the following functions:

$$x_i^j(t+1) = \left(\bigvee_{p=1}^{m_j} f_i^{j_p}(t) \right) \quad (2)$$

$$f_i^{j_p}(t) = \left(\bigwedge_{k=1}^{m_{j_p}} x_k^{j_p}(t) \right) \quad (3)$$

m_j are the number of rules for j^{th} logic level

m_{j_p} are the number of genes in p^{th} rule of j^{th} level

$$x \in \{0,1\}$$

\wedge and \vee represent logical AND and OR

Equation 3 represents a single rule in the rule table and Equation 2 represents the state of the bit corresponding to logic level j of gene i at time $t + 1$ as the logic OR of all the rules corresponding to level j of gene i .

A snapshot of the activity of all the genes in the network at a time t is called the state of the network. The state of the network is represented by a Boolean vector V_t of size $N \times p$, where N is the number of genes in the network and p is the sum of number of logic levels of all the genes. Each gene has n_i continuous bits in vector V_t representing the corresponding logic levels. We call these continuous bits *fragments*. Only one of these n_i bits will be 1 since a gene can only be at one logic (or expression) level at any given instance. Another similar Boolean vector V_{t+1} , is used to represent status of the genes at next step.

Asynchronous model of the transition between two time steps is used. In this asynchronous model, we assume that only one gene can change its state from one step to another but there is no priority on which gene changes its state. All the genes are equally likely to change their expression state. This way one state can have upto N successor states where each successor state (V_{t+1}) will differ from the previous state (V_t) in only one fragment. Another way to look at this asynchronous model is to say that time points are close enough, so that only one gene can make a transition between two time points. This model has been used previously studied in [Garg et al., 2007, Thomas, 1991].

Let $T_i(V_t, V_{t+1})$ be the BDD representing transition of gene i from V_t to V_{t+1} and $T(V_t, V_{t+1})$ be the BDD representing the transition from state of the network at time t to state at time $t + 1$. The relation between $T_i(V_t, V_{t+1})$ and $T(V_t, V_{t+1})$ is given by Equation 4. Equation 4 says that all genes make asynchronous transitions and state of the network at time t can have multiple successor states.

$$T(V_t, V_{t+1}) = T_0(V_t, V_{t+1}) \vee T_1(V_t, V_{t+1}) \vee \dots \vee T_N(V_t, V_{t+1}) \quad (4)$$

To impose the constraint that two consecutive states differ in atmost one gene evaluation, we define $T_i(V_t, V_{t+1})$ as in Equation 5, which states that for gene i , its evaluation at the next time step $v_i^{j'} (\in V_{t+1})$ and function $g_i^j(V_t) (= x_i^j(t+1))$ has the same value for all logic levels j , and all the other genes remain at their activation level from the previous time step.

$$T_i(V_t, V_{t+1}) = \left\{ \bigwedge_{j=1}^{n_i} \left(v_i^{j'} \leftrightarrow g_i^j(V_t) \right) \right\} \wedge \bigwedge_{k \neq i} \left\{ \bigwedge_{j=1}^{n_k} \left(v_k^{j'} \leftrightarrow v_k^j \right) \right\} \quad (5)$$

In Equation 5, \leftrightarrow is the logic equivalence operation and evaluates to 1 if the Boolean variable on left and right of \leftrightarrow have the same value. Equation 5 says that gene i takes the value decided by the function g_i at next time step and all the other genes ($k \neq i$) remain at the same level in next time step. Equation 5 represents all the transitions (Figure 4) that may exist in the state transition diagram of the gene regulatory networks. Variables $v_i^j (\in V_t)$ represent the state at the tail of the edge and $v_i^{j'} (\in V_{t+1})$ represents the head of the edge. If we know the initial state (i.e. vector V_t), then vector V_{t+1} can be determined by the following steps :

- 1) Construct a BDD 'X' which represents the vector V_t .
- 2) Take logical 'AND' of BDD 'X' with the BDD $T(V_t, V_{t+1})$.
- 3) Existentially quantify out (or remove) variables in V_t from the resulting BDD.
- 4) Swap variables $v_i^{j'} \in V_{t+1}$ with $v_i^j \in V_t$ in the BDD got from the last step.

Resulting BDD from step 3 above gives the vector V_{t+1} . To compute states reachable from this new state, change the variable names $v_i^{j'} (\in V_{t+1})$ with $v_i^j (\in V_t)$ as in step 4 and repeat steps 1-3. If these, steps are repeated many times, then at one stage V_t will be same as V_{t+1} . This state V_t is then the steady state of the system, as once reached it can not be escaped.

