A Qualitative Framework for Analysing Homeostasis in Gene Networks

Sohei Ito[1], Shigeki Hagihara[2] and Naoki Yonezaki[2]
[1]*Department of Fisheries Distribution and Management, National Fisheries University, 2-7-1 Nagata-Honcho, Shimonoseki, Yamaguchi, Japan*
[2]*Department of Computer Science, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo, Japan*

Keywords: Gene Regulatory Network, Homeostasis, Temporal Logic, Realisability.

Abstract: Toward the system level understanding of the mechanisms contributing homeostasis in organisms, a computational framework to model a system and analyse its properties is indispensable. The purpose of this work is to provide a framework which enables testing and validating homeostatic properties on gene regulatory networks *in silico*. Based on a qualitative analysis framework for gene networks using temporal logic, we proposed a novel formulation of homeostasis by the notion of *realisability*. This formulation of homeostasis yields a qualitative method to analyse homeostasis of gene networks. In this formulation, homeostasis is captured by a response not for just an instantaneous stimulation such as dose-response relationships but for any input scenario e.g. oscillating or continuous inputs, which is difficult to be captured by quantitative models. Moreover, we can consider any number of inputs from an environment without difficulty. Such flexibility is a notable advantage of our framework. We demonstrate the usefulness of our framework in analysing a number of small but tricky networks.

1 INTRODUCTION

Qualitative methods in modelling and simulation of gene networks (de Jong et al., 2003; Fages et al., 2004; Bernot et al., 2004; Batt et al., 2005) are useful in that we do not need quantitative information since such information on kinetic parameters or molecular concentrations are usually absent, as we can see from current databases e.g. Reactome (Croft et al., 2011), GeneCards (Safran et al., 2010), Metacyc (Karp et al., 2002), Ingenuity® Knowledge Base and KEGG (Kanehisa et al., 2011). Ito et al. (Ito et al., 2010) proposed a method for analysing gene networks using linear temporal logic (LTL) (Emerson, 1990), in which, a gene network is modelled as an LTL formula which specifies its possible behaviours.

Their method for analysing gene networks is closely related with verification of *reactive system* specifications (Barringer et al., 1984; Pnueli and Rosner, 1989; Abadi et al., 1989; Wong-Toi and Dill, 1991; Mori and Yonezaki, 1993; Vanitha et al., 2000; Hagihara and Yonezaki, 2006). A reactive system is a system that responds to requests from an environment at an appropriate timing. Systems controlling an elevator or a vending machine are typical examples of reactive systems. Biological systems with external inputs or signals can be naturally considered as reactive systems.

Realisability (Pnueli and Rosner, 1989; Abadi et al., 1989) is a desirable property of reactive system specifications which requires systems to behave according to a specification in reaction to any input from an environment. In terms of biological systems, this property means that a system behaves with satisfying a certain property (e.g. keeping a concentration within some range) in reaction to any input from an environment (e.g. for any stress or stimulation). That is to say, the system is *homeostatic* with respect to the property.

Using this correspondence, we formulate the notion of homeostasis by realisability of reactive systems. Our formulation captures homeostasis of not only logical structure of gene networks but also properties of any dynamic behaviours of networks. For example, we can analyse in our framework whether a given network maintains oscillation over time in response to any input sequence. This formulation yields not only a novel and simple characterisation of homeostasis but also provides a method to automatically check homeostasis of a system using realisability checkers (Jobstmann and Bloem, 2006; Jobstmann et al., 2007; Filiot et al., 2009; Bloem et al., 2010).

Ito S., Hagihara S. and Yonezaki N..
A Qualitative Framework for Analysing Homeostasis in Gene Networks.
DOI: 10.5220/0004731400050016
In *Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms* (BIOINFORMATICS-2014), pages 5-16
ISBN: 978-989-758-012-3

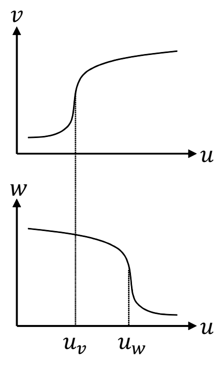

Figure 1: Regulation effect.

Based on this formulation we analyse some homeostatic properties of a number of small but tricky gene networks.

This paper is organised as follows. Section 2 and 3 reviews the qualitative analysis method using LTL (Ito et al., 2010) on that our work is grounded. In section 4, we introduce the notion of realisability and formulate homeostasis by this notion. Based on this formulation, we show some example networks and analyse homeostatic properties of them in section 5. Section 6 discusses some related works. The final section offers conclusions and 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 (Thomas and Kauffman, 2001). The sigmoid nature of gene regulation is shown in Figure 1, where gene u activates v and inhibits w. Each axis represents the concentration of products for each gene.

Some important landmark concentration values for u are 1) the basal level, 2) the level u_v at which u begins to affect v, and 3) the level u_w at which u begins to affect w. The values u_v and u_w are thresholds of gene u. 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). This switching view of genes leads us an abstract representation of network behaviours, *transition systems*.

Let us consider a simple network depicted in Figure 2, in which gene x activates gene y and receives the positive input from the environment.

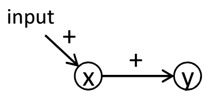

Figure 2: A simple example.

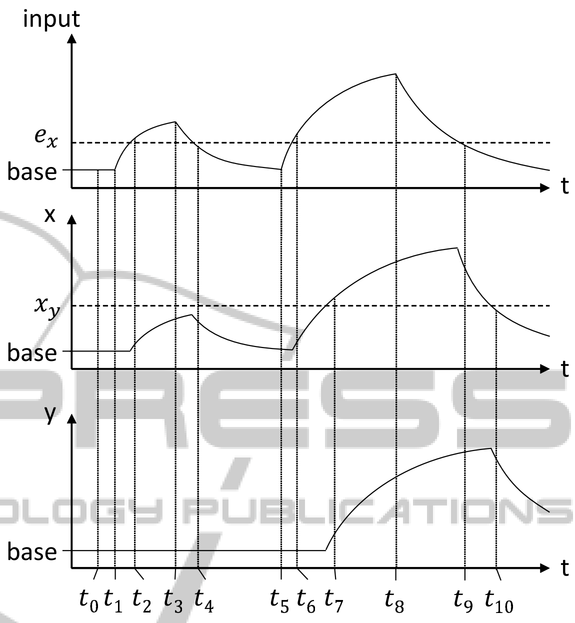

Figure 3: An example behaviour of the network in Figure 2.

