Infer Gene Regulatory Networks from Time Series Data with Probabilistic Model Checking

Michele Ceccarelli[†‡], Luigi Cerulo[†§], Giuseppe De Ruvo[*], Vittoria Nardone[*], Antonella Santone[*]
[*]Dep. of Engineering, University of Sannio, Benevento (Italy)
[†]Dep. of Science and Technology, University of Sannio, Benevento (Italy)
[‡]QCRI - Qatar Computing Research Institute, Doha, Qatar
[§]BioGeM s.c.a r.l., Institute of Genetic Research "Gaetano Salvatore", Ariano Irpino, AV (Italy)

***Abstract*—Gene regulatory relationships constitute a complex mechanism of interactions adopted by cells to control behaviours and functions of a living organism. The identification of such relationships from genomics data through a computational approach is a challenging task as the large number of possible solutions is typically high in contrast to the number of available independent data points. Literature approaches address the problem by reducing the search space and/or extend the amount of independent information.**

In this paper we propose a probabilistic variant of a previous proposed approach based on formal methods. The method starts with a formal specification of gene regulatory hypotheses and then determines which is the probability that such hypotheses are explained by the available time series data. Both direction and sign (inhibition/activation) of regulations can be detected whereas most of literature methods are limited just to undirected and/or unsigned relationships.

We empirically evaluated the probabilistic variant on experimental and synthetic datasets showing that the levels of accuracy are in most cases higher than those obtained with the previous method, outperforming, indeed, the current state of art.

I. Introduction

The expression of protein coding genes are controlled by particular genes named Transcription Factors that have the ability to bind the DNA sequence of gene promoters. The promotion or the blocking of RNA polymerase recruitment allows the activation or the inhibition of a target gene expression activity. The identification of such molecular interactions is crucial to understand the mechanism that allow a cell to live. The problem of reconstructing a gene regulatory network is undetermined as the number of possible solutions is typically high compared to the number of available independent data points. Many possible solutions explain the data equally well but only one of them has a biological meaning [1]. Different strategies have been proposed in literature aiming at inferring interaction hypotheses from high throughput genomic and proteomic data. Most of them identify transcriptional activities by reducing the search space and/or extending the amount of independent information [2], [3], [4], [5], [6], [7].

System biology approaches the problem with a system of ordinary differential equations (ODE) that model the interaction process among the various components. The model is deterministic and can be simulated, analyzed, and possibly solved, but requires a very detailed knowledge of the biological system and a huge computational power [8].

Computational models mimic biological phenomena and adopt heuristics to simplify the model. They can be deterministic, nondeterministic, and/or stochastic, and have the advantages of effectively representing the biological systems without precise quantitative relationships. The most representatives are Boolean networks and Bayesian models [9], [10].

Correlation methods aim at detecting gene–gene interactions with a similarity measure— *i.e.*, mutual information or statistical correlation—filtered by a properly estimated threshold. The main advantage is that they scale on very huge datasets allowing for genome–wide analysis but in most cases are not able to detect the direction nor even the sign of regulation. The main representatives are: ARACNE [11], TD–Aracne [12], PCA-CMI [13], and CLR [14].

Supervised approaches consider the reconstruction of gene regulatory networks as a learning task [15], [16]. Although supervised methods outperform generally the previous mentioned unsupervised approaches they require a training set, *i.e.*, a set of genes where the information that they interact is known in advance, to estimate a function to discriminate whether a new gene interaction exists or not. This is basically not a problem in simple model organisms but such methods are constrained to learn from a training set of only positive examples which makes the training task more challenging [17], [18].

In this paper we extend an approach, based on formal methods [19], exploiting probabilistic model checking that is able to capture the aleatory nature of the problem. In particular, we employed PRISM [20], a popular probabilistic model checker, in order to model and verify Discrete Time Markov Chains which we used to represent gene networks. More precisely, we show how to use PRISM and probabilistic model checking to infer gene regulatory networks from time course experimental data. Discretized time-series data are used to build a formal model, which take into account the evolution of gene expressions over time. The model is then used to construct the gene regulatory network establishing gene-gene interactions. This is obtained using the model checking technique [21], [22], [23], *i.e.*, checking whether the formal model satisfies the temporal properties, expressed in a probabilistic temporal logic, modeling the underlying biological knowledge. We compare the approach with the state of art obtaining promising results in terms of precision and recall.

The remainder of this paper is organized as follows. First, in the next Section we give basic definitions of probabilistic model checking and some preliminaries on biology. Section III introduces the approach. In Section IV we outline the empirical evaluation procedure, report and discuss the outcomes of the study. In Section V we summarise related work. Finally, in Section VI we conclude and provide new inputs for further research.