Steps 1-4, is the standard procedure used in the field of Model Checking and Verification for doing reachability analysis to compute reachable states in a state transition diagram. In our previous paper [Garg et al., 2007], we had explained the algorithm based on these steps to compute genuine steady states in the state transition diagram for the Boolean Network. The beauty of the formulation presented in this paper is that the same set of algorithms can be used with modified BDD structure to compute the steady states in multi-valued networks. To make this paper self complete, we explain in brief the algorithms proposed in [Garg et al., 2007] in next section, without repeating the technical explanation which can be obtained in [Garg et al., 2007].

D. Steady State Computation

All the steady states in the state transition diagram can be computed efficiently by Algorithm 1, which is based on the following set of definitions and theorems.

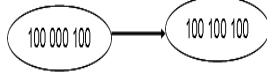


Fig. 4. A sample transition in state transition diagram.

Definiton 1: Forward image, $I_T^f(S(V_t))$ is the set of immediate successors of the states in $S(V_t)$ on the state transition graph.

Definiton 2: Backward image, $I_T^b(S(V_t))$ is the set of immediate predecessors of the states in $S(V_t)$ on the state transition graph.

Definiton 3: Forward reachable states $FR(S_0)$ from the states S_0 are all the states that can be reached from S_0 by iteratively computing forward image in the transition relation $T(V_t, V_{t+1})$ until no new states are reachable.

Definiton 4: Backward reachable states, $BR(S_0)$, are all the states in $T(V_t, V_{t+1})$ whose forward reachable states contain S_0 .

Definiton 5: Steady State is the set of states $SS(V_i)$ having the following two properties :

- 1) Forward image $I_T^f(SS(V_t))$ is same as $SS(V_t)$.
- 2) For all the states in $SS(V_i)$, if that state is reached once, then the probability of revisiting that state is one.

[Hachtel et al., 1996]

Theorem 1: A state $i \in S$ is a steady state if and only if $FR(i) \subseteq BR(i)$. State i is transient otherwise. [Xie and Beerel, 1998].

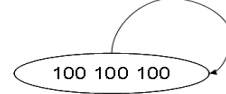
Theorem 2: If state $i \in S$ is transient, then states in $BR(i)$ are all transient. If state i is steady, then all the states in $FR(i)$ are steady states. In the latter case set $\{BR(i) - FR(i)\}$ are all transient. [Xie and Beerel, 1998].

Based on the point 1 and 2 of definition 5 of steady state, three kinds of steady states as shown in Figure 5, can be defined. Based on the definitions 1-5 and Theorems 1 and 2, Algorithms 1 can list all the steady states. More details about the Algorithm 1 can be found in [Garg et al., 2007], where they are explained in detail.

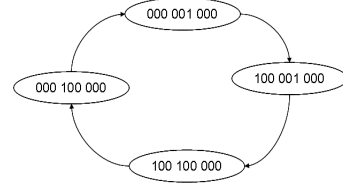
IV. AUTOMATIC EXTRACTION OF RULES

A lot of work has been reported in literature, where the multiple valued rules like the ones given in section II-A and II-B can be found. However, often extracting these rules is not very straightforward. In such a situation, one has the knowledge about the connectivity of the network but no knowledge about the rules. We have extended genYsis to automatically extract rules from the gene regulatory networks, given the mathematical function that relates the expression of output gene as a function of expression of its input genes.

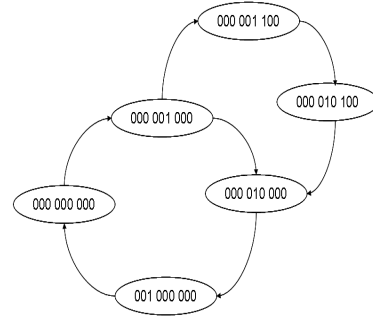
To give an example, we used the sigmoid function (Equation 6) introduced in Mendoza [2006], to represent the relationship between the expression of input genes and the output gene.



(a) Self Loop



(b) Simple Loop



(c) Complex Loop

Fig. 5. Different types of steady states.

$$x_j(t+1) = \frac{-e^{0.5h} + e^{-h(\omega_j-0.5)}}{(1 - e^{0.5h})(1 + e^{-h(\omega_j-0.5)})} \quad (6)$$

$$\omega_j = \begin{cases} \left(1 - k_j^i \left(\frac{1+\sum \beta_n}{\sum \beta_n}\right) \left(\frac{\sum \beta_n x_n^i}{1+\sum \beta_n x_n^i}\right)\right) \times \\ \quad k_j^a \left(\frac{1+\sum \alpha_n}{\sum \alpha_n}\right) \left(\frac{\sum \alpha_n x_n^a}{1+\sum \alpha_n x_n^a}\right) \\ \left(1 - \left(\frac{1+\sum \beta_n}{\sum \beta_n}\right) \left(\frac{\sum \beta_n x_n^i}{1+\sum \beta_n x_n^i}\right)\right) \\ \left(\frac{1+\sum \alpha_n}{\sum \alpha_n}\right) \left(\frac{\sum \alpha_n x_n^a}{1+\sum \alpha_n x_n^a}\right) \end{cases} \quad (7)$$

$$0 \leq \alpha, \beta, k, x \leq 1$$

Equation 6 has the value of expression of a gene between '0' and '1'. The first expression for ω is used if both activators and inhibitors are acting on a gene, second and third expressions are used if only inhibitors and only activators respectively are acting on a gene. In the former case, the gene may have some weight associated to inhibiting effect and activating effect, given by k_j^i and k_j^a respectively. α and β are the weights on each transition in the signaling network. If all the gene expressions x_j are discretized to n_j discrete

