Modular analysis of gene networks by linear temporal logic

Sohei Ito[1,*], Takuma Ichinose[2], Masaya Shimakawa[2], Naoko Izumi[3], Shigeki Hagihara[2], Naoki Yonezaki[2]

[1]The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

[2]Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo, 152-8552, Japan

[3]Jumonji University, 2-1-28 Sugasawa, Niiza, Saitama 352-8510, Japan

Summary

Despite a lot of advances in biology and genomics, it is still difficult to utilise such valuable knowledge and information to understand and analyse large biological systems due to high computational complexity. In this paper we propose a modular method with which from several small network analyses we analyse a large network by integrating them. This method is based on the qualitative framework proposed by authors in which an analysis of gene networks is reduced to checking satisfiability of linear temporal logic formulae. The problem of linear temporal logic satisfiability checking needs exponential time in the size of a formula. Thus it is difficult to analyse large networks directly in this method since the size of a formula grows linearly to the size of a network. The modular method alleviates this computational difficulty. We show some experimental results and see how we benefit from the modular analysis method.

1 Introduction

In the recent progress in biology and genomics, we have much information about gene regulation in many species from several published databases [1, 2, 3, 4]. It, however, is still difficult to utilise such data in understanding and analysing biological systems in some large scale using them. One reason for such difficulty is high computational complexity. In some algorithms, analysis of a large system needs much time and space which hinders the direct application of them to real problems.

The qualitative analysis method proposed by authors [5] also suffers from high computational complexity in analysing large networks. This method enables analysis of gene regulatory networks without real kinetic parameters, which is useful when we do not have such information but are interested in checking some qualitative property, e.g. whether a certain gene oscillates, whether a gene is always active, and so on. If such property is computationally possible, biologists are motivated to check whether the property is really observed.

In this method, behaviours are captured as transition systems using propositions for gene states (ON or OFF) and for thresholds on gene activation and inhibition. We characterise possible behaviours of networks by specifying changes in concentration levels of gene products and

*To whom correspondence should be addressed. Email: ito@kb.is.s.u-tokyo.ac.jp

changes in gene states using linear temporal logic (LTL) [6]. Expected biological properties such as reachability, stability and oscillation are also described in LTL. We check satisfiability of these formulae to investigate whether some or all behaviours satisfy the corresponding biological property.

Due to the complexity of LTL satisfiability checking (PSPACE-complete [7]), known algorithms have exponential time complexity with respect to the length of an input formula. The length of a formula specifying possible behaviours of a network is proportional to the size of the network. Thus analyses of large networks are generally intractable. To circumvent this computational difficulty, here we develop a modular analysis method with which we can analyse a large network by individually analysing its subnetworks and integrating the results of their analyses. Experimental results shows that this modular method is actually effective in analysing large networks.

This paper is organised as follows. Section 2 introduces the logical structure which describes abstract behaviours of gene regulatory networks. In Section 3, we review our qualitative method in which networks are analysed by LTL satisfiability checking. In Section 4, we introduce the modular analysis method and show some experimental results. In Section 5, we discuss some related works. The final section offers some conclusions and discusses future directions.

2 Logical conceptualisation of behaviours

In gene regulation, a regulator is often inefficient below a threshold concentration, and its effect rapidly increases above this threshold [8]. The sigmoid nature of gene regulation is shown in Fig. 2, where gene u activates v and inhibits w (Fig. 1). Each axis represents the concentration of products for each gene.

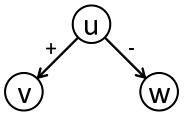

Figure 1: Gene u activates v and inhibits w.

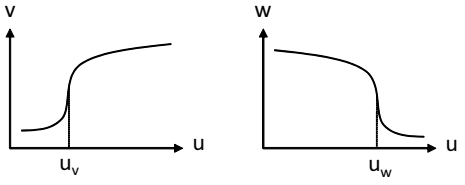

Figure 2: Regulation effect.

Important landmark concentration values for u are, 1) the level u_v at which u begins to affect v, and 2) the level u_w at which u begins to affect w. In this case, whether genes are active or not can be specified by the expression levels of their regulator genes. If the concentration of u exceeds u_v then v is active (ON), and if the concentration of u exceeds u_w then w is not active (OFF). We exploit this switching view of genes to capture behaviours of gene networks in transition systems.