We consider the behaviour depicted in Figure 3 in which e_x is a threshold level of the input to activate x and x_y a threshold of x to activate y. At time t_0, the input, gene x and gene y are at basal level. At time t_1, the input to x is coming and the level begins to increase. At time t_2, since the level of input to x exceeds e_x, gene x is being expressed. At time t_3, the input to x is stopped and the level begins to decrease. At time t_4, since the level of input to x falls below e_x, gene x stops being expressed; that is, the level of x is decreasing. At time t_5 the input to x is again coming and at time t_6 gene x is being expressed since the input level is over e_x. At time t_7, gene x is expressed over x_y, so gene y is being expressed. At time t_8 the input to x is stopped and at time t_9 falls below e_x so gene x stops being expressed. At time t_{10}, since gene x falls below x_y, gene y stops being expressed, after which gene x and y stay at their basal level.

This behaviour can be represented as a transition system in Figure 4. A transition system consists of *states* (represented as circles) and *transitions* (represented as arrows). A state represents a current status of the system, e.g. what genes are active or what are the expression levels of genes. A transition represents a change of states. To describe status of the system, we introduce logical propositions that repre-

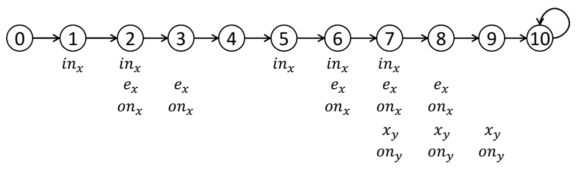

Figure 4: A Transition system corresponding to Figure 3.

sent 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 in_x, on_x, on_y, e_x, and x_y[1]. The meaning of each proposition is:

- on_x, on_y: whether gene x or y is active,
- in_x: whether the input to gene x is coming,
- x_y: whether the concentration of the products of gene x *exceeds* the threshold x_y.
- e_x: whether the level of the input to gene x *exceeds* the threshold e_x,

Propositions depicted below each state in Figure 4 shows the status of it. For example, state 2 represents the situation that the input to gene x is coming, the level of input is above the threshold e_x and gene x is ON. We can observe that state 0 represents the situation at time t_0, state 1 represents t_1, ... and state 10 represents t_{10}.

A single state transition can represent any length of time, since the actual duration of the transition (in real time) is immaterial[2] in this abstraction. Therefore, the difference between $t_2 - t_0$ and $t_7 - t_4$, the duration of the input to x in Figure 3, is not captured directly. We, however, can see that the latter duration is longer by comparing the propositions in state 1 to 3 and in state 5 to 9: the latter duration is sufficiently long for x to activate y.

Note that the real values of thresholds are also irrelevant. Propositions such as x_y merely represent the fact that the concentration of x is above the threshold at which x affects y.

In this abstraction, behaviours are identified with each other if they have the same transition system. This abstraction seems rather simple but preserves essential qualitative features of the dynamics (Snoussi and Thomas, 1993; Thomas and Kauffman, 2001).

Any behaviour of gene networks can be abstracted as transition systems. Sometimes, we need more propositions for expression levels of genes besides threshold values. We can introduce any number of them. We will see an example of such extra propositions in section 5, in which we prepare two propositions for one activation of a gene to capture a level of the activation.

[1]Symbols e_x and x_y are already used to represent the thresholds but we can clearly distinguish them from the context

[2]This property is called *speed independence* (Rabinovich, 1998)

3 MODELLING BEHAVIOURS OF GENE NETWORKS IN LTL

3.1 Linear Temporal Logic

First we introduce linear temporal logic.

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 $\sigma[i]$. Let AP be a set of propositions. A *time structure* is a sequence $\sigma \in (2^{AP})^\omega$ where 2^{AP} is the powerset of AP. The formulae in LTL are defined as follows.

- $p \in AP$ 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: $\perp \equiv p \wedge \neg p$ for some $p \in AP$, $\top \equiv \neg\perp$, $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$, $\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$, $F\phi \equiv \top U \phi$, $G\phi \equiv \neg F \neg\phi$, and $\phi W \psi \equiv (\phi U \psi) \vee G\phi$. We assume that $\wedge, \vee$ and U binds more strongly than $\rightarrow$ and unary connectives binds more strongly than binary ones.

Intuitively, $\neg\phi$ means 'ϕ is not true', $\phi \wedge \psi$ means 'both ϕ and ψ are true', $\phi \vee \psi$ means 'ϕ or ψ is true', and $\phi U \psi$ means 'ϕ continues to hold until ψ holds'. $\perp$ is a false proposition and $\top$ is a true proposition. $\phi \rightarrow \psi$ means 'if ϕ is true then ψ is true' and $\phi \leftrightarrow \psi$ means 'ϕ is true if and only if ψ is true'. $F\phi$ means 'ϕ holds at some future time', $G\phi$ means 'ϕ holds globally', $\phi W \psi$ 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.

Let σ be a time structure and ϕ be a formula. We write $\sigma \models \phi$ for ϕ is true in σ and we say σ *satisfies* ϕ. The satisfaction relation $\models$ is defined as follows.

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

where $\sigma^i = \sigma[i]\sigma[i+1]\ldots$, i.e. the i-th suffix of σ. An LTL formula ϕ is *satisfiable* if there exists a time structure σ such that $\sigma \models \phi$.

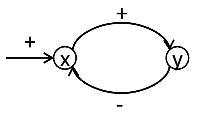

Figure 5: An example network.

3.2 Specifying Possible Behaviours of Gene Networks in LTL

Now we review the method proposed in (Ito et al., 2010) to model behaviours of a given network in linear temporal logic, using an example gene network depicted in Figure 5.

