Infer Gene Regulatory Networks from Time Series Data with Formal Methods

Michele Ceccarelli[2,3], Luigi Cerulo[2,3], Antonella Santone[1]
[1] Dept. of Engineering, University of Sannio, Benevento - Italy
[2] Dept. of Science and Technology, University of Sannio, Benevento - Italy
[3] BioGeM s.c.a r.l., Institute of Genetic Research “Gaetano Salvatore”, Ariano Irpino (AV) - Italy

***Abstract*—Reverse engineering of regulatory relationships from genomics data is emerging as crucial to dissect the complex underlying regulatory mechanism occurring in a cell. In this paper we propose a novel reverse engineering algorithm that makes use of formal methods, usually adopted in engineering to specify and verify concurrent software and hardware systems. With a formal specification of gene regulatory hypotheses we are able to prove mathematically whether a time course experiment belongs or not to the formal specification, determining in fact whether a gene regulation exists or not. The method is capable to detect both direction and sign (inhibition/activation) of regulations whereas most of literature methods which are limited to undirected and/or unsigned relationships. The method, empirically evaluated on experimental and synthetic datasets, reaches high levels of accuracy, outperforming literature methods in terms of precision and recall, despite the computational cost increases exponentially with the size of the network.**

Keywords: *Reverse Engineering; Gene Regulatory Network; Formal Methods.*

I. Introduction

The reconstruction of molecular interactions between relevant biological entities, such as genes and proteins, is crucial to dissect the complex mechanism underlying the behaviors of cells living in an environment. The number of methods, aiming at inferring interaction hypotheses from high throughput genomic and proteomic data, rapidly increased in the last years [1], [2], [3], [4], [5], [6], [7]. Most of them focus on identifying gene regulatory relationships, *i.e.* transcriptional activities arising among active genes in a cell. Gene expression is mainly regulated by transcription factors, *i.e.* proteins coded by specific genes, that bind, alone or with other proteins in a complex, targets cis-regulatory regions and control target transcriptional activity by promoting or blocking the recruitment of RNA polymerase. From a computational point of view the reconstruction of gene regulatory networks is an undetermined problem as the large number of possible solutions is typically high in contrast to the number of available independent data points. Many possible solutions can fit the available data, explaining the data equally well, but only one of them can be the biologically true solution [3]. Different strategies have been proposed in literature to reduce the search space and/or extend the amount of independent information. 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 [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 to effectively represent the biological systems without precise quantitative relationships. The most representatives are boolean networks and Bayesian models [9], [10]. Correlation methods aims 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 outperforms generally the previous mentioned unsupervised approaches they require a set of genes (training set), where the information that they interact is known in advance, to estimate a function to discriminate whether a new gene interactions exists. This is basically not a problem in known 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 focus on an alternative method, based on formal methods, to the authors' knowledge, never used before to infer gene regulatory networks. Actually, in literature formal methods have been already used, but mainly to analyze, verify and simulate the behavior of known regulatory systems [19], [20], [21], [22]. Formal methods are techniques for specifying and verifying concurrent systems developed over the past three decades. Among all we adopt the Calculus of Communicating Systems (CCS) of Milner [23] which is one of the best known process algebras, a small class of formal methods that find their roots in algebra. We show how to use CCS and model checking to infer gene regulatory networks from time course experimental data. Discretized time-series data are used to build a CCS 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, *i.e.* checking whether the CCS model satisfies the temporal properties, expressed in a 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 paper is organized as follows. The next Section introduces the approach and outlines the empirical evaluation procedure. Section on results reports and discusses the outcomes of the study, and the last Section concludes the paper outlining directions for future work.

II. Methods

A. Basic definitions

Formal methods are mathematically based languages, techniques, and tools for specifying and verifying complex systems. For applying formal methods, we need:

(i) A precise notation for defining systems. For this aim, we use Milner's Calculus of Communicating Systems (CCS) [23]. CCS is one of the most well known process algebras. CCS contains basic operators to build finite processes, communication operators to express concurrency, and some notion of recursion to capture infinite behaviour. The semantics of a CCS term p is precisely defined by means of the structural operational semantics. The semantic definition is given by a set of conditional rules describing the transition relation of the automaton corresponding to the behavior expression defining p. Throughout the paper we use the following CCS operators:

- "+": the process $p + q$ is a process that non-deterministically behaves either as p or as q.
- ";": this operator is suitably used to express the sequentialization between two processes. The process $p; q$ means that p must terminate before the process q can start its execution.
- "$\|$": the process $p \| q$ represents the parallel execution of p and q and terminates only if both processes terminate.