We now illustrate how we capture behaviours of gene regulatory networks as transition systems using a simple example network (Fig. 3) in which gene x activates gene y and gene y activates gene z.

Let the threshold of x for y be x_y and that of y for z be y_z. We consider the behaviour depicted in Fig. 4 and consider to express it as a transition system. In this behaviour, x begins its

Figure 3: Simple example.

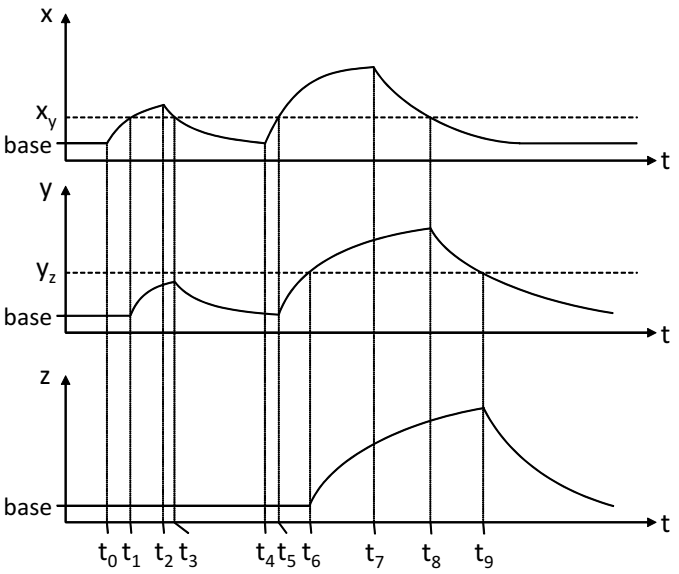

Figure 4: Change of concentrations with time.

expression at time t_0; that is, the concentration of its products begins to increase. At time t_1, the concentration of the products of x exceeds x_y, which is the threshold for the activation of y. Thus y begins to be expressed at t_1. At time t_2, x stops being expressed and the concentration of its products begins to decrease. At time t_3, the concentration of products of x falls below x_y and y stops being expressed; that is, the concentration of y begins to decrease. After a while, x begins to be expressed again at time t_4. At t_5, y begins to be expressed. In this case, y crosses the activation threshold for z at time t_6 and z begins to be expressed. At t_7, x stops being expressed and begins to decrease. At t_8, x falls below x_y and y stops being expressed. At t_9, y falls below y_z and z stops being expressed, after which x, y and z stay at their basal level.

We introduce some logical propositions to obtain a symbolic representation of behaviours of this network. Based on the above observation, we introduce propositions that represent whether genes are active or not (ON or OFF) and whether concentrations of products of genes exceed threshold values. In this network, we introduce the propositions on_x, on_y, on_z, x_y and y_z. Propositions on_x, on_y, on_z mean whether or not gene x, y or z is active respectively, x_y whether gene x is expressed *beyond* the threshold x_y[1], and y_z whether gene y is expressed *beyond* the threshold y_z.

Using these propositions, we discretise the above behaviour to the sequence of states (called *transition system*) shown in Fig. 5, where $s_0, \ldots, s_{10}$ are states, arrows represent state transitions that show the temporal evolution of the system, and the propositions below each state means that they are true in that state.

State s_0 represents the interval $[0, t_0)$, state s_1 represents the interval $[t_0, t_1)$, ... and state s_{10} represents $[t_9, \infty)$.

A single state transition can represent any length of time, since the actual duration of the transition (in real time) is immaterial[2]. Therefore, the difference between $t_2 - t_0$ and $t_7 - t_4$, the

[1]Note that the symbol x_y is used for both the threshold and proposition but we can clearly distinguish them from the context

[2]This property is called *speed independence* [9].