II. Background on Biology and Probabilistic Model Checking

The central dogma of biology explains the flow of genetic information within a biological system. Information stored in DNA molecules are transcribed into RNA and then translated into proteins, which are the working elements of living organisms. DNA information is transmitted through reproduction from parents to offspring and the resulting phenotype results in a combination of their parents phenotypes. Most of DNA information is stored in small regions named genes and spread among the genome. The control on when and how genes are transcribed and translated into protein is demanded to a complex interaction network that is able to respond to external stimuli and to different environment conditions. Unveiling the complex topology of such gene interaction networks is crucial in biology research. As an example, several genetic based disorders, such as cancer, are due to the alteration of important sub networks that are related to DNA repairing, cell apoptosis, and inflammatory response. Knowing how genes interact in such sub networks in both healthy and disease conditions is crucial to find for new drug targets and to understand the prognosis of patients.

Model checking is a technique to establish the correctness of hardware or software systems in an automated fashion. The goal of this technique is to try to predict system behaviour, or more specifically, to formally prove that all possible executions of the system conform to the requirements.

In the case of probabilistic model checking, the models are probabilistic, in the sense that they encode the probability of making a transition between states. We suppose the reader familiar with probabilistic model checker. The reader can refer to [24] for further details.

In this paper we use discrete-time Markov chains (DTMCs), which specify the probability $\pi(s, s')$ of making a transition from state s to some target state s', where the probabilities of reaching the target states from a given state must sum up to 1, *i.e.* $\sum_{s'} \pi(s, s') = 1$.

A probabilistic model checker tool automates the correctness proving process. A popular probabilistic model checker is PRISM [20], successfully adopted in many application domains such as communication and multimedia protocols, randomised distributed algorithms, security protocols, biological systems [25], [26], [27].

In PRISM the system model is defined with a probabilistic variant of Reactive Modules [28], and system behaviours are defined in terms of properties written in Probabilistic Computation Tree Logic (PCTL*) [23] — a probabilistic extension of the temporal logic CTL*.

III. Approach

The approach starts from a set of k time course experiments and a set of gene perturbation definitions. A perturbation definition specifies whether a gene of the network is forced to be up-regulated (value Up), down-regulated (value $Down$) or left free to evolve, without any excitement (value $Basal$). For each perturbation definition the values of gene expressions resulting from the application of such a perturbation are collected in a time-series aiming at describing their evolution over time. Each time course experiment is firstly discretized, *i.e.*, the expression level of each gene A is mapped to one of the following possible states: Up (high logic value), $Basal$ (basal logic value) and $Down$ (low logic value). Discretized values are obtained by applying one of the approaches in the literature. The one adopted in this paper uses an equal frequency discretization method that divides the expression levels into three bins containing approximately the same number of expression values [29].

From discretized time-series and perturbations definitions a PRISM model is built. Such a model takes into account the evolution of gene expression over time which refers to a particular perturbation as follows. For each gene of the network we generate a *module*. The evolution of the state of the gene is modelled as a sequence of *commands* indicating the sequence of states in which the gene expression value can be during a perturbation. An additional *module* is generated to implement the perturbation choice by means of a variable. Such a variable is used to connect the values belonging to a time-serie with the corresponding perturbation. Figures 2 and 1 show an example of variable declaration (`perturb_num`), while Figure 4 show an example of variable usage. We assigned the same probability $(1/n)$ to each perturbation, *i.e.* equally distributed among the number of the available perturbations.

As an example consider the discretized time series and perturbations reported respectively in Table I and Table II (in both tables it holds that U=Up,B=Basal and D=Down). For the sake of clarity, three perturbations have been considered. In the first column the perturbation of all three genes is set to basal (B).

```
[st0] perturb_num=0 -> (A_state_p'=Basal)
                     & (B_state_p'=Basal)
                     & (C_state_p'=Basal);
```

Fig. 1: States of genes in the first perturbation represented in the form of PRISM command.

This line represents the state of genes in the first perturbation, *i.e.*, the one applied on level expression, in this particular case the expression of each gene is free to evolve since none of them is excited. Similarly occurs for other perturbations of Table I. To select a perturbation the PRISM command shown in Figure 2 is adopted.

```
[] perturb_num=n -> 1/n:(perturb_num'=0)
                  + 1/n:(perturb_num'=1)
                  + ...
                  + 1/n:(perturb_num'=n-1);
```

Fig. 2: Command for perturbation choice.