(ii) A precise notation for defining properties. This need can be solved using a temporal logic as, for example, the mu-calculus logic [24]. Temporal logics present constructs allowing to state in a formal way that, for instance, all scenarii will respect some property at every step, or that some particular event will eventually happen, and so on.

(iii) An algorithm to check if a system satisfies a property. For this last need, several algorithms exist. The most used verification methodology is model checking [25]. In the model checking framework, systems are modeled as transition systems and requirements are expressed as formulae in temporal logic. A model checker then accepts two inputs, a transition system and a temporal formula, and returns true if the system satisfies the formula and false otherwise.

B. Inference algorithm

Given a set of n genes and a set of k time course experiments, the approach starts considering a directed graph containing n vertices (Figure 1). The vertices of the graph are biologically mapped to genes, while the edges express the effects of genes on each other. Two types of edges exit. An edge $X \longrightarrow Y$ means that the increment of X in time causes a direct increment of Y (activation). On the other hand, an edge $X \dashv Y$ means that the increment of X in time causes a direct decrement of Y (inhibition). Initially, we suppose that the graph is a complete digraph, *i.e.*, a directed graph in which every pair of distinct vertices is connected by two pairs (both for $\dashv$ and $\longrightarrow$) of unique edges (one in each direction).

Each time course experiment is firstly discretized, *i.e.*, the expression level of each gene X is mapped to one of the following possible states: Up, Basal, and Down. Among approaches presented in literature [26], we adopt an equal frequency discretization method that divides the expression levels into 3 intervals containing approximately the same number of expression values. From discretized time-series data a CCS model has been built, which take into account the evolution of gene expressions over time. For each edge e of the graph we generate an edge–process of the form $P_e = p_1 + \cdots + p_k$, where each p_i ($\forall i \in [1..k]$) is a perturbation–process referring to the i-th perturbation taken into account (k is the number of perturbed experiments available). The + operator refers to the fact that each perturbation is independent from each other. The i-th perturbation–process p_i ($\forall i \in [1..k]$) is modeled as a sequence of network–state–processes of the form $q_1; \ldots; q_m$ indicating the sequence of states in which the gene network can be during the i-th perturbation. In particular, starting form q_1, each network–state–process q_j ($j \in [1..m-1]$) evolves into the network–state–process q_{j+1} and is modeled as the parallel composition of n gene–state–processes, specifying one of the possible discrete state of each gene (Up, Basal, Down). The CCS model is then used to prune the original graph checking whether all edges are correct (*i.e.*, explains the data or not) and eliminating the arcs that do not satisfy certain properties expressed in a temporal logic. This is performed by using model checking techniques. Two properties have been defined to model two fundamental biological functions between two genes X and Y: gene activation and gene inhibition. For each activator edge $e : X \longrightarrow Y$, we check on the CCS model the following property, expressed using the selective mu-calculus logic [27]:

$$\psi = [X\_Up]_{\varnothing} \langle Y\_Up \rangle_{\varnothing} [Y\_Down]_{\varnothing} \, \mathtt{ff}$$

which states for: "*if* the activator edge e exists, *then* whenever the gene X is Up the gene Y must become Up and then it cannot become Down". If the model does not satisfy ψ, the edge e is eliminated. Similarly, for each inhibitor edge $e : X \dashv Y$, we check on the CCS model the following property:

$$\varphi = [X\_Up]_{\varnothing} \langle Y\_Down \rangle_{\varnothing} [Y\_Up]_{\varnothing} \, \mathtt{ff}$$

which states for: "*if* the inhibitor edge e exists, *then* whenever the gene X is Up the gene Y must become Down and then it cannot become Up".