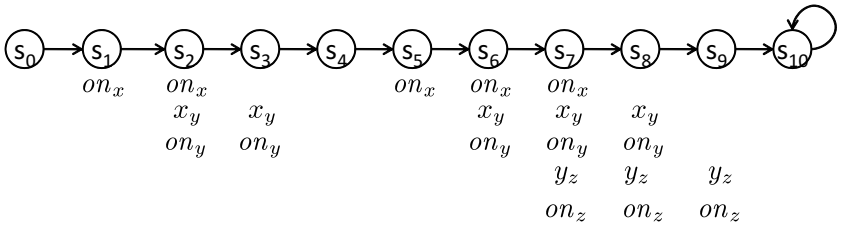

Figure 5: State transition system corresponding to Fig. 4.

duration of the input signal to x, in Fig. 4 is not captured directly. The transition system depicted in Fig. 5 captures whether the concentration of y exceeds y_z; that is, we can infer that the latter duration is sufficiently long for x to activate y by comparing the propositions that are true in s_1 to s_3 and in s_5 to s_9. Moreover, the real values of thresholds are irrelevant. Propositions such as x_y merely represent the fact that the concentration of x is above the threshold level at which x affects y.

In our abstraction, behaviours are identified with each other if they have the same transition system. Such logical abstraction preserves essential qualitative features of the dynamics [10, 8].

3 Qualitative analysis of gene regulatory networks in LTL

In this section, we show how to analyse behaviours of gene regulatory networks using LTL. First we introduce the time structure of LTL. If A is a finite set, A^ω denotes the set of all infinite sequences on A. The i-th element of $\sigma \in A^\omega$ is denoted by $\sigma[i]$.

Definition 1 *Let AP be a set of propositions. A* time structure *is a sequence $\sigma \in \mathfrak{P}(AP)^\omega$ where $\mathfrak{P}(AP)$ is the powerset of AP, i.e. the set of all subsets of AP.*

We next define formulae in LTL.

Definition 2 *Let AP be a set of propositions. Then $\forall p \in AP$, p is a formula. If ϕ and ψ are formulae, then $\neg\phi, \phi \wedge \psi, \phi \vee \psi$, and $\phi U \psi$ are also formulae.*

We introduce the following abbreviations: $\bot \equiv p \wedge \neg p$ for some $p \in AP$, $\top \equiv \neg\bot$, $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$, $\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$, $F\phi \equiv \top U \phi$, $G\phi \equiv \neg F \neg\phi$, and $\phi W \psi \equiv (\phi U \psi) \vee G\phi$.

Intuitively, $\neg\phi$ means 'ϕ is not true', $\phi \wedge \psi$ 'both ϕ and ψ are true', $\phi \vee \psi$ 'ϕ or ψ is true', $\phi U \psi$ 'ϕ continues to hold until ψ holds', $\bot$ a false proposition, $\top$ a true proposition, $F\phi$ 'ϕ holds at some future time', $G\phi$ 'ϕ holds globally', and $\phi W \psi$ is the 'weak until' operator in that ψ is not obliged to hold, in which case ϕ must always hold. The formal semantics are given below.

Definition 3 *Let σ be a time structure and ϕ be a formula. We write $\sigma \models \phi$ for 'ϕ is true in σ'.*

The satisfaction relation $\models$ is defined inductively as follows:

$$\begin{array}{lll} \sigma \models p & \text{iff} & p \in \sigma[0] \textit{ for } p \in AP \\ \sigma \models \neg\phi & \text{iff} & \sigma \not\models \phi \\ \sigma \models \phi \wedge \psi & \text{iff} & \sigma \models \phi \textit{ and } \sigma \models \psi \\ \sigma \models \phi \vee \psi & \text{iff} & \sigma \models \phi \textit{ or } \sigma \models \psi \\ \sigma \models \phi U \psi & \text{iff} & (\exists i \geq 0)(\sigma^i \models \psi \textit{ and } \forall j(0 \leq j < i)\sigma^j \models \phi) \end{array}$$

where $\sigma^i = \sigma[i]\sigma[i+1]\ldots$, the i-th suffix of σ.

Finally we introduce the notion of *satisfiability*.

Definition 4 *An LTL formula ϕ is* satisfiable *if there exists some time structure σ such that $\sigma \models \phi$. A time structure σ such that $\sigma \models \phi$ is called a* model *of ϕ.*

In our analysis method, we describe a behaviour description φ for a given network and a biological property ψ which we are checking against the network. We can perform two kinds of analysis in this method. One is to check satisfiability of $\varphi \wedge \psi$. If $\varphi \wedge \psi$ is satisfiable then there is some behaviour σ such that $\sigma \models \varphi \wedge \psi$. Thus we know there is some behaviour of the network which satisfies the property. The other is to check unsatisfiability of $\varphi \wedge \neg\psi$, which means there is no behaviour such that it satisfies φ but does not satisfy ψ. Thus we know all behaviours of the network satisfies the property.