In this network gene x activates gene y and gene y inhibits gene x. Gene x has a positive environmental input. Let x_y be the threshold of gene x to activate gene y, y_x the threshold of gene y to inhibit gene x and e_x the threshold of the input to activate gene x. To specify possible behaviours of this network, we introduce the following propositions.

- on_x, on_y: whether gene x and y are ON respectively.
- x_y, y_x: whether gene x and y are expressed beyond the threshold x_y and y_x respectively.
- in_x: whether the input to x is ON.
- e_x: whether the positive input from the environment to x is beyond the threshold e_x.

The basic principles for characterising behaviours of a gene network are as follows:

- Genes are ON when their activators are expressed beyond some thresholds.
- Genes are OFF when their inhibitors are expressed beyond some thresholds.
- If genes are ON, the concentrations of their products increase.
- If genes are OFF, the concentrations of their products decrease.

We express these principles in LTL using the propositions introduced above.

Genes' Activation and Inactivation. Gene y is positively regulated by gene x. Thus gene y is ON if gene x is expressed beyond the threshold x_y, which is the threshold of gene x to activate gene y. This can be described as

$$G(x_y \leftrightarrow on_y)$$

in LTL. Intuitively this formula says gene y is ON if, and only if, gene x is expressed beyond x_y due to positive regulation effect of gene x toward gene y. As for gene x, it is negatively regulated by gene y and has positive input from the environment. A condition for activation and inactivation of such multi-regulated genes depends on a function which merges the multiple effects. We assume that gene x is ON if gene y is not expressed beyond y_x and the input from the environment to gene x is beyond e_x; that is, the negative effect of gene y is not operating and the positive effect of the input is operating. Then this can be described as

$$G(e_x \wedge \neg y_x \rightarrow on_x).$$

This formula says that if the input level is beyond e_x (i.e. proposition e_x is true) and gene y is not expressed beyond y_x (i.e. proposition y_x is false; $\neg y_x$ is true), then gene x is ON (i.e. proposition on_x is true).

For the inactivation of gene x, we have choices to specify the rule. Let us assume that gene x is OFF when the input from the environment is under e_x and gene y is expressed beyond y_x, that is, the activation to gene x is not operating and the inhibition to gene x is operating, in which case gene x will surely be OFF. This is specified as

$$G(\neg e_x \wedge y_x \rightarrow \neg on_x). \tag{1}$$

For another choice, let us assume that the negative effect from gene y overpowers the positive input from the environment. Then we write

$$G(y_x \rightarrow \neg on_x), \tag{2}$$

which says that if the inhibition from gene y is operating, gene x becomes OFF regardless of the environmental input to gene x. Yet another choice is

$$G(\neg e_x \vee y_x \rightarrow \neg on_x) \tag{3}$$

which says that gene x is OFF when the positive input is not effective *or* negative regulation from gene y is effective. For example, although gene y is not expressed beyond the threshold y_x (i.e. the negative effect of gene y is not effective), gene x is OFF if the positive effect of the input is not effective.

We also have several options for the activation of gene x. The choice depends on a situation (or assumption) of a network under consideration.

Changes of Expression Levels of Genes over Time. If gene x is ON, it begins to be expressed and in some future it will reach the threshold for gene y unless gene x becomes OFF. This can be described as

$$G(on_x \rightarrow F(x_y \vee \neg on_x)).$$

This formula means 'if gene x is ON, in some future the expression level of gene x will be beyond x_y, or otherwise gene x will become off'. This situation is

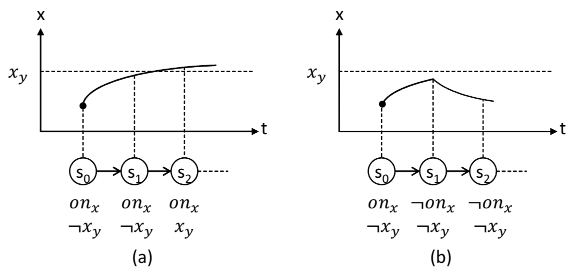
x
x_y
t
s_0 s_1 s_2
on_x on_x on_x
$\neg x_y$ $\neg x_y$ x_y
(a)
on_x $\neg on_x$ $\neg on_x$
$\neg x_y$ $\neg x_y$ $\neg x_y$
(b)

Figure 6: If gene x is ON, (a) the expression level of gene x is over x_y, or (b) gene x becomes OFF before gene x reaches x_y, where $s_0s_1s_2$c is a time structure.

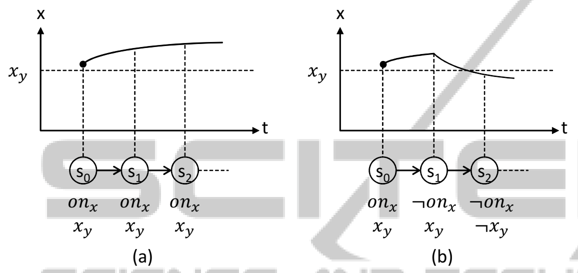

Figure 7: If gene x is ON and the current expression level is over x_y, (a) the expression level of gene x keeps over x_y, or (b) gene x becomes OFF and as a result the expression level may fall below x_y.

depicted in Figure 6. If gene x is ON and expressed beyond x_y, it keeps the level until gene x is OFF. This can be described as

$$G(on_x \wedge x_y \to x_y W \neg on_x).$$

This formula means 'if gene x is ON and the current expression level of gene x is over x_y, gene x keeps its level until gene x becomes OFF, or otherwise gene x keeps its level always'. This situation is depicted in Figure 7.

If gene x is OFF, its product decreases due to degradation. Thus if gene x is OFF and the current expression level of x is over x_y, it will fall below x_y in some future unless x becomes ON again. This can be specified as

$$G(\neg on_x \to F(\neg x_y \vee on_x)).$$

If the expression level of gene x is under x_y and x is OFF then it keeps the level (i.e. it does not increase and exceed x_y) until x is ON. This can be specified as

$$G(\neg on_x \wedge \neg x_y \to \neg x_y W on_x).$$

We have similar formulae for gene y and the input into gene x from the environment for increase and decrease of them.