The command basically assigns to each perturbation a probability equal to $1/n$, with n equal to the number of perturbations available (in our example $n = 3$, because we have considered only three perturbations, see Table I). Time-series, referred in Table II, are modelled as in Figure 4. Due to space limitation only the module of gene A is shown. The evolution of gene's expression over time is defined as a sequence of state transitions. Instead, to connect a time-series with the correspondent perturbation in the *guard* a specific variable is adopted.

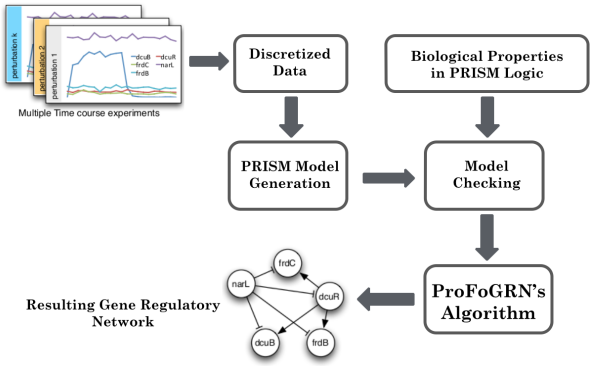

Fig. 3: Workflow of the Approach

The inference process starts from a PRISM model containing the two modules explained previously and consists basically to verify the presence of two kinds of edges: activation ($X \longrightarrow Y$) and inhibition ($X \dashv Y$). An activation relationship $X \longrightarrow Y$ means that the increment of X in time causes a direct increment of Y. Instead an inhibition relationship $X \dashv Y$ means that the increment of X in time causes a direct decrement of Y. In the following the PCTL* property for the activation relationship $A \longrightarrow C$:

$$\textbf{P=?}\ [(A\_state <= Init\ \textbf{U}\ (A\_state = Up))\ \textbf{U}\ (A\_state = Up\ \textbf{\&}\ (\textbf{F}\ (\textbf{G}\ C\_state = Up)))]$$

The formula returns the probability that whenever the gene A is Up the gene C eventually becomes Up and then it remains Up. Similarly, the PCTL* property for the inhibition relationship $A \dashv C$ is:

$$\textbf{P=?}\ [(A\_state <= Init\ \textbf{U}\ (A\_state = Up))\ \textbf{U}\ (A\_state = Up\ \textbf{\&}\ (\textbf{F}\ (\textbf{G}\ C\_state = Down)))]$$

The formula returns the probability that whenever the gene A is Up the gene C must become $Down$ and then it remains $Down$. Note that each module does not return in the its initial state. The above properties are checked on the formal probabilistic model and the probability of each formula is returned. According to a threshold, we decide to insert the edge in the final network or not. In our experiments we set a threshold to the maximum probability returned by the checked formulae.

In addition to the ones above, other three properties have been introduced:

$$\textbf{P=?}\ [A\_state = Init\ \textbf{U}\ A\_state = Up] \quad (1)$$

$$\textbf{P=?}\ [(A\_state = Init\ \textbf{U}\ (\textbf{G}\ A\_state = Up))] \quad (2)$$

$$\textbf{P=?}\ [\textbf{F}\ (\textbf{G}\ A\_state = Up)] \quad (3)$$

These three formulae, unlike the previous one, refer to a single gene instead of a pair of genes. The first returns the probability that a gene A immediately reaches the high logic value (action $A\_state = Up$) after its initial state (action $A\_state = Init$). The second formula returns the probability that a gene immediately reaches the high logic value (action $A\_state = Up$) after its initial state (action $A\_state = Init$) and then its level expression gene remains high ($\textbf{G}\ A\_state = Up$). Finally, the third formula returns the probability that a gene eventually becomes high and then it remains at a high logic value.

We define "strong gene" a gene that satisfies the above three properties with a probability greater than or equal to a certain ϵ. The threshold ϵ is equal to $1/k$, where k is the number of the perturbations where the gene is left free to evolve, *i.e.*, its value is $Basal$. The strong genes are used when analysing an activactor edge like $A \longrightarrow C$. Whenever C is a strong gene, the edge $A \longrightarrow C$ is not inserted in the inferred network, since it more likely that C is an activator gene rather than an activated gene.

Figure 3 shows the inference process.