Issues related to complexity and scalability

The complexity of the extraction of the CCS model from discretized experimental data is linear in the length of the timeseries, while model checking algorithms for branching time logic run in time $O(nm)$ [25], where n is the size of

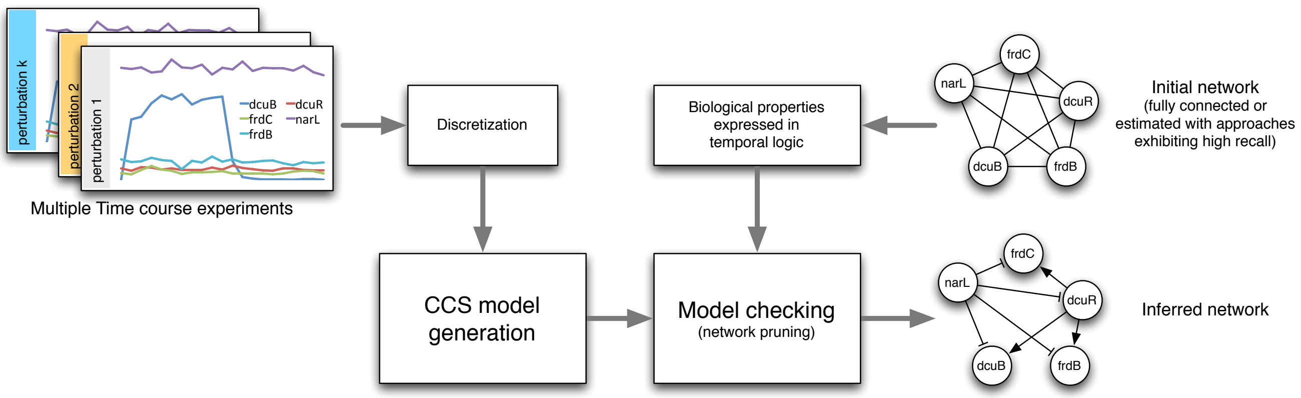

Fig. 1. The inference process.

the transition system, which depends on the number of genes and the number of expression level transitions, and m is the size of an alternation free temporal logic formula, which in our case depends on the size of the two previously reported rules. From the scalability point of view, it is well-known that most current formal verification tools are mainly applicable to small-scale applications because of *the state explosion problem* which says that the state space grows exponentially in the number of concurrent processes (*i.e.* in our case the number of genes). In modeling real-world biological systems we compose in parallel all genes, each one specified as a CCS process by growing exponentially the state space. Our current implementation is able to manage in a reasonable time networks with up to 100 genes. Networks with more genes can be treated with recently proposed techniques, such as reduction techniques based on partial order techniques [28], compositional techniques [29], [30], and abstraction approaches [31]. We integrate our approach with the abstraction technique based on the selective mu-calculus [27]. Often the property one wants to check does not concern the whole transition system, but only some parts of it. An approach to reduce the number of states is the definition of suitable abstraction criteria by means of which a smaller transition system can be obtained, including only the parts that “influence” the property. In [27] a temporal logic is proposed, called *selective mu-calculus*, which has the characteristic that each formula allows immediately pointing out the parts of the transition system that do not alter the truth value of the formula itself.

To face scalability issues we also apply another improvement to our approach. As shown in Figure 1, instead of starting from a complete digraph G, we can use a network generated with an approach exhibiting high recall. For this purpose, we have developed an *empirical method* that generates a graph with less edges than G. The empirical method is not based on logic and model checking but it uses wrapping techniques on times series data. In particular, this method starts with an empty network. Through some perturbations, it simulates the deactivation and activation of a single gene at a time and observes the behavior of other genes. If some conditions hold it introduces edges. More precisely, an activator edge between A and B ($A \longrightarrow B$) is introduced if disabling the gene A, the value of B decreases, or if activating the gene A, the value of B increases. Dually for the inhibitor edges. This method provides a good recall, to the detriment of the precision, *i.e.,* with respect to the real gene regulatory network, more arcs are added. The graph generated by the empirical method becomes the input of the formal method described in Section II-B which is able to prune more false positive arcs.

C. *Empirical evaluation procedure*

We empirically evaluate the proposed approach (in the following denoted FormalM) against a selection of literature methods especially suited for the reconstruction of gene regulatory networks from time series data, that are: ARACNE [11], CLR [14], TD–Aracne [12], TSNIF [8], and BANJO [10].