The conjunction of above formulae (i.e. joining by $\wedge$ operator) is the specification of possible behaviours of the network. In other words, time structures which satisfy the formula are possible behaviours of the network.

This method for modelling behaviours of gene regulatory networks can be contrasted to usual quantitative methods like ordinary differential equation models. We qualitatively model gene regulatory networks by temporal logic formulae instead of quantitative analytical formulae. Note that we have several possible temporal logic specifications for a single network depending on order of threshold values, functions for multi-regulations and how we capture increase and decrease of expression of genes. Interested reader may wish to consult (Ito et al., 2010; Ito et al., 2013b) for detail.

4 REALISABILITY AND HOMEOSTASIS

In this section we discuss the connection between reactive systems and gene networks. Based on this connection, we formulate homeostasis of gene networks by realisability of reactive systems.

A *reactive system* is defined as a triple $\langle X, Y, r \rangle$, where X is a set of events caused by the environment, Y is a set of events caused by the system and $r : (2^X)^+ \to 2^Y$ is a reaction function. The set $(2^X)^+$ denotes the set of all finite sequences on subsets of X, that is to say, finite sequences on a set of environmental events. A reaction function determines how the system reacts to environmental input sequences. Reactive system is a natural formalisation of systems which appropriately respond to requests from the environment. Systems controlling vending machines, elevators, air traffic and nuclear power plants are examples of reactive systems. Gene networks which respond to inputs or stimulation from the environment such as glucose increase, change of temperature or blood pressure can also be considered as reactive systems.

A specification of a reactive system stipulates how it responds to inputs from the environment. For example, for a controller of an elevator system, a specification will be e.g. 'if the open button is pushed, the door opens' or 'if a call button of a certain floor is pushed, the lift will come to the floor'. It is important for a specification of a reactive system to satisfy *realisability* (Pnueli and Rosner, 1989; Abadi et al., 1989), which requires that there exists a reactive system such that for any environmental inputs of any timing, it produces system events (i.e. responds) so that it satisfies the specification.

To verify a reactive system specification, it should be described in a language with formal and rigorous semantics. Widespread research in specifying and developing reactive systems lead to the belief that temporal logic is the useful tool for reasoning them (Barringer, 1987; Pnueli and Rosner, 1989; Abadi et al., 1989; Vardi, 1995). LTL is known to be one of many other formal languages suitable for this purpose and several realisability checkers of LTL are available (Jobstmann and Bloem, 2006; Jobstmann et al., 2007; Filiot et al., 2009; Bloem et al., 2010).

Now we define the notion of realisability of LTL specifications. Let AP be a set of atomic propositions which is partitioned into X, a set of input propositions, and Y, a set of output propositions. X corresponds to input events and Y to output events. We denote a time structure σ on AP as $\langle x_0, y_0\rangle\langle x_1, y_1\rangle\ldots$ where $x_i \subseteq X$, $y_i \subseteq Y$ and $\sigma[i] = x_i \cup y_i$. Let ϕ be an LTL specification. We say $\langle X, Y, \phi\rangle$ is *realisable* if there exists a reactive system $RS = \langle X, Y, r\rangle$ such that

$$\forall \tilde{x}.behave_{RS}(\tilde{x}) \models \phi,$$

where $\tilde{x} \in (2^X)^\omega$ and $behave_{RS}(\tilde{x})$ is the infinite behaviour determined by RS, that is,

$$behave_{RS}(\tilde{x}) = \langle x_0, y_0\rangle\langle x_1, y_1\rangle\ldots,$$

where $\tilde{x} = x_0 x_1 \ldots$ and $y_i = r(x_0 \ldots x_i)$.

Intuitively ϕ is realisable if for any sequence of input events there exists a system which produces output events such that its behaviour satisfies ϕ.

Example 1. *Let* $X = \{push_{open}, push_{close}\}$ *and* $Y = \{door_{open}\}$. *The specification*

$$G(push_{open} \to F\,door_{open})$$

is realisable since there is a reactive system $\langle X, Y, r\rangle$ *with* $r(\bar{x}a) = \{door_{open}\}$ *where* $\bar{x}$ *is any finite sequence on* 2^X *and* $a \supseteq \{push_{open}\}$. *The specification*

$$G((push_{open} \to F\,door_{open}) \wedge (push_{close} \to \neg door_{open}))$$

is not realisable since for input sequence $\{push_{open}, push_{close}\}^\omega$ *there is no output sequence which satisfies the specification.*

Realisability can be interpreted as the ability of a system to maintain its internal condition irrespective of environmental inputs. In the context of gene networks, realisability can be naturally interpreted as *homeostasis*. For example, a network for controlling glucose level responds to an environmental inputs such as glucose increase or decrease in a manner to maintain its glucose level within a normal range. In the framework described in section 3, behaviour specifications of gene networks can be regarded as reactive system specifications. Based on this connection, we formulate homeostasis by realisability.

Let $\langle I, O, \phi\rangle$ be a behaviour specification of a gene network where I is the set of input propositions, O is the set of output propositions and ϕ is an LTL formula characterising possible behaviours of the network. Let ψ be a certain biological property of the network. A network property ψ is homeostatic in this network if for any input sequence $x_0 x_1 \ldots$ there exists a reaction function r such that the behaviour $\sigma = \langle x_0, r(x_0)\rangle\langle x_1, r(x_0, x_1)\rangle\ldots$ is a behaviour of the network (i.e. $\sigma \models \phi$) and σ also satisfies the property ψ (i.e. $\sigma \models \psi$). Thus we have the following simple definition of homeostasis:

Definition 1. *A property* ψ *is* homeostatic *with respect to a behaviour specification* $\langle I, O, \phi\rangle$ *if* $\phi \wedge \psi$ *is realisable.*

In this definition we consider responses of a system not only to initial instantaneous inputs such as dose-response relationship but also to any input sequences (e.g. inputs are oscillating or sustained), which is difficult to be captured by ordinary differential models. Moreover, we have any number of environmental inputs thus we can consider homeostasis against compositive environmental inputs.

Based on the method described in section 3 and this formulation, we can analyse homeostasis of gene networks using realisability checkers. In the next section, we demonstrate our method in analysing a number of small but tricky networks.