Now we review how we describe behaviour description of a given network in our method [5], using an example gene network of mucus production in *Pseudomonas aeruginosa* [11, 12] depicted in Fig. 6.

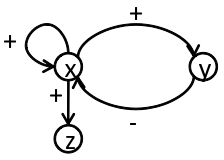

Figure 6: The network of mucus production in *P. aeruginosa*, where x positively regulates mucus production, represented as z, and y inhibits x, which x positively regulates.

In this network, z represents alginate synthesis (i.e. mucus production), gene x activates mucus production, and gene y inhibits gene x. We introduce the set of propositions $\{on_x, on_y, on_z, x_x, x_y, x_z, y_x\}$, where z is not a gene, but on_z means that mucus is produced.

Gene y is positively regulated by gene x. Therefore y is active if x is expressed beyond the threshold for y. This can be described as $G(x_y \rightarrow on_y)$ in LTL which says that if gene x is expressed beyond the threshold x_y, gene y becomes ON. Similarly, activation for z is described as $G(x_z \rightarrow on_z)$. For x's activation, we know that x is active when x is expressed beyond the threshold x_y and y below the threshold y_x (since y inhibits x). This is described as $G(x_x \wedge \neg y_x \rightarrow on_x)$. Similarly we have the formulae for genes' inactivation such as $G(\neg x_y \rightarrow \neg on_y)$, $G(\neg x_z \rightarrow \neg on_z)$ and $G(\neg x_x \vee y_x \rightarrow \neg on_x)$. Now we describe the order of threshold values x_x, x_y and x_z. It is known that x_z is the highest [12]. Thus there are two possibilities for the order, $x_x < x_y < x_z$ or $x_y < x_x < x_z$. Suppose we choose the former, then we know that if x

is beyond x_y, x is also beyond x_x. This can be described as $G(x_y \rightarrow x_x)$. Similarly we have $G(x_z \rightarrow x_y)$.

If gene x is active, it begins to be expressed and in some future its level will reach the first threshold x_x. Otherwise gene x becomes OFF before its level reaches x_x. This can be described as $G(on_x \rightarrow F(x_x \vee \neg on_x))$. Similarly, if x is still active, it continues growing and will reach x_y, except that gene x becomes OFF before reaching x_y. Typical situations are illustrated in Fig. 7. This can be described as $G(on_x \wedge x_x \rightarrow x_x U(x_y \vee \neg on_x))$. The situation is similar for growing from x_y to x_z. Thus we have a similar formula $G(on_x \wedge x_y \rightarrow x_y U(x_z \vee \neg on_x))$. If gene x is expressed beyond x_z, the highest threshold, then it keeps its level as long as gene x is ON. This can be described as $G(on_x \rightarrow x_z W \neg on_x)$.

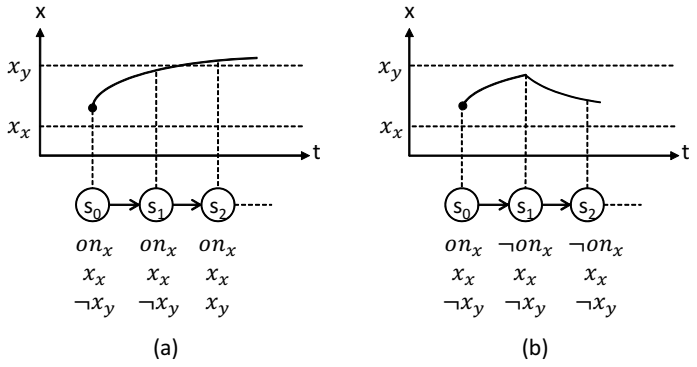

Figure 7: Typical behaviours when gene x is ON. (a) The expression level eventually increases. (b) Gene x becomes OFF before its level reaches x_y

If gene x is not active, its product decreases due to degradation. Thus if x is inactive and the current level of x is over x_z, then it will fall below x_z in some future. Otherwise gene x becomes ON before its level falls below x_z. The situations are similar when the current expression level of x is x_x or x_y. This can be described as $G(\neg on_x \rightarrow F(\neg x_z \vee on_x))$, $G(\neg on_x \wedge x_z \rightarrow x_z U(\neg x_y \vee on_x))$, $G(\neg on_x \wedge x_y \rightarrow x_y U(\neg x_y \vee on_x))$ and $G(\neg on_x \wedge \neg x_x \rightarrow \neg x_x W on_x)$. We have similar formulae for gene y.