ARACNE, CLR, and TD-Aracne adopt mutual information to compare expression profiles from a set of gene expression data and different network pruning heuristics. BANJO is based on Bayesian networks and can infer gene networks from steady-state and/or multiple time-series gene expression data. BANJO outputs a signed directed graph but is not able to infer networks involving cycles. TSNIF describes the regulatory network as a system of linear ordinary differential equations representing the rate of synthesis of a transcript as a function of the concentrations of every other transcript in a cell, and the external perturbation. TSNIF identifies both directions and signs of relationships. With the exception of BANJO all other considered methods assume that a single perturbation experiment is performed (e.g. treatment with a compound, gene knock down/up, etc.) and a number of time points following the perturbation are measured, thus taking, in fact, no advance from multiple perturbation experiments. To avoid such a limitation for ARACNE, CLR, and TD-Aracne we concatenate the time series of each perturbation into a single huge time series. Instead, for TSNIF we aggregate the predicted networks obtained for each perturbation into a final network by summing the edge weights of each outcome. Such tools are configured and tuned according to their user guides and/or published papers in order to obtain their best exhibition in terms of precision and recall. 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 [32]. 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 experimental data we adopt three public available datasets:

1) *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 public available by Ronen *et al.* [33].

2) *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 public available by Spellman *et al.* [34].

3) *IRMA*. An in vivo fully controlled experimental network built in the *S. cerevisiae* and composed of five genes [35] 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.

To estimate the unknown performance of a methods we adopt the measures of precision and recall, also known respectively as Positive Predictive Value (PPV) and Sensitivity, and their F-measure (Fm), defined as follows:

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

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 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, thus true positives are counted taking into account both direction and sign.

With simulated datasets we evaluate the performance of each method against the following influencing factors:

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* [32]. We generate 20 random networks of n genes where $n \in \{4, 5, 6, 10, 15, 20, 30, 50, 70, 100\}$. For each random network we generate $2n + 1$ time course experiments of 100 time points each. Simulated time series are generated following the DREAM4 challenge procedure consisting of one experiment without perturbations, n knock down experiments, and n knock up experiments, one for each gene. The knock down/up stimulus is applied at time point 0 and removed at the middle of time series, i.e. time point 50.

Noise in data. Noise in data due to measurement errors is a critical factor influencing in general the performance of machine learning methods. GeneNetWeaver allows for generating time series with a mix of Gaussian and log-normal noise of different magnitude which has been shown is an effective model of noise in microarray experiments. We consider 5 increasing levels of noise (n_0 refers to data without noise) modeled with the following mix of Gaussian and Log-normal standard deviation pairs: $ns_1 = \{0.025, 0.075\}$, $ns_2 = \{0.050, 0.150\}$, $ns_3 = \{0.075, 0.225\}$, $ns_4 = \{0.100, 0.300\}$, $ns_5 = \{0.125, 0.375\}$.

Number of available perturbations. Typically the availability of an insufficient number of experiment perturbations allows for converging to many suboptimal solutions that can fit the available data equally well, but only one of them can be the biologically true solution [3], thus affecting negatively the reconstruction of gene regulatory networks. We sample without replacement 10 times a random subset of r pairs of knock down/up perturbations from the total number of $2n$ available knock down/up experiments generated with GeneNetWeaver. We vary r from 1 to n obtaining a representative sample consisting of almost $10n$ pairs of knock down/up experiments out of 2^n, which is the total population.

III. Results and Discussion

Figure 2 reports results of experiments performed with in silico datasets showing factors that could influence the prediction accuracy. Figure 2A shows the average performance of each method at different network size obtained with the full set of perturbations and expression levels generated without noise (n_0). It can be noticed that the performance decreases with the number of genes (shown in logarithmic scale). Each method decreases at a different rate. FormalM preserves a quite good performance also with large networks ranging between 60% and 78% (78%–100% Precision, 55%–65% Recall). Instead all other methods drop to almost 10% on networks of 100 genes. With small networks (number of genes ranging between 4 and 6) BANJO and TD-Aracne exhibit a similar performance while their performance decrease at different rate on larger networks, with BANJO decreasing more faster than TD-Aracne. TSNIF performs better than correlation methods but worser than TD-Aracne and BANJO for small networks. Figure 2B shows the average performance of each method at different noise levels