IV. Performance Evaluation

A. Empirical Evaluation Procedure

To estimate the unknown performance of a new inference method we adopt the metrics of precision and recall, and their F-measure (Fm), defined as follows:

$$PR = \frac{TP}{TP + FP};\quad RC = \frac{TP}{TP + FN};\quad Fm = \frac{2PR\ RC}{PR + RC}$$

where TP is the number of predicted edges that are correct (True Positives), FP is the number of predicted edges that are wrong (False Positives), and FN is the number of true edges that were not predicted (False Negatives). The outcome of literature reverse engineering methods is not necessary a directed graph and may not take into account the sign (activation or inhibition) of gene regulations. Thus, to make such methods mutually comparable we compute the number of correct edges by considering the actual regulatory graph as: *Undirected*, when edges are counted as true positives

```
module GENE_A

        A_state : int;
        time_TS : [0..3] init 0;

// PERT. -> 0
        // initial state PERTURBATION 0
        [st0] A_state=Init & perturb_num=0 -> (A_state'=Down) & (time_TS'=1);

        [st1] A_state=Down & time_TS=1 & perturb_num=0 -> (A_state'=Basal) & (time_TS'=2);
        [st2] A_state=Basal & time_TS=2 & perturb_num=0 -> (A_state'=Up) & (time_TS'=3);

// PERT. -> 1
        // initial state PERTURBATION 1
        [st0] A_state=Init & perturb_num=1 -> (A_state'=Down) & (time_TS'=1);

        [st1] A_state=Down & time_TS=1 & perturb_num=1 -> (A_state'=Down) & (time_TS'=2);
        [st2] A_state=Down & time_TS=2 & perturb_num=1 -> (A_state'=Down) & (time_TS'=3);
// PERT. -> 2
//      ...

endmodule
```

Fig. 4: Module of gene A as defined in PRISM (Init=3, Down=0, Basal=1, Up=2)

Gene	Pert. 0	Pert. 1	Pert. 2
A	B	D	B
B	B	D	B
C	B	B	U

TABLE I: Example of three perturbations

Gene	Pert. 0			Pert. 1			Pert. 2		
	0	1	2	0	1	2	0	1	2
A	D	B	U	D	D	D	B	U	U
B	B	B	B	D	D	D	D	D	D
C	B	B	B	D	B	U	U	U	U

TABLE II: Fragment of discretized time course experiment of three perturbations

regardless they direction; *Directed*, when edges have direction of regulation, thus true positives are counted regarding their direction; and *Signed*, when edges have both direction and sign of regulation, *i.e.* true positives are counted taking into account both direction and sign.

We evaluate the performance of each tool on both simulated (in silico) and experimental (in vivo) data. For simulated data we adopt GeneNetWeaver, an in silico benchmark generation tool for profiling the performance of network inference methods [30]. GeneNetWeaver adopts a kinetic network model of ordinary and stochastic differential equations and is adopted in the DREAM network inference challenge, an annual reverse engineering "competition" where the goal is to predict network connectivities from gene expression datasets. For the experimental data we adopt three publicly available datasets:

- *E. coli* SOS pathway. An eight genes regulating the SOS response of DNA damage. Following the work of [12] we choose genes involved in DNA damage tolerance and repair activated through *recA* and *lexA*. We adopt the first 14 point time course profiles of an experiment made publicly available by Ronen *et al.* [30].
- S. *cerevisiae* cell cycle. An eleven genes network controlling the G1 step of cell cycle. Following the work of [12] we choose genes whose mRNA levels respond to the induction of *Cln3* and *Clb28*, two cell cycle regulators, and a 16 point time course profiles made publicly available by Spellman *et al.* [31].
- *IRMA*. An in vivo fully controlled experimental network built in the S. cerevisiae and composed of five genes [32]. The network is perturbed by culturing cells in presence of galactose or glucose. Galactose activates the GAL1-10 promoter, cloned upstream of a *Swi5* mutant in the network, and it is able to activate the transcription of all the five network genes. Two perturbations corresponding to growing cells from glucose to galactose medium and to reverse shift are made respectively of 16 and 21 time points.

With simulated datasets we evaluate the performance of each method against the *number of genes*. GeneNetWeaver allows for generating biologically relevant regulatory networks of given number of genes by selecting random connected genes from the overall known gene regulatory network of *E. coli* [30]. We generate 10 random networks of n genes where $n \in \{4, 5, 10, 15, 20, 30\}$. For each random network we generate $2n + 1$ time course experiments of 20 time points each.

We empirically evaluate the proposed approach (in the following denoted **ProFoGRN**) against FormalM [19], an approach based on formal methods introduced by some authors of this paper. FormalM starts with a formal specification of

gene regulatory hypotheses and then mathematically proves whether a time course experiment belongs or not to the formal specification, determining the presence or absence of a gene regulation. To this aim FormalM adopts the Calculus of Communicating Systems (CCS) of Milner [33], whilst **ProFoGRN** adopt the PRISM specification language.