5 DEMONSTRATION: ANALYSIS OF HOMEOSTASIS FOR EXAMPLE NETWORKS

First we consider the network in Figure 5 again. The network in Figure 5 is expected to have a function that whenever gene x becomes ON, the expression of gene x will be suppressed afterward. This function maintains the expression level of gene x to its normal range (low level). Despite of the extreme situation that the input to gene x is always ON, the expression of gene x inevitably ceases due to the activation of gene y and its negative effect on gene x. Therefore this function is expected to be homeostatic. Now we formalise this verbal and informal reasoning with our framework. The property 'whenever gene x becomes ON, the expression of gene x will be suppressed afterward' is formally stated in LTL as:

$$G(on_x \to F(\neg on_x \wedge \neg x_y)). \quad (4)$$

This formula says that the property 'if on_x is true, it becomes false and gene x is suppressed below x_y in some future' always holds. We check whether this formula is realisable with respect to a behaviour specification introduced in section 3.2. There are 6 propositions $on_x, on_y, x_y, y_x, in_x$ and e_x for this network. The

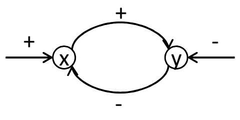

Figure 8: The network in Figure 5 with a negative input for gene *y*.

partition of input propositions and output propositions are straightforward, that is, in_x is the only input proposition since the environment only controls the input to gene x. Other propositions represent internal states of the network. Note that the environment cannot directly control the proposition e_x, which represents whether the level of the input exceeds e_x. To exceed the level e_x, the environment needs to give the input for a certain duration.

We had three options in the inactivation rule of gene x, i.e. formulae (1),(2) and (3). In all choices the property is realisable since even if in_x is always true, we need y_x being false for the activation of gene x due to the clause $G(e_x \wedge \neg y_x \rightarrow on_x)$. If in_x is always true, gene x will be expressed beyond x_y, and it induces y's expression. As a result, gene y can be expressed beyond y_x at which gene y inhibits gene x. Thus on_x may not be always true. If we replace the clause for the activation of gene x as $G(e_x \rightarrow on_x)$, which says if the input is effective gene x must be ON regardless of the negative effect of gene y, then the property is not realisable.

For realisability checking, we used Lily[3] (Jobstmann and Bloem, 2006) which is a tool for checking realisability of LTL specifications. To use Lily, we specify input propositions, output propositions and an LTL formula. The result of checking (Yes or No) is output to command-line and if it is YES, it also outputs a state diagram.

Now we assume gene y accepts negative input from the environment (Figure 8). We have the extra input proposition in_y and output proposition e_y. We describe the activation rule for gene y as follows in which gene y can be OFF by the negative input from the environment:

$$G(\neg e_y \wedge x_y \rightarrow on_y),$$
$$G(e_y \rightarrow \neg on_y).$$

Is the property (4) homeostatic with respect to this behaviour specification? The realisability checker answers 'No'. The reason is that if the input for gene y is always ON, gene y cannot be ON, therefore the negative effect from gene y to gene x cannot be effective. In this input scenario gene x cannot become OFF after gene x becomes ON.

[3]http://www.iaik.tugraz.at/content/research/design_verification/lily/

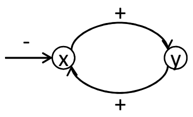

Figure 9: A bistable switch.

Now we consider the next example depicted in Figure 9. In this network we provide two thresholds y_x^0 and y_x^1 for gene y. The threshold y_x^0 is the level enough to activate gene x when the negative input from the environment is not effective. The threshold y_x^1 is the level enough to activate gene x regardless of negative effect from the environment, that is, y_x^1 is the threshold beyond which gene y overpowers the environmental input. The behaviour specification for this network will be somewhat complicated. First, we describe the fact that the threshold y_x^1 is greater than y_x^0, which is simply described as follows:

$$G(y_x^1 \rightarrow y_x^0),$$

which says 'if gene y is expressed beyond y_x^1, it is also beyond y_x^0 (since $y_x^1 > y_x^0$)'. Note that the proposition y_x^1 means 'gene y is expressed *beyond* the threshold y_x^1'.

The activation rules and inactivation rules for gene x are as follows:

$$G(\neg y_x^0 \rightarrow \neg on_x), \quad (5)$$
$$G(e_x \wedge y_x^0 \wedge \neg y_x^1 \rightarrow \neg on_x), \quad (6)$$
$$G(\neg e_x \wedge y_x^0 \rightarrow on_x), \quad (7)$$
$$G(y_x^1 \rightarrow on_x). \quad (8)$$

Formula (5) says that if gene y is under y_x^0, gene x is OFF regardless of the environmental input. Formula (6) says that if gene y is in between y_x^0 and y_x^1 but the negative input is effective, gene x is OFF. Formula (7) says that gene x is ON when negative input is not effective and gene y is expressed over y_x^0. Formula (8) says that gene x is ON when gene y is just expressed over y_x^1.

The activation rule for gene y is simple:

$$G(x_y \leftrightarrow on_y)$$

The change of the expression level of gene y when it is ON are described as follows:

$$G(on_y \rightarrow F(y_x^0 \vee \neg on_y)), \quad (9)$$
$$G(on_y \wedge y_x^0 \rightarrow y_x^0 W \neg on_y), \quad (10)$$
$$G(on_y \wedge y_x^1 \rightarrow y_x^1 W \neg on_y). \quad (11)$$

Formula (9) says that if gene y is ON, it will reach the first threshold y_x^0 or otherwise it will become OFF. Formula (10) says that if gene y is ON and the current level is over the first threshold y_x^0, it will keep over y_x^0

(this means it can be expressed beyond y_x^1), or otherwise gene y becomes OFF. Formula (11) says that if gene y is ON and the current level is over the highest threshold y_x^1, it keeps y_x^1 or otherwise it will become OFF.