The conjunction (i.e. each clause is connected by $\wedge$) of above formulae is a characterisation of possible behaviours of the network. All models which satisfies the formula are (abstracted) possible behaviours of the network.

Let us consider checking multi-stationarity in mucus production in the network, that is, whether the bacteria have two stable behaviours, one is mucoid and the other is non-mucoid. Mucoid stable behaviour can be simply described as $G on_z$ and non-mucoid one as $G\neg on_z$. The result of satisfiability checking was true for both properties (we used our implementation for LTL satisfiability checker based on the Aoshima's algorithm [13]). Therefore, it is computationally *possible* that the wild-type bacteria have both mucoid and non-mucoid behaviour. This result motivates us to verify this hypothesis experimentally.

Now we assume that the negative effect from y to x is superior to the positive effect from x to x. In this case the bacteria may not become mucoid state since $x_y < x_z$. We check this hypothesis. We modify the behavioural specification by replacing the clause $G(\neg x_x \wedge y_x \rightarrow \neg on_x)$ with $G(y_x \rightarrow \neg on_x)$. Then we verify whether the modified behavioural specification with the property $G on_z$ is not satisfiable. This is actually the case for both orderings of x_x and

x_y. This result means the hypothesis that the bacteria may have a stable mucoid state is rebutted by the assumption that the negative effect of y overpowers the positive effect of x.

Analysis in our method is based on LTL satisfiability checking, which is a PSPACE-complete problem [7]. Therefore, the known algorithms require exponential time related to the size of an input formula. The length of a formula specifying possible behaviours of a network is proportional to the size of the network, and accordingly, analyses of large networks are generally intractable in our method. Thus we developed a modular analysis method to enable the analysis of larger networks.

4 Modular analysis of gene regulatory networks

To check satisfiability of a LTL formula, we can construct a Büchi automaton[3] which is equivalent to the formula [14]. The size of corresponding automaton is exponential in the size of an input formula. If the set of words that the automaton accepts is not empty, the formula is satisfiable. It is known that we can check non-emptiness of a Büchi automaton in linear time with respect to the size of an automaton. As a result, LTL satisfiability checking problem can be solved in exponential in the size of an input formula.

It is known that Büchi automata are closed under intersection, that is to say, for any Büchi automata $\mathcal{A}$ and $\mathcal{B}$, one can construct a Büchi automaton that accepts $L(\mathcal{A}) \cap L(\mathcal{B})$, where the alphabet sets of $\mathcal{A}$ and $\mathcal{B}$ are the same. ($L(\mathcal{A})$ means the set of words that are accepted by $\mathcal{A}$.) We exploit this fact in modular analysis method.

Let $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_n$ be a behavioural specification of a network and ψ be a biological property. We assume we are checking whether $\varphi \wedge \psi$ is satisfiable (the situation in checking unsatisfiability of $\varphi \wedge \neg\psi$ is the same). In stead of constructing a Büchi automaton $\mathcal{A}_{\varphi\wedge\psi}$ corresponding to $\varphi \wedge \psi$, we individually construct Büchi automata $\mathcal{A}_{\varphi_1}, \ldots, \mathcal{A}_{\varphi_n}, \mathcal{A}_{\psi}$ corresponding to $\varphi_1, \ldots, \varphi_n, \psi$ respectively. Then we abstract (i.e. forget) local propositions (i.e. propositions which only appear in the formula) in each $\varphi_1, \ldots, \varphi_n, \psi$ from $\mathcal{A}_{\varphi_1}, \ldots, \mathcal{A}_{\varphi_n}, \mathcal{A}_{\psi}$. Finally we intersect abstracted automata $\mathcal{A}^-_{\varphi_1}, \ldots, \mathcal{A}^-_{\varphi_n}, \mathcal{A}^-_{\psi}$, and name it $\mathcal{B}$. Then, we have:

Theorem 1 *$L(\mathcal{A}_{\varphi\wedge\psi}) \neq \emptyset$ if and only if $L(\mathcal{B}) \neq \emptyset$.*

We omit the formal presentation of this framework and proofs due to the page limitations.