With in-silico experiments FormalM has got the best than other tools available in literature so we compare our new approach just with the FormalM baseline omitting the performance results obtained with other literature tools. Instead, with in-vivo networks results are very different among literature tools not allowing the identification of a baseline. Thus the performance of our new approach is compared against the results published in [19] comprising: ARACNE [11], CLR [14], TD–Aracne [12], TSNIF [8], and BANJO [34].

B. Results and Discussion

Figure 5 reports results of in-silico experiments. The figure shows the average performance of two methods at different network sizes obtained with the full set of available perturbations and respective gene expression levels. It can be noticed that the performance decreases when increasing the number of genes. Both approaches preserve a quite good performance. In particular the range of FormalM is between 73% and 91% ($88\% - 100\%$ Precision, $65\% - 91\%$ Recall). Instead the range of our approach is between 76% and 91% ($87\% - 100\%$ Precision, $70\% - 87\%$ Recall). In fact, as can be seen in Figure 5 in the average the F-Measure achieved by our tool is higher than that of FormalM. Moreover, Figure 5 shows separately the Precision and the Recall of the two methods. It turns out that ProFoGRN outperforms FormalM in Recall when increasing the number of genes, even if the precision of our method is slightly lower that FormalM.

Tables III, IV, and V show the performance obtained with experimental datasets in terms of precision and recall computed on undirected, directed, and signed graphs. All the methods perform well with undirected relationships whilst for directed and signed graphs differences among methods is remarkable. On *E. coli* SOS network ProFoGRN outperforms all the other methods in undirected and directed predictions, CRL and BANJO follow respectively. In the signed prediction TSNIF as a top choice but our tool is immediately after. On *S. Cervisiae* cell cycle network FormalM yet outperforms all other methods with a high recall (100% for directed and undirected, 87% for signed) and an adequate precision. Both TD-Aracne and BANJO exhibit good performance in predicting directed connections. Instead, the sign of connections are predicted with a good accuracy only by FormalM (F-measure of 30%). Unfortunately on this network the accuracy of ProFoGRN is low in correlation with other methods but the scores in all of three types of graphs are comparable with those of the other tools. On *IRMA* network TSNIF outperforms all other methods in undirected prediction, while our tool reveals better to predict the direction and the sign of relationships. It should be underlined that ProFoGRN's score in undirected prediction is immediately after the one of TSNIF (F-Measure of 86%).

TABLE III: Results on E. coli SOS

Method	Graph	Pr	Rc	Fm
ARACNE	Undirected	0.44	0.57	0.50
CLR	Undirected	0.35	1.00	0.52
TD-Aracne	Undirected	0.22	1.00	0.36
FormalM	Undirected	0.25	1.00	0.40
TSNIF	Undirected	0.21	0.43	0.29
BANJO	Undirected	0.36	0.71	0.48
ProFoGRN	Undirected	1.00	1.00	**1.00**
TD-Aracne	Directed	0.12	0.88	0.22
FormalM	Directed	0.14	1.00	0.25
TSNIF	Directed	0.12	0.25	0.17
BANJO	Directed	0.26	0.62	0.37
ProFoGRN	Directed	1.00	1.00	**1.00**
FormalM	Signed	0.04	0.25	0.06
TSNIF	Signed	0.12	0.25	**0.17**
BANJO	Signed	0.05	0.12	0.07
ProFoGRN	Signed	0.14	0.13	0.14

TABLE IV: Results on S. Cerevisiae cell cycle

Method	Graph	Pr	Rc	Fm
ARACNE	Undirected	0.24	0.20	0.22
CLR	Undirected	0.26	0.45	0.33
TD-Aracne	Undirected	0.36	0.85	0.51
FormalM	Undirected	0.36	1.00	**0.53**
TSNIF	Undirected	0.23	0.25	0.24
BANJO	Undirected	0.32	0.35	0.33
ProFoGRN	Undirected	0.30	0.18	0.23
TD-Aracne	Directed	0.21	0.61	0.31
FormalM	Directed	0.21	1.00	**0.35**
TSNIF	Directed	0.16	0.17	0.17
BANJO	Directed	0.21	0.26	0.23
ProFoGRN	Directed	0.20	0.09	0.12
FormalM	Signed	0.18	0.87	**0.30**
TSNIF	Signed	0.08	0.09	0.08
BANJO	Signed	0.17	0.22	0.19
ProFoGRN	Signed	0.20	0.09	0.12