We have similar formulae for the change of the expression level of gene y when it is OFF.

$$G(\neg on_y \to F(\neg y_x^1 \vee on_y)),$$
$$G(\neg on_y \wedge \neg y_x^1 \to \neg y_x^1 W on_y),$$
$$G(\neg on_y \wedge \neg y_x^0 \to \neg y_x^0 W on_y).$$

For the change of the expression level of gene x and the environmental input we have similar formulae except they have only one threshold.

We check the bistability of the expression of gene x, that is to say, if gene x can always be ON or always be OFF. These properties are described as follows:

$$G on_x, \tag{12}$$
$$G \neg on_x. \tag{13}$$

By using Lily, we checked that both properties are really homeostatic. Informal reasoning for the first property (12) is as follows. Suppose that the input sequence such that the negative input to x is always effective, which is the best choice for the environment to inactivate gene x. The system's response to satisfy the bistability is to start at a state in which both gene x and y are ON and gene x and gene y are expressed beyond x_y and y_x^1, respectively. Since gene y is expressed beyond y_x^1, gene x can continue to be ON regardless of negative input to x. The expression of gene y is supported by the positive effect from gene x. For the second property (13), we assume that the negative input is always ineffective. The system's response is simply to start a state that both gene x and y is OFF and gene x and y are expressed below x_y and y_x^0, respectively. For x to be ON, we need y_x^0 being true but the system can control gene y to be OFF since gene x is OFF.

We expect both gene x and y are either ON or OFF simultaneously. This can be checked by the following properties:

$$G on_x \wedge G on_y, \tag{14}$$
$$G \neg on_x \wedge G \neg on_y. \tag{15}$$

Both properties are really homeostatic. Therefore gene x and gene y are 'interlocked' in a sense.

We further investigate this 'interlocking' property. Can gene x (and gene y) always be ON by its own? That is to say, are the following properties homeostatic?

$$G on_x \wedge G \neg on_y, \tag{16}$$
$$G \neg on_x \wedge G on_y. \tag{17}$$

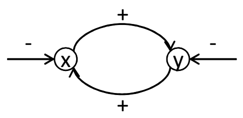

Figure 10: A bistable switch with a negative input to gene y.

The answers are 'No' for both properties. To keep gene x being ON gene x must be expressed beyond y_x and this prevents gene y to be always OFF. Thus the property (16) is not homeostatic. This property is even not satisfiable. That is to say, there is no input sequence to satisfy the property (16). Conversely, to keep gene y being ON gene x must be expressed beyond x_y and this prevents gene x to be always OFF. Thus the property (17) is not homeostatic and not satisfiable too.

Interestingly, provided gene y accepts a negative input from the environment (Figure 10), the properties (12) and (13) are still homeostatic. Even if both negative inputs are always effective, each gene can be expressed thanks to the positive effect from the other gene. We confirmed the properties (12) and (13) are really homeostatic with respect to the following behaviour specification in which we have two thresholds for gene x (only activation rules for gene y are shown):

$$G(\neg x_y^0 \to \neg on_y),$$
$$G(e_y \wedge x_y^0 \wedge \neg x_y^1 \to \neg on_y),$$
$$G(\neg e_y \wedge x_y^0 \to on_y),$$
$$G(x_y^1 \to on_y).$$

Moreover, the properties (14) and (15) are still homeostatic. The properties (16) and (17) are also not homeostatic but are satisfiable in contrast to the previous case since if the environment appropriately controls the inputs, gene y can be ON and OFF alternately but keeps the expression level beyond y_x^0 so that gene x can be ON indefinitely.

The last examples are anti-stress networks (Zhang and Andersen, 2007) depicted in Figure 11. The networks in Figure 11 control the upper right objects to keep them within the tolerable ranges. Though these networks are schematic, we are just interested in the control mechanisms which contribute homeostasis against environmental stresses. Let us consider the network of Figure 11 (c). If the amount of O_2 becomes low, the network tries to recover the level of O_2. The property can be described as follows:

$$G(\neg o_2 \to F o_2)$$

In this formula we interpret proposition o_2 as 'the amount of O_2 is within the tolerable range' so $\neg o_2$ means it deviates the tolerable range. The behaviour

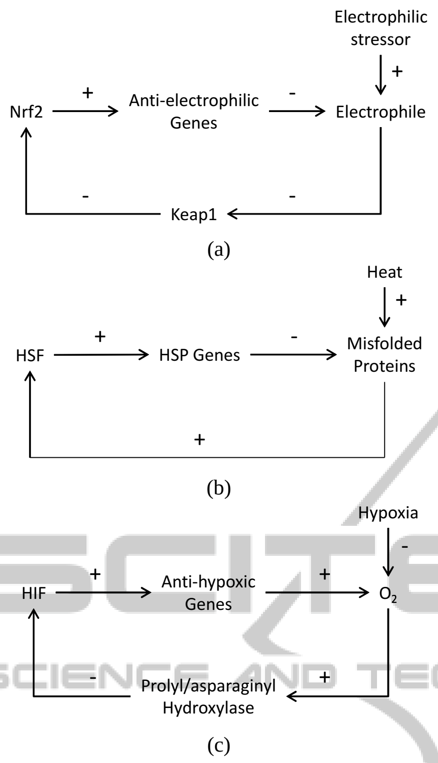

Figure 11: Schematic representations of anti-stress gene regulatory networks that meditate (a) electrophilic stress response, (b) heat shock response and (c) hypoxic response.

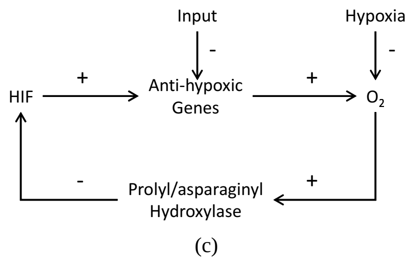

Figure 12: The network (c) with hypothetic negative inputs.

specification of the network is obtained as usual[4]. We checked the property is really homeostatic.

Now we have a question: is this homeostatic function broken by the assumption that anti-hypoxic genes receive environmental negative input? (Fig. 12) To check this hypothesis, we modified the behaviour specification in which anti-hypoxic genes receive a negative input from the environment. The activation rule for anti-hypoxic genes is modified considering the negative input. The result of realisability checking was 'No'. This analysis indicates that the homeostasis of this network may be broken by some environmental factor which hinders the operations of anti-hypoxic genes. Such analysis is difficult by observing dose-response relationship based on ordinary differential models.