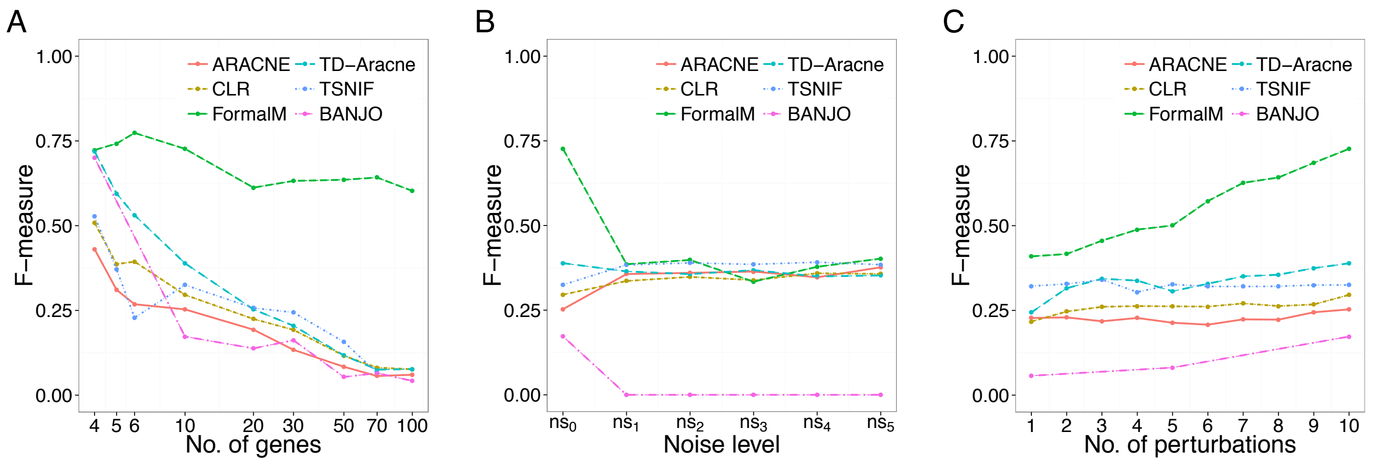

Fig. 2. Simulated experiments results with different influencing factors: A) network size; B) noise level; C) number of available perturbations

size obtained with gene networks of size 10 and the full set of perturbations (*i.e.* 1 unperturbed and 10 knock down/up time series pairs). For BANJO and FormalM the performance decreases with the noise level, instead the performance of correlation methods and TSNIF is almost not affected by noise. In particular the F-measure of FormalM drops from 75% (89% Precision, 57% Recall) to 40% (32% Precision, 56% Recall), which is the level of other methods. The drop affects mainly the Precision and is mainly due to time series discretization which is critical for noisy data [26]. Figure 2C shows the average performance of each method at different number of perturbations obtained with gene networks of size 10 and noise level n_0. The performance of each method increases with the number of available perturbations attesting that more data points, coming form independent experiments, allows for recovering more correct gene connections. The best increment rate can be observed for FormalM and BANJO, both exploiting the presence of multiple time course experiments. In particular, FormalM exhibits an F-measure of 40% (30% Precision, 90% Recall) with 1 pair of knock down/up experiments conducted on a random gene, and reaches the level of 75% (89% Precision, 57% Recall) when all genes were perturbed with a knock down/up experiment pair.

Tables I, II, and III show the performance obtained with experimental datasets in terms of precision and recall computed on undirected, directed, and signed graphs. All methods perform well with undirected relationships while for directed and signed graphs differences among methods is remarkable. On *E. coli* SOS network correlation methods (ARACNE and CLR) outperform all other methods in undirected predictions, whereas BANJO and TSNIF that perform better in directed and signed prediction respectively. FormalM is able to predict correctly the direction of many relationships (it scores immediately after BANJO) but classifies as activations almost all inhibitions of the *E. coli* SOS network. On *S. Cerevisiae* cell cycle network FormalM outperforms all other methods with a high recall (100% for directed and undirected, 87% for signed) and a quite good 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%). On IRMA network TSNIF outperform all other methods in predicting the direction, while BANJO reveals better to predict the sign of relationships. Anyhow FormalM and TD-Aracne are able to predict the direction with a quite good accuracy (F-measure respectively of 57% and 44%).