TABLE V: Results on IRMA

Method	Graph	Pr	Rc	Fm
ARACNE	Undirected	0.60	0.43	0.50
CLR	Undirected	0.75	0.86	0.80
TD-Aracne	Undirected	0.61	1.00	0.76
FormalM	Undirected	0.70	1.00	0.82
TSNIF	Undirected	1.00	0.86	**0.92**
BANJO	Undirected	0.67	0.86	0.75
ProFoGRN	Undirected	0.86	0.86	0.86
TD-Aracne	Directed	0.32	0.75	0.44
FormalM	Directed	0.40	1.00	0.57
TSNIF	Directed	0.71	0.62	0.67
BANJO	Directed	0.45	0.62	0.53
ProFoGRN	Directed	0.71	0.86	**0.78**
FormalM	Signed	0.25	0.62	0.36
TSNIF	Signed	0.43	0.38	0.40
BANJO	Signed	0.36	0.50	0.42
ProFoGRN	Signed	0.50	0.50	**0.50**

V. Related Work

In this paper we have introduced a gene network reverse engineering approach based on probabilistic model checking. The main distinctive feature of this methodology is the use

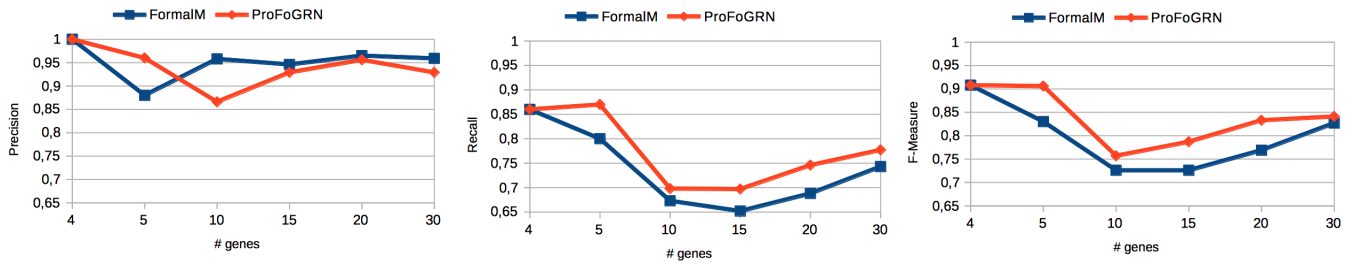

Fig. 5: FormalM vs ProFoGRN Precision, Recall, F-Measure

of PRISM (to the Authors knowledge, never used before) to infer gene regulatory networks from time series. This is a continuation and an improvement of a previous work of some of the authors [19]. In [19] the authors show how to use Calculus of Communicating Systems (CCS) of Milner [33] and model checking to infer gene regulatory networks from time course experimental data. However, the introduction of probabilities in the formal model has allowed us to outperform the results obtained by the previous CCS based approach.

In literature probabilistic formal approaches have been used mainly to analyse, verify, and simulate the behaviour of known biochemical systems [35], [36], [37], [38], [39], rather than to infer gene regulatory networks. For example, in [35], the authors describe a new modelling and analysis approach for signal transduction networks in the presence of incomplete data. Their analysis includes simulation and the models are based on high level descriptions of stochastic transition systems, *i.e.*, continuous time Markov chains (CTMCs). In [36] the stochastic process algebra PEPA [40] is used to model a pathway. In [37] the authors illustrate the applicability of probabilistic model checking to a complex biological system: the FGF (Fibroblast Growth Factor) signalling pathway. They give a detailed description of how this case study can be modelled in the probabilistic model checker PRISM. They demonstrate how several temporal properties, including some with reward-based measures, are applicable to the study of biological systems. Finally, in [38], Ciocchetta et al. (2009) present another use of PRISM, combining probabilistic model checking with stochastic simulation. Model checking is an exhaustive technique, but is subject to the state-explosion problem. Simulation, by contrast, only explores a single trajectory at a time, but is less sensitive to the search-space size. Essentially, Ciocchetta et al. (2009) try to combine the advantages of both these methods by employing a simulator to obtain approximate bounds on the amounts of each species.

VI. Conclusion and Future Work

In this work we have improved a gene network reverse engineering approach based on formal methods[19]. As occurred with the previous one, our approach is able to detect both direction and sign (inhibition/activation) of relationships and exploits multiple perturbation data. The main distinctive element of this paper is the use of PRISM, a popular probabilistic model checker already used to reach various goals. As far as we know, there are no works regarding the application of PRISM to the inference of gene regulatory networks. We have implemented a tool to assess the validity of our methodology (ProFoGRN). Probabilistic model checking allowed us to introduce probabilities and achieve encouraging and promising results, even better comparing with FormalM - the previous method introduced by some of the authors of this paper.

Experiments performed on simulated data attested the superiority of our approach with respect to the most relevant literature methods - including FormalM. Futhermore, large networks and few number of time series experiments exhibit high levels of performance even with the weakness of a rather large computational cost. Our results are better or comparable with the ones of existing literature methods but we need more experiments to make more general conclusion.