This theorem yields our modular analysis method. If a formula φ is large (it is the case when a network is large), we can divide the formula into several parts and construct automata individually, and simplify them by abstracting local propositions and finally construct the intersected automaton and check the emptiness of it. As a consequence, the obtained automaton will be semantically simple and the cost of checking non-emptiness will be reduced. Note that this method, however, needs extra costs of intersecting automata whose complexity is linear in the product of the sizes of intersected automata.

For the analysis of gene regulatory networks, behaviour specification of a network can be decomposed into the specifications for its subnetworks. Therefore, we can take φ_i as a behavioural

[3] A kind of ω-automata accepting infinite words

specification for each subnetwork. The local propositions for φ_i are propositions concerning nodes and edges which are 'confined' to subnetworks, that is to say, nodes that are only connected by edges in the subnetwork. Subnetworks that contain many such local propositions represent a good division. Note that propositions contained in ψ are global propositions.

Here we say that a network A is a subnetwork of a network B if nodes and edges of A are also nodes and edges of B. This definition of subnetworks is just structural and independent of functional characteristics of them. In theory, this definition of subnetworks is sufficient. But in practice, we need a guideline to choose subnetwork decompositions. A decomposition based on the functions of subnetworks may be a good division. We need to investigate this issue further in future.

Experiments. We apply our modular method to networks depicted in Figs.8, 9 and 10. The network in Fig. 8 is an artificial one. Fig. 9 is a network in *Arabidopsis* obtained from ReIN[4] and Fig. 10 is a network in *Escherichia coli* taken from [15]. In these networks, we treat the propositions for thresholds which are locally used in the subnetworks. Gene state propositions (i.e. on propositions) are treated as global and did not abstracted.

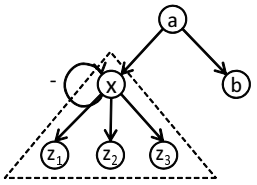

Figure 8: A simple example network

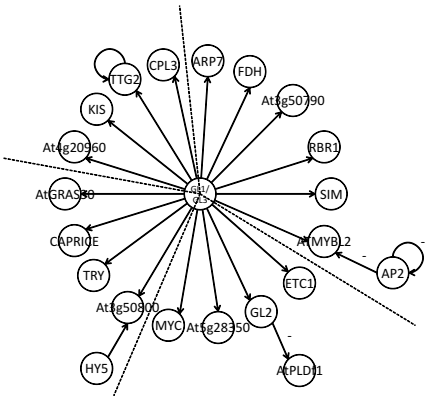

Figure 9: A network in *Arabidopsis*

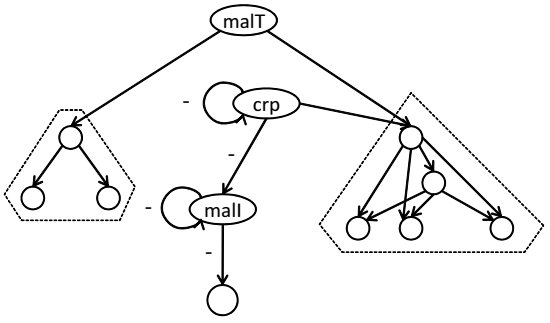

Figure 10: The network from *E. coli* involving the *malT* gene

In Table 1, we show the sizes and number of propositions for each automata and analysis time[5] (the sum of automata construction and non-emptiness testing). We used our implementation

[4]http://arabidopsis.med.ohio-state.edu/REIN/

[5]The following computational environment was used: openSUSE 11.0, Intel(R) Pentium(R) D CPU 3.00GHz and 2GB of RAM.

Table 1: Results

Network	Method	Propositions	Time
Fig. 8	Direct	12	0.020s
	Modular	7	0.052s
Fig. 9	Direct	46	17.357s
	Modular	23	12.631s
Fig. 10	Direct	28	94.454s
	Modular	11	37.365s

for translations from LTL to Büchi automata[6] and determining their intersections.