Algorithm 1: Algorithm to compute all Steady States

```
1 all_Steady_States(T)
2 begin
3   T' ← T
4   while T' ≠ ∅ do
5     s ← initial_state(T')
6     FR(s) ← forward_set(s, T')
7     BR(s) ← backward_set(s, T')
8     if FR(s) ∧ BR(s) = ∅ then
9       | report s ∨ FR(s) as a steady state
10    else
11      | report s ∨ BR(s) as all transient states
12    | T' ← T' ∧ s ∨ BR(s)
13 end
14 initial_state(T)
15 begin
16   P = random_path_to_1_node(T)
17   s(Vt) = ∃v ∈ Vt P
18   RS(0) ← ∅, FS(0) ← {s}
19   k ← 0
20   while FS(k) ≠ ∅ do
21     | FS(k+1) = If(FS(k))(Vt+1 ← Vt) ∧ RS(k)
22     | RS(k+1) = RS(k) ∨ FS(k+1)
23     | k ← k + 1
24   s ← random_path_to_1_node(FS(k-1))
25   return s
26 end
```

levels, then the Equations 6 and 7 can be discretized at these discrete levels to give rules in the format used in section II.

The benefit of computing gene expression evolution using the Equations 6 and 7 is that the expression is continuous between ‘0’ and ‘1’ level and the rules can be automatically generated even if there is no prior knowledge about the multi-valued interactions is available. The choice of sigmoid is due to the fact that it is the closest non-linear continuous function to the Boolean function.

In genYsis, the number of discretization of each gene can be chosen to be different. We applied genYsis on the T-Cell Network reported in section II-A, by automatically generating rules using the sigmoid function above. Three wild type steady states were reported by genYsis. These three states are contained in the four steady states reported in [Mendoza, 2006] and correspond to the first three states of wild type in Table II.

V. CONCLUSION

The methodology presented here was tested on two biologically significant networks, the T helper cell network and the *Arabidopsis thaliana* network. Results reported in the paper were corroborated by the same reported in the literature. Enhanced model representation maintains the efficiency of the core algorithm of GenYsis. In future, we will extend our approach to bring discrete modelling closer to experimentally generated data.

REFERENCES

J.L. Bowman, D.R. Smyth, and E.M. Meyerowitz. Genes directing flower development in *arabidopsis*. *Plant Cell*, (1):37–52, 1989.

- Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.
- E. Coen and E.M. Meyerowitz. The war of the whorls: genetic interactions controlling flower development. *Nature*, (353):31–37, 1991.
- M.K. DeyHoles and L.E. Sieburth. Seperable whorl-specific expression and negetive regulation by enhancer elements within the *agamous* second intron. *Plant Cell*, (12):1799–1810, 2000.
- C. Espinosa-Soto, P. Padilla-Longoria, and E.R. Alvarez-Buylla. A gene regulatory network model for cell fate determination during arabidopsis thalina flower development that is robust and recovers experimental gene expression profiles. *Plant Cell*, (16):2923–2939, 2004.
- A. Garg, I. Xenarios, L. Mendoza, and G.. DeMicheli. Efficient methods for dynamic analysis of genetic networks and in silico gene perturbation experiments. *Lecture Notes in Bioinformatics*, (4453):62–76, 2007.
- G. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Markovian analysis of large finite state machines. *IEEE Transactions on Computer-Aided Design*, 15:1479–1493, 1996.
- Z. Liu and E.M. Meyerowitz. *LEUNIG* regulates *agamous* expression in *arabidopsis* flowers. *Development*, (121):975–991, 1995.
- L. Mendoza. A network model for the control of the differentiation process in th cells. *BioSystems*, (84):101–114, 2006.
- L. Mendoza and E.R. Alvarez-Buylla. Dynamics of the genetic regulatory network for arabidopsis thalina flower morphogenesis. *J. theor. Biol.*, (193):307–319, 1998.
- L. Mendoza and I. Xenarios. A method for the generation of standardized qualitative dynamical systems of regulatory networks. *Theoretical Biology and Medical Modelling*, 3, 2006.
- L. Mendoza, D. Thieffry, and E.R. Alvarez-Buylla. Genetic control of flower morphogenesis in arabidopsis thaliana: a logical analysis. *Bioinformatics*, 15:593–606., 1999.
- L. Sánchez and D. Thieffry. Segmenting the fly embryo: a logical analysis of the pair-rule cross-regulatory module. *J. Theor. Biol.*, 224:517–537, 2003.
- R. Thomas. Regulatory networks seen as asynchronous automata: a logical description. *J. Theor. Biol.*, 153:1–23, 1991.
- Aiguo Xie and Peter A. Beerel. Efficient state classification of finite state markov chains. In *Design Automation Conference*, pages 605–610, 1998.