In this work we enclosed in the model just two fundamental biological rules, modelling activator and inhibitor relationships. We plan to add more biological knowledge thanks to the power of PCTL* and would investigate for including new rules, such as those regarding typical regulatory patterns which recur throughout networks (*i.e.* feed-forward loops, single-input modules, dense overlapping regulons). We even want to design and develop user-friendly tools to add new properties without knowing the logic and to help both experts and non-experts to set probabilities [41]. Integrating our tool into existing bioinformatics frameworks such as Cytoscape[42], will eventually help the community to apply our approach to more real world cases.

Acknowledgements

This work was supported by a research project funded by MiUR (Ministero dell'Università e della Ricerca) under grant FIRB2012-RBFR12QW4I.

References

[1] R. D. Smet and K. Marchal, "Advantages and limitations of current network inference methods," *Nature Reviews Microbiology*, vol. 8, no. 10, p. 717, 2010.

[2] T. S. Gardner and J. J. Faith, "Reverse-engineering transcription control networks," *Physics of Life Reviews*, vol. 2, no. 1, pp. 65 – 88, 2005.

[3] M. Bansal, V. Belcastro, A. Ambesi-Impiombato, and D. di Bernardo, "How to infer gene networks from expression profiles," *Molecular Systems Biology*, no. 3, February 2007.

[4] M. Bansal and A. Califano, "Genome-wide dissection of posttranscriptional and posttranslational interactions." *Methods Mol Biol*, vol. 786, 2012.

[5] H. Hache, H. Lehrach, and R. Herwig, "Reverse engineering of gene regulatory networks: a comparative study," *EURASIP J. Bioinformatics Syst. Biol.*, vol. 2009, pp. 1–12, 2009.

[6] J.-P. Vert, *Reconstruction of Biological Networks by Supervised Machine Learning Approaches*. Wiley, 2010, pp. 163–188.

[7] M. Grzegorczyk, D. Husmeier, and A. V. Werhli, "Reverse engineering gene regulatory networks with various machine learning methods," *Analysis of Microarray Data*, 2008.

[8] A. Polynikis, S. J. Hogan, and M. di Bernardo, "Comparing different ODE modelling approaches for gene regulatory networks." *Journal of theoretical biology*, August 2009.

[9] S. Liang, S. Fuhrman, and R. Somogyi, "Reveal, a general reverse engineering algorithm for inference of genetic network architectures," *Pac Symp Biocomput*, pp. 18–29, 1998.

[10] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis, "Advances to bayesian network inference for generating causal networks from observational biological data," *Bioinformatics*, vol. 20, no. 18, pp. 3594–3603, Dec. 2004.

[11] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano, "ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context," *BMC Bioinformatics*, vol. 7 Suppl 1, p. S7, 2006.

[12] P. Zoppoli, S. Morganella, and M. Ceccarelli, "Timedelay-aracne: Reverse engineering of gene networks from time-course data by an information theoretic approach," *BMC-Bioinformatics*, vol. 11, p. 154, 2010.

[13] X. Zhang, X.-M. Zhao, K. He, L. Lu, Y. Cao, J. Liu, J.-K. Hao, Z.-P. Liu, and L. Chen, "Inferring gene regulatory networks from gene expression data by path consistency algorithm based on conditional mutual information," *Bioinformatics*, vol. 28, no. 1, pp. 98–104, 2012.

[14] Faith et al., "Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles," *PLoS Biol*, vol. 5, no. 1, p. e8, 01 2007.

[15] F. Mordelet and J.-P. Vert, "SIRENE: supervised inference of regulatory networks," *Bioinformatics*, vol. 24, no. 16, pp. 76–82, 2008.

[16] J. R. Bock and D. A. Gough, "Predicting protein-protein interactions from primary structure," *Bioinformatics*, vol. 17, no. 5, pp. 455–460, 2001.

[17] L. Cerulo, C. Elkan, and M. Ceccarelli, "Learning gene regulatory networks from only positive and unlabeled data," *BMC Bioinformatics*, vol. 11, no. 1, p. 228, 2010.

[18] L. Cerulo, V. Paduano, P. Zoppoli, and M. Ceccarelli, "A negative selection heuristic to predict new transcriptional targets," *BMC Bioinformatics*, vol. 14, no. Suppl 1, p. S3, 2013.

[19] M. Ceccarelli, L. Cerulo, and A. Santone, "De novo reconstruction of gene regulatory networks from time series data, an approach based on formal methods," *Methods*, vol. 69, no. 3, pp. 298 – 305, 2014.

[20] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.

[21] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*. MIT Press, 2001.