For the network of Fig. 8, we do not benefit from our modular method. The reason is that the network itself is not large and there are few local propositions. We could not compensate the extra cost of intersecting automata. For the networks of Figs. 9 and 10, modular analysis is beneficial as there are many local propositions. Especially for Fig. 10, in the direct analysis, the non-emptiness testing time is dominant (about 88 sec), but in modular analysis it takes about 26 sec. This improvement is attributed to the reduction of the number of propositions.

5 Related work

In this section, we describe some other qualitative methods for biological systems.

BIOCHAM [16] is a language and programming environment for modeling and simulating biochemical systems. Reactions are defined as rewriting rules like `A + B => C`, and simulations are performed by replacing objects on the left-hand side with those on the right-hand side. The result of simulation are represented as a transition graph whose nodes are possible states of objects. A biological property is given in computational tree logic and checked in the resulting transition graph. In BIOCHAM, presence or absence of objects is the only matter considered in contrast to our method.

SMBioNet [17] is a tool for formally analysing temporal properties of gene regulatory networks. In SMBioNet, genes have concentration thresholds to activate or inhibit each of their regulating genes. A temporal evolution of a system is specified by a transition function on the vectors of expression levels of genes. The specification of behaviours is more flexible in our method than that of SMBioNet in the sense that we can express temporal ordering of event occurrences by LTL.

GNA [18] is a computational tool for the modeling and simulation of gene regulatory networks. GNA achieves simulation using piecewise linear differential equation models and generates state transition systems that represent possible behaviours. This method assumes that the functions of multivariate regulation are known but such functions are unknown in most of networks. Therefore our method is more applicable, considering the current databases of gene regula-

[6]For technical reasons, we used generalised Büchi automata from which we can construct equivalent Büchi automata.

tion such as Reactome[7] [1], GeneCards[8] [2], Metacyc[9][3], Ingenuity® Knowledge Base[10], and KEGG[11][4].

It is also unclear how to conduct modular analysis in the methods discussed here.

6 Conclusion

In this paper, we presented a modular method for analysing the dynamics of gene regulatory networks using LTL satisfiability checking. As experiments show, this method is useful in analysing large networks.

Our approach is based on the correspondence between biological systems and reactive systems. A reactive system is a system that responds to requests from an environment at an appropriate time. Systems controlling an elevator or a vending machine are typical examples of reactive systems. A reactive system specification is said to be *realisable* if there is a model that can respond with appropriate timing to any requests [19, 20]. This suggests a connection between homeostasis in biology and realisability in reactive systems. In our framework, external signals can be seen as requests from the environment. We think we can also check homeostasis of gene networks in our framwork by using realisability checkers.

Another interesting subject is deriving an additional constraint φ' to force all behaviours of a gene regulatory network to satisfy some observed property ψ. This may facilitate finding meaningful biological facts such as the order of thresholds, regulation functions for multiple regulation or existence of hidden nodes, or suggest how we modify the current model of the biological system to adapt it to an observed property. To achieve this, we need a method that extracts useful information from automata that are generally huge, since they represent possible behaviours of networks.

The last issue, which is rather computer scientific, is studying an efficient simplification algorithm of Büchi automata to reduce the size of automata. It is desirable to have small automata when we intersect them in modular analysis method. Regarding this matter, a simplification algorithm is proposed by Somenzi and Bloem [21] which is implemented in GOAL [22]. Their algorithm is polynomial in the size of automata, so the amount of analysis time may increase in modular analysis. Their algorithm is rather strong for our purpose since the accepting language of automata is unchanged. If we are only interested in non-emptiness of automata, we can apply a more aggressive simplification in which the accepting language may be changed while preserving non-emptiness.

References

[1] D. Croft, G. O'Kelly, G. Wu et al. Reactome: a database of reactions, pathways and biological processes. *Nucleic Acids Research*, 39(Database-Issue):691–697, 2011.

[7] http://www.reactome.org/ReactomeGWT/entrypoint.html
[8] http://www.genecards.org/
[9] http://metacyc.org/
[10] http://www.ingenuity.com/science/knowledge_base.html
[11] http://www.genome.jp/kegg/

[2] M. Safran, I. Dalah, J. Alexander et al. GeneCards Version 3: the human gene integrator. *Database: The Journal of Biological Databases and Curation*, 2010(0):baq020+, 2010.