IV. Conclusions and Future Work

We introduced a gene network reverse engineering approach based on formal methods, a set of techniques for specifying and verifying concurrent systems. With respect to the most of literature reverse engineering methods, our approach is able to detect both direction and sign (inhibition/activation) of relationships and exploits multiple perturbation data. Results reported in this work are encouraging and promising. Experiments performed on simulated data attested the superiority of our approach with respect to the most relevant literature methods. Moreover our method preserves high level of performance also with large networks and with a few number of time series experiments although computational cost is a crucial drawback of our approach. In vitro experiments, with real noisy experimental data, showed that our method exhibits a performance that is higher or at least comparable to other literature methods but more experiments are necessary to make more general conclusion. In this work we enclosed in the model just two fundamental biological rules, modeling activator and inhibitor relationships. The underlying process algebra allows for adding more biological knowledge and in future work we would investigate for including new rules, such as those modeling typical regulatory patterns which recur throughout networks (*i.e.* feed-forward loops, single-input modules, dense overlapping regulons).

Acknowledgements

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

TABLE I
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
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**
FormalM	Signed	0.04	0.25	0.06
TSNIF	Signed	0.12	0.25	**0.17**
BANJO	Signed	0.05	0.12	0.07

TABLE II
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
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
FormalM	Signed	0.18	0.87	**0.30**
TSNIF	Signed	0.08	0.09	0.08
BANJO	Signed	0.17	0.22	0.19

TABLE III
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
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
FormalM	Signed	0.25	0.62	0.36
TSNIF	Signed	0.43	0.38	0.40
BANJO	Signed	0.36	0.50	**0.42**

REFERENCES

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

[2] 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.

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

[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] G. Batt, D. Ropers, H. D. Jong, J. Geiselmann, R. Mateescu, and D. Schneider, "Validation of qualitative models of genetic regulatory networks by model checking: Analysis of the nutritional stress response in escherichia coli," *Bioinformatics*, vol. 21, pp. 19–28, 2005.

[20] A. Regev, W. Silverman, and E. Y. Shapiro, "Representation and simulation of biochemical processes using the pi-calculus process algebra," in *Pacific Symposium on Biocomputing*, 2001, pp. 459–470.

[21] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani, "A bayesian approach to model checking biological systems," in *CMSB*, 2009, pp. 218–234.

[22] C. Li, M. Nagasaki, C. H. Koh, and S. Miyano, "Online model checking approach based parameter estimation to a neuronal fate decision simulation model in caenorhabditis elegans with hybrid functional petri net with extension," *Mol. BioSyst.*, vol. 7, pp. 1576–1592, 2011.

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

[24] C. Stirling, "An introduction to modal and temporal logics for ccs," in *Concurrency: Theory, Language, And Architecture*, ser. Lecture Notes in Computer Science, A. Yonezawa and T. Ito, Eds., vol. 491. Springer, 1989, pp. 2–20.

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

[26] 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.

[27] R. Barbuti, N. D. Francesco, A. Santone, and G. Vaglini, "Selective mu-calculus and formula-based equivalence of transition systems," *J. Comput. Syst. Sci.*, vol. 59, no. 3, pp. 537–556, 1999.

[28] A. Santone and G. Vaglini, "Partial order interpretation of a mu-calculus-like temporal logic," in *ICSOFT*, 2013.

[29] A. Santone, G. Vaglini, and M. L. Villani, "Incremental construction of systems: An efficient characterization of the lacking sub-system," *Sci. Comput. Program.*, vol. 78, no. 9, pp. 1346–1367, 2013.

[30] A. Santone, "Automatic verification of concurrent systems using a formula-based compositional approach," *Acta Inf.*, vol. 38, no. 8, pp. 531–564, 2002.

[31] A. Santone and G. Vaglini, "Abstract reduction in directed model checking ccs processes," *Acta Inf.*, vol. 49, no. 5, pp. 313–341, 2012.

[32] 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.

[33] M. Ronen, R. Rosenberg, B. I. Shraiman, and U. Alon, "Assigning numbers to the arrows: Parameterizing a gene regulation network by using accurate expression kinetics," *PNAS*, vol. 99, no. 16, pp. 10 555–10 560, Aug. 2002.

[34] 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, no. 12, pp. 3273–3297, Dec. 1998.

[35] 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.