[22] A. Fantechi, S. Gnesi, and G. Ristori, "Model checking for action-based logics," *Formal Meth. in Sys. Des.*, vol. 4, no. 2, pp. 187–203, 1994.

[23] A. Bianco and L. De Alfaro, "Model checking of probabilistic and nondeterministic systems," in *Foundations of Software Technology and Theoretical Computer Science*. Springer, 1995, pp. 499–513.

[24] E. Ábrahám, B. Becker, C. Dehnert, N. Jansen, J.-P. Katoen, and R. Wimmer, "Counterexample generation for discrete-time markov models: An introductory survey," in *Formal Methods for Executable Software Models*. Springer, 2014, pp. 65–121.

[25] F. Dannenberg, M. Kwiatkowska, C. Thachuk, and A. Turberfield, "Dna walker circuits: Computational potential, design, and verification," *Natural Computing*, 2014, to appear.

[26] S. Konur and M. Fisher, "Formal analysis of a vanet congestion control protocol through probabilistic verification," pp. 1–5, 2011.

[27] T. Deshpande, P. Katsaros, S. Basagiannis, and S. Smolka, "Formal analysis of the DNS bandwidth amplification attack and its countermeasures using probabilistic model checking," pp. 360–367, 2011.

[28] R. Alur and T. A. Henzinger, "Reactive modules," *Formal Methods in System Design*, vol. 15, no. 1, pp. 7–48, 1999.

[29] Y. Li, L. Liu, X. Bai, H. Cai, W. Ji, D. Guo, and Y. Zhu, "Comparative study of discretization methods of microarray data for inferring transcriptional regulatory networks," *BMC Bioinformatics*, vol. 11, no. 1, pp. 520+, 2010.

[30] T. Schaffter, D. Marbach, and D. Floreano, "GeneNetWeaver: In silico benchmark generation and performance profiling of network inference methods," *Bioinformatics*, vol. 27, no. 16, pp. 2263–2270, 2011.

[31] Spellman et al., "Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization." *Molecular biology of the cell*, vol. 9, pp. 3273–3297, 1998.

[32] Cantone et al., "A Yeast Synthetic Network for In Vivo Assessment of Reverse-Engineering and Modeling Approaches," *Cell*, vol. 137, no. 1, pp. 172–181, Apr. 2009.

[33] R. Milner, *Communication and concurrency*, ser. PHI Series in computer science. Prentice Hall, 1989.

[34] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis, "Advances to bayesian network inference for generating causal networks from observational biological data," *Bioinformatics*, vol. 20, no. 18, pp. 3594–3603, 2004.

[35] M. Calder, V. Vyshemirsky, D. Gilbert, and R. Orton, "Analysis of signalling pathways using the prism model checker," in *Proceedings of Computational Methods in Systems Biology (CMSB 2005*, 2005, pp. 179–190.

[36] M. Calder, S. Gilmore, and J. Hillston, "Modelling the influence of rkip on the erk signalling pathway using the stochastic process algebra pepa." in *T. Comp. Sys. Biology*, 2006, pp. 1–23.

[37] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn, "Probabilistic model checking of complex biological pathways," *Theoretical Computer Science*, vol. 391, no. 3, pp. 239 – 257, 2008, converging Sciences: Informatics and Biology.

[38] F. Ciocchetta, S. Gilmore, M. L. Guerriero, and J. Hillston, "Integrated simulation and model-checking for the analysis of biochemical systems," *Electronic Notes in Theoretical Computer Science*, vol. 232, no. 0, pp. 17 – 38, 2009, proceedings of the Third International Workshop on the Practical Application of Stochastic Modelling (PASM 2008).

[39] M. Z. Kwiatkowska, G. Norman, and D. Parker, "Using probabilistic model checking in systems biology," *SIGMETRICS Performance Evaluation Review*, vol. 35, no. 4, pp. 14–21, 2008.

[40] J. Hillston, *A Compositional Approach to Performance Modelling*. New York, NY, USA: Cambridge University Press, 1996.

[41] G. De Ruvo and A. Santone, "An eclipse-based editor to support LOTOS newcomers," in *2014 IEEE 23rd International WETICE Conference, WETICE 2014, Parma, Italy, 23-25 June, 2014*, 2014, pp. 372–377. [Online]. Available: http://dx.doi.org/10.1109/WETICE.2014.39

[42] R. Saito, M. E. Smoot, K. Ono, J. Ruscheinski, P.-L. Wang, S. Lotia, A. R. Pico, G. D. Bader, and T. Ideker, "A travel guide to cytoscape plugins," *Nature methods*, vol. 9, no. 11, pp. 1069–1076, 2012.