[3] P. D. Karp, M. Riley, S. M. Paley and A. Pellegrini-Toole. The MetaCyc Database. *Nucleic Acids Research*, 30(1):59–61, 2002.

[4] M. Kanehisa, S. Goto, Y. Sato, M. Furumichi and M. Tanabe. KEGG for integration and interpretation of large-scale molecular data sets. *Nucleic Acids Research*, 2011.

[5] S. Ito, N. Izumi, S. Hagihara and N. Yonezaki. Qualitative analysis of gene regulatory networks by satisfiability checking of linear temporal logic. In *Proceedings of the 10th IEEE International Conference on Bioinformatics & Bioengineering*, pages 232–237. 2010.

[6] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. 1990.

[7] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32:733–749, 1985.

[8] R. Thomas and M. Kauffman. Multistationarity, the basis of cell differentiation and memory. II. logical analysis of regulatory networks in terms of feedback circuits. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 11(1):180–195, 2001.

[9] A. Rabinovich. On translations of temporal logic of actions into monadic second-order logic. *Theor. Comput. Sci.*, 193:197–214, 1998.

[10] E. Snoussi and R. Thomas. Logical identification of all steady states: the concept of feedback loop characteristic states. *Bulletin of Mathematical Biology*, 55(5):973–991, 1993.

[11] M. J. Schurr, D. W. Martin, M. H. Mudd and V. Deretic. Gene cluster controlling conversion to alginate-overproducing phenotype in *Pseudomonas aeruginosa*: functional analysis in a heterologous host and role in the instability of mucoidy. *Journal of Bacteriology*, 176:3375–3382, 1994.

[12] J. Guespin and M. Kauffman. Positive feedback circuits and adaptive regulations in bacteria. *Acta biotheoretica*, 49(4):207–218, 2001.

[13] T. Aoshima, K. Sakuma and N. Yonezaki. An efficient verification procedure supporting evolution of reactive system specifications. In *Proceedings of the 4th International Workshop on Principles of Software Evolution*, IWPSE '01, pages 182–185. ACM, New York, NY, USA, 2001.

[14] M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Inf. Comput.*, 115:1–37, 1994.

[15] U. Alon. *An introduction To Systems Biology: Design Principles Of Biological Circuits*. Chapman and Hall/CRC, 2006.

[16] F. Fages, S. Soliman and N. Chabrier-Rivier. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry*, 4:64–73, 2004.

[17] G. Bernot, J. Comet, A. Richard and J. Guespin. Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *J. Theor. Biol.*, 229(3):339–347, 2004.

[18] H. de Jong, J. Geiselmann, G. Hernandez and M. Page. Genetic network analyzer: Qualitative simulation of genetic regulatory networks. *Bioinformatics*, 19(3):336–344, 2003.

[19] M. Abadi, L. Lamport and P. Wolper. Realizable and unrealizable specifications of reactive systems. In *ICALP '89: Proceedings of the 16th International Colloquium on Automata, Languages and Programming*, pages 1–17. Springer-Verlag, London, UK, 1989.

[20] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL '89: Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 179–190. ACM, New York, NY, USA, 1989.

[21] F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Proceedings of the 12th International Conference on Computer Aided Verification*, CAV '00, pages 248–263. Springer-Verlag, London, UK, 2000.

[22] Y.-K. Tsay, Y.-F. Chen, M.-H. Tsai, K.-N. Wu and W.-C. Chan. GOAL: a graphical tool for manipulating Büchi automata and temporal formulae. In *Proceedings of the 13th international conference on Tools and algorithms for the construction and analysis of systems*, TACAS'07, pages 466–471. Springer-Verlag, Berlin, Heidelberg, 2007.

[23] T. Aoshima. *On a verification System for Reactive System Specifications*. Ph.D. thesis, Tokyo Institute of Technology, 2003.

[24] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page and D. Schneider. Validation of qualitative models of genetic regulatory networks by model checking : Analysis of the nutritional stress response in *Escherichia coli*. *Bioinformatics*, 21(Suppl.1):i19–i28, 2005.

[25] U. Alon. Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8(6):450–461, 2007.