The homeostatic properties for other two networks are similarly checked. Basic network topologies are almost the same and the modification of network specifications are minor.

[4]We have '*on*' propositions for each node and threshold propositions for each edge.

6 RELATED WORK

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

BIOCHAM (Fages et al., 2004) is a language and programming environment for modelling and simulating biochemical systems, and checking their temporal properties. Reactions are written as rules like `A+B=>C`, and simulations are performed by replacing objects on the left-hand side with those on the right-hand side. Since there are many possible rules that can be applied in each state, there are many possible successor states for each state depending on the rule applied. After simulation, we have a non-deterministic transition graph whose nodes are possible states and edges are state transitions. The set of possible behaviours of the simulation over-approximates the set of all behaviours of the system depending on the kinetic parameters. A biological property is written in computation tree logic (CTL), a type of branching time logic, and checked in the resulting transition graph by the model checking technique (Clarke et al., 1999). In BIOCHAM, presence or absence of objects is the only matter considered. How we represent the interaction between biological systems and environments in BIOCHAM is not presented, so it is unclear how we capture homeostasis in BIOCHAM.

SMBioNet (Bernot et al., 2004) is a tool for formally analysing temporal properties of gene regulatory networks. In SMBioNet, genes have concentration thresholds for activation or inhibition of each of their regulating genes. A configuration of systems is represented as a vector of expression values, which are segmented by threshold. For example, if a gene has two thresholds, then it has three levels – 0, 1, and 2. Behaviours of a network are captured as a transition system on the vectors of values for genes in the network. Temporal evolution of a system is described by a transition function on the vectors. Temporal properties are described in CTL, and verification of them is conducted by model checking on the resulting transition systems. Since the models of SMBioNet are deterministic, it is not clear how to consider any sequence of environmental inputs in SMBioNet.

GNA (de Jong et al., 2003) is a computational tool for the modelling and simulation of gene regulatory networks. GNA achieves simulation using piecewise linear differential equation models and generates state transition systems that represent possible behaviours of networks. The qualitative dynamics of a system are completely determined by inequality constraints defining the ordering between thresholds and stable equilibria of the system. Network properties of interest are checked automatically using model checking (Batt et al., 2005). Since the models of GNA are based on piecewise linear differential equation, interactions between biological systems and environments over time cannot be directly captured. How we take this essential aspect into consideration by GNA is not clear.

As we mentioned, our work is based on the method proposed by Ito et al. (Ito et al., 2010). For a behaviour specification ϕ and a biological property ψ, they check *satisfiability* of the formula $\phi \land \psi$ to know whether there is a behaviour which satisfies the property, or check unsatisfiability of $\phi \land \neg\psi$ to investigate whether all behaviours satisfy the property. They did not distinguish input propositions and output propositions. This means they only considered whether there exist an input sequence to which a network can respond without violating the biological property ψ.

7 CONCLUSIONS

In this paper we formulated the notion of homeostasis in gene regulatory networks by realisability in reactive systems. This formulation allows the automatic analysis of homeostasis of gene regulatory networks using realisability checkers. We analysed several networks with our method. In the analyses we can easily 'tweak' a network (such as appending extra-inputs from the environment) and observed whether the homeostatic properties can be maintained. Such flexibility in analysing networks is an advantage of our framework in the situation that we do not have the definite network topologies. To test several hypothetic networks, our method is more suitable than quantitative approaches using ordinary differential equation models.

There are several interesting future directions based on this work. First is to find more interesting applications in real biological examples. In association with this topic, we are interested in 'conditional' homeostasis which means that under certain constraints on input sequences, a property is homeostatic. This can be easily formulated as follows. Let I and O be the input and output propositions respectively. Let $\langle I, O, \phi \rangle$ be a behavioural specification and ψ be a property. Let σ be an assumption about input sequences e.g. 'inputs to gene *x* and gene *y* come infinitely often but not simultaneously'. Then the property ψ is *conditionally homeostatic* with respect to $\langle I, O, \phi \rangle$ under a condition σ if $\langle I, O, \sigma \rightarrow \phi \land \psi \rangle$ is realisable. The motivation of this definition is that in more realistic situation it is too strong to require a system to respond to *any* input sequence.

The next topic is to develop a method to suggest how we modify the model of a network when an expected or observed property is not homeostatic in a model. This problem is closely related to refinement of reactive system specifications (Aoshima et al., 2001; Hagihara et al., 2009). We hope the techniques developed so far for verification of reactive systems can be imported to analysis of gene networks.

Another important future work is to develop a method to overcome high complexity in checking realisability of LTL formulae. The complexity of realisability checking is doubly exponential in the length of the given specification (Pnueli and Rosner, 1989). Thus it is intractable to directly apply our method to large networks. To circumvent this theoretical limitation we are interested in some approximate analysis method (Ito et al., 2013b) or modular analysis method (Ito et al., 2013a) in which a network is divided into several subnetworks and analyse them individually.

The last topic is to extend our method with some quantitative temporal logic (e.g. probability or real time) (Tomita et al., 2011; Tomita et al., 2012) to enable quantitative analysis.

REFERENCES

Abadi, M., Lamport, L., and Wolper, P. (1989). Realizable and unrealizable specifications of reactive systems. In *ICALP '89: Proceedings of the 16th International Colloquium on Automata, Languages and Programming*, volume 372 of *LNCS*, pages 1–17, London, UK. Springer-Verlag.

Aoshima, T., Sakuma, K., and Yonezaki, N. (2001). 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, New York, NY, USA. ACM.

Barringer, H. (1987). Up and down the temporal way. *The Computer Journal*, 30(2):134–148.

Barringer, H., Kuiper, R., and Pnueli, A. (1984). Now you may compose temporal logic specifications. In *Proceedings of the sixteenth annual ACM symposium on*

Theory of computing, STOC '84, pages 51–63, New York, NY, USA. ACM.