TABLE II
STEADY STATES OBSERVED IN WILD TYPE AND MUTATED TH GENE NETWORK

Mutation	Gene Name												
	IFN- γ	IRAK	IL4	STAT1	GATA3	IFN- γ R	SOCS1	IL4R	IL12R	STAT6	IL18R	STAT4	T-bet
Wild Type	h	l	l	m	l	m	h	l	l	l	l	l	h
	l	l	h	l	h	l	l	h	l	h	l	l	l
	l	l	l	l	l	l	l	l	l	l	l	l	l
	m	l	l	m	l	m	h	l	l	l	l	l	m
IFN- γ ⁻	l	l	l	l	l	l	h	l	l	l	l	l	h
	l	l	l	l	l	l	l	l	l	l	l	l	l
	l	l	l	l	l	l	h	l	l	l	l	l	m
	l	l	h	l	h	l	l	h	l	h	l	l	l
IFN- γ R ⁻	h	l	l	l	l	l	h	l	l	l	l	l	h
	m	l	l	l	l	l	h	l	l	l	l	l	m
	l	l	l	l	l	l	l	l	l	l	l	l	l
	l	l	h	l	h	l	l	h	l	h	l	l	l

TABLE III
STEADY STATES OBSERVED IN WILD TYPE AND MUTATED ARABIDOPSIS GENE NETWORK

Mutation	Gene Name															Cell Type
	FT	EMF1	LFY	TFL1	AP1	FUL	AP2	SEP	AG	PI	AP3	WUS	LUG	CLF	UFO	
Wild Type	0	1	0	2	0	0	0	0	0	0	0	1	1	1	1	<i>Infl3</i>
	1	0	2	0	0	2	1	1	2	2	2	0	1	1	1	<i>St1</i>
	1	0	2	0	0	2	1	1	2	1	0	0	1	1	0	<i>Car</i>
	1	0	2	0	2	0	1	1	0	2	2	0	1	1	1	<i>Pe1</i>
	0	1	0	2	0	0	0	0	0	0	0	1	1	1	0	<i>Infl4</i>
	0	1	0	2	0	0	0	0	0	0	0	0	1	1	1	<i>Infl2</i>
	1	0	2	0	2	0	1	1	0	0	0	0	1	1	0	<i>Sep</i>
	1	0	2	0	0	2	1	1	2	2	2	0	1	1	0	<i>Pe2</i>
	1	0	2	0	2	0	1	1	0	2	2	0	1	1	0	<i>St2</i>
	0	1	0	2	0	0	0	0	0	0	0	0	1	1	0	<i>Infl1</i>
AP3 ⁻	0	1	0	2	0	0	0	0	0	0	0	1	1	1	0	<i>Infl4</i>
	1	0	2	0	2	0	1	1	0	0	0	0	1	1	0	<i>Sep</i>
	1	0	2	0	0	2	1	1	2	1	0	0	1	1	0	<i>Car</i>
	1	0	2	0	2	0	1	1	0	0	0	0	1	1	1	<i>Sep</i>
	1	0	2	0	0	2	1	1	2	1	0	0	1	1	1	<i>Car</i>
	0	1	0	2	0	0	0	0	0	0	0	1	1	1	1	<i>Infl3</i>
	0	1	0	2	0	0	0	0	0	0	0	0	1	1	0	<i>Infl1</i>
	0	1	0	2	0	0	0	0	0	0	0	0	1	1	1	<i>Infl2</i>
AP2 ⁻	0	1	0	2	0	0	0	0	0	0	0	0	1	1	1	<i>Infl2</i>
	1	0	2	0	0	2	0	1	2	2	2	0	1	1	1	<i>St1</i>
	0	1	0	2	0	0	0	0	0	0	0	1	1	1	1	<i>Infl3</i>
	1	0	2	0	0	2	0	1	2	1	0	0	1	1	0	<i>Car</i>
	0	1	0	2	0	0	0	0	0	0	0	1	1	1	0	<i>Infl4</i>
	1	0	2	0	0	2	0	1	2	2	2	0	1	1	0	<i>St2</i>
	0	1	0	2	0	0	0	0	0	0	0	0	1	1	0	<i>Infl1</i>