Batt, G., Ropers, D., de Jong, H., Geiselmann, J., Mateescu, R., Page, M., and Schneider, D. (2005). 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.

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

Bloem, R., Cimatti, A., Greimel, K., Hofferek, G., Könighofer, R., Roveri, M., Schuppan, V., and Seeber, R. (2010). RATSY – a new requirements analysis tool with synthesis. In *Proceedings of the 22nd international conference on Computer Aided Verification*, volume 6174 of *LNCS*, pages 425–429, Berlin, Heidelberg. Springer-Verlag.

Clarke, E., Grumberg, O., and Peled, D. (1999). *Model Checking*. MIT Press.

Croft, D., O'Kelly, G., Wu, G., Haw, R., Gillespie, M., Matthews, L., Caudy, M., Garapati, P., Gopinath, G., Jassal, B., Jupe, S., Kalatskaya, I., Mahajan, S., May, B., Ndegwa, N., Schmidt, E., Shamovsky, V., Yung, C., Birney, E., Hermjakob, H., D'Eustachio, P., and Stein, L. (2011). Reactome: a database of reactions, pathways and biological processes. *Nucleic Acids Research*, 39(Database-Issue):691–697.

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

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

Fages, F., Soliman, S., and Chabrier-Rivier, N. (2004). Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *J. Biol. Phys. Chem.*, 4:64–73.

Filiot, E., Jin, N., and Raskin, J.-F. (2009). An antichain algorithm for ltl realizability. In *Proceedings of the 21st International Conference on Computer Aided Verification*, volume 5126 of *LNCS*, pages 263–277, Berlin, Heidelberg. Springer-Verlag.

Hagihara, S., Kitamura, Y., Shimakawa, M., and Yonezaki, N. (2009). Extracting environmental constraints to make reactive system specifications realizable. In *Proceedings of the 2009 16th Asia-Pacific Software Engineering Conference*, APSEC '09, pages 61–68, Washington, DC, USA. IEEE Computer Society.

Hagihara, S. and Yonezaki, N. (2006). Completeness of verification methods for approaching to realizable reactive specifications. In *Completeness of Verification methods for Approaching to Realizable Reactive Specifications*, volume 348, pages 242 – 257.

Ito, S., Ichinose, T., Shimakawa, M., Izumi, N., Hagihara, S., and Yonezaki, N. (2013a). Modular analysis of gene networks by linear temporal logic. *J. Integrative Bioinformatics*, 10(2).

Ito, S., Ichinose, T., Shimakawa, M., Izumi, N., Hagihara, S., and Yonezaki, N. (2013b). Qualitative analysis of gene regulatory networks using network motifs. In *Proceedings of the 4th International Conference on Bioinformatics Models, Methods and Algorithms (BIOINFORMATICS2013)*, pages 15–24.

Ito, S., Izumi, N., Hagihara, S., and Yonezaki, N. (2010). 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.

Jobstmann, B. and Bloem, R. (2006). Optimizations for ltl synthesis. In *Proceedings of the Formal Methods in Computer Aided Design*, FMCAD '06, pages 117–124, Washington, DC, USA. IEEE Computer Society.

Jobstmann, B., Galler, S., Weiglhofer, M., and Bloem, R. (2007). Anzu: a tool for property synthesis. In *Proceedings of the 19th international conference on Computer aided verification*, volume 4590 of *LNCS*, pages 258–262, Berlin, Heidelberg. Springer-Verlag.

Kanehisa, M., Goto, S., Sato, Y., Furumichi, M., and Tanabe, M. (2011). KEGG for integration and interpretation of large-scale molecular data sets. *Nucleic Acids Res.*

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

Mori, R. and Yonezaki, N. (1993). Several realizability concepts in reactive objects. In *Information Modeling and Knowledge Bases IV*, pages 407–424.

Pnueli, A. and Rosner, R. (1989). 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, New York, NY, USA. ACM.

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

Safran, M., Dalah, I., Alexander, J., Rosen, N., Iny Stein, T., Shmoish, M., Nativ, N., Bahir, I., Doniger, T., Krug, H., Sirota-Madi, A., Olender, T., Golan, Y., Stelzer, G., Harel, A., and Lancet, D. (2010). GeneCards Version 3: the human gene integrator. *Database*, 2010(0):baq020+.

Snoussi, E. and Thomas, R. (1993). Logical identification of all steady states: the concept of feedback loop characteristic states. *B. MATH. BIOL.*, 55(5):973–991.

Thomas, R. and Kauffman, M. (2001). Multistationarity, the basis of cell differentiation and memory. II. logical analysis of regulatory networks in terms of feedback circuits. *CHAOS*, 11(1):180–195.

Tomita, T., Hagihara, S., and Yonezaki, N. (2011). A probabilistic temporal logic with frequency operators and its model checking. In *INFINITY*, pages 79–93. EPTCS.

Tomita, T., Hiura, S., Hagihara, S., and Yonezaki, N. (2012). A temporal logic with mean-payoff constraints. In *Proceedings of the 14th international conference on Formal Engineering Methods: formal methods and software engineering*, volume 7635 of *LNCS*, pages 249–265, Berlin, Heidelberg. Springer-Verlag.

Vanitha, V., Yamashita, K., Fukuzawa, K., and Yonezaki., N. (2000). A method for structuralisation of evolutional specifications of reactive systems. In *ICSE 2000, The Third International Workshop on Intelligent Software Engineering (WISE3)*, pages 30 – 38.

Vardi, M. Y. (1995). An automata-theoretic approach to fair realizability and synthesis. In *Proceedings of the 7th International Conference on Computer Aided Verification*, volume 939 of *LNCS*, pages 267–278, Berlin, Heidelberg. Springer-Verlag.

Wong-Toi, H. and Dill, D. L. (1991). Synthesizing processes and schedulers from temporal specifications. In *Proceedings of the 2nd International Workshop on Computer Aided Verification*, volume 531 of *LNCS*, pages 272–281, London, UK, UK. Springer-Verlag.

Zhang, Q. and Andersen, M. E. (2007). Dose response relationship in anti-stress gene regulatory networks. *PLoS Comput. Biol.*, 3(3).