A genetically modified Hoare logic that identifies the parameters of a gene network

G. Bernot[1], J.-P. Comet[1], O. Roux[2]

[1] University Nice Sophia Antipolis
I3S laboratory, UMR CNRS 7271
CS 40121, 06903 Sophia Antipolis CEDEX, France

[2] IRCCyN, UMR CNRS 6597, BP 92101
1 rue de la Noë, 44321 Nantes Cedex 3, France

The main difficulty when modelling gene networks is the identification of the parameters that govern the dynamics. Here we present a new approach based on Hoare logic and weakest preconditions (a la Dijkstra) that generates constraints on the parameter values: Once proper specifications are extracted from biological traces, they play a role similar to programs in the classical Hoare logic. We firstly remind the discrete modelling for genetic networks defined by René Thomas. Then, we define the Hoare/Dijkstra method extended to gene networks, that extracts the weakest precondition on parameter values.

1 Thomas' gene regulatory networks with multiplexes

Our formal framework [KCRB09] is based on the discrete approach of René Thomas [TK01]: A gene network is a labelled directed graph (left part of Figure 1) in which vertices are either *variables* (within circles) or *multiplexes* (within rectangles). Variables abstract genes or their products, and multiplexes contain propositional formulas that encode situations in which a group of variables (inputs of multiplexes) influence the evolution of some variables (outputs of multiplexes). In the figure the multiplex μ_2 expresses that the variable x can help the activation of the variable y when it is at least equal to 1. In general multiplexes can represent combined biological phenomena, one of the simplest being the formation of complexes (in which case the formula would contain a conjunction). In the figure, μ_1 reflects an auto-activation of x at level 2 which is controled by μ_3. Because μ_3 contains a negation, μ_1 is *inhibited* by y.

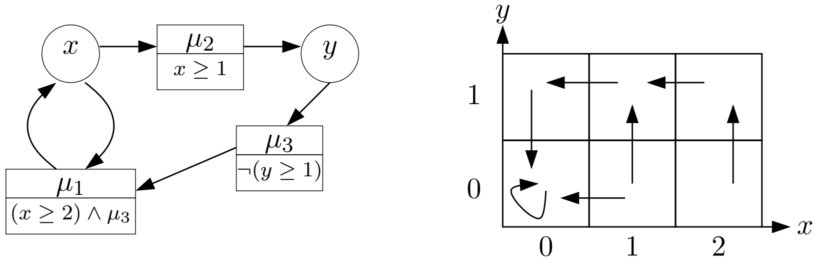

Figure 1: **(Left)** Discrete gene network with variables x and y, multiplexes μ_1, μ_2 and μ_3 with associated formulas $\varphi_{\mu_1} \equiv ((x \geqslant 2) \wedge \mu_3)$, $\varphi_{\mu_2} \equiv (x \geqslant 1)$ and $\varphi_{\mu_3} \equiv \neg(y \geqslant 1)$. **(Right)** Its state graph obtained when choosing parameters $K_{x,\varnothing} = 0$, $K_{x,\{\mu_1\}} = 2$, $K_{y,\varnothing} = 0$, and $K_{y,\{\mu_2\}} = 1$.

As shown in the right part of Figure 1, this gives rise to 6 qualitative regions in the phase space, which we call (discrete) states. A *state* is an assignment of integer values to the variables. Such an assignment allows a natural evaluation of any formula within a multiplex: By replacing variables by their values we get a propositional formula whose atoms are the results of the integer inequalities. Then, we say that a multiplex m, predecessor of a variable v in the graph, is a *resource* of v iff its substituted formula is true: at the state $(x = 2, y = 1)$, μ_2 is the only resource of y whereas φ_{μ_1} is false and consequently, the set of resources of x is empty.

At a given state η, each variable v evolves in the direction of a specific level that only depends on the set of resources of v. This level is the integer value of a parameter $K_{v,\rho(\eta,v)}$, where $\rho(\eta, v)$

is the set of resources of v at η. Hence, at state η, v can increase if $\eta(v) < K_{v,\rho(\eta,v)}$, it can decrease if $\eta(v) > K_{v,\rho(\eta,v)}$, and it is stable if $\eta(v) = K_{v,\rho(\eta,v)}$. In the Thomas' method, the variables evolve asynchronously by unit steps toward their respective $K_{\dots}$. The dynamics of a gene network is then described by an asynchronous state graph (right part of Figure 1).

2 Hoare triples for gene networks

An *assertion* is a formula whose terms are sums or subtractions between integers, variables or parameters $K_{\dots}$ of the gene network, predicates are equalities or inequalities, and connectives are the usual ones of first order logic. A *Hoare triple* is an expression of the form "$\{P\}\, p\, \{Q\}$" where P and Q are assertions (pre- and post-conditions) and p is a *trace specification.*

Trace specifications are indeed the key concept to formalize the observations of a biologist during experiments. They are inductively defined as follows:

- For each variable v of the gene network, the expressions "$v+$", "$v-$" and "$v := n$" (where $n \in \mathbb{N}$) are trace specifications (increase, decrease or assignment of variable value).
- If e is an assertion then "$assert(e)$" is a trace specification.
- If p_1 and p_2 are trace specifications then so is $(p_1; p_2)$ (sequential composition).
- If p_1 and p_2 are trace specifications and if e is an assertion, then so is $(if\ e\ then\ p_1\ else\ p_2)$.
- If p is a trace specification and if e and I are assertions, then $(while\ e\ with\ I\ do\ p)$ is also a trace specification. The assertion I is called the invariant of the *while* loop.
- If p_1 and p_2 are trace specifications then so are $\forall(p_1, p_2)$ and $\exists(p_1, p_2)$ (quantifiers).

Conventionnaly, we call *the empty trace* $\varepsilon = assert(true)$.

For lack of space we do not define here the formal semantics of trace specifications [BCK+15]. Intuitivelly, "$v+$" (resp. "$v-$", "$v := n$") means that the expression level of variable v has been observed as increasing by one unit (resp. decreasing by one unit, or set to a particular value n by the experimental protocole). "$assert(e)$" expresses a property observed on the current state without change of state. The sequential composition concatenates two specifications whereas "if" chooses between two specifications according to the assertion e. The loop invariant I, as in classical Hoare logic, facilitates proofs through *while* loops. Finally, the quantifiers $\forall$ and $\exists$ group together several specifications.

3 A Hoare logic for gene networks

We define our Hoare logic by giving the rule for each instruction of a trace specification. First, let us introduce a few conventional assertions.

If ω is a subset of the set of predecessors of a variable v in the network, the assertion Φ_v^ω characterizes the states such that ω is the set of resources of v:

$$\Phi_v^\omega \equiv \Big(\bigwedge_{m \in \omega} \varphi_m\Big) \wedge \Big(\bigwedge_{m \notin \omega,\ m\ predecessor\ of\ v} \neg\varphi_m\Big)$$

Then, the formula Φ_v^+ and Φ_v^- characterize the states such that v can increase / decrease:

$$\Phi_v^+ \equiv \bigwedge_{\omega \subset \{predecessors\ of\ v\}} (\Phi_v^\omega \Rightarrow K_{v,\omega} > v) \qquad \Phi_v^- \equiv \bigwedge_{\omega \subset \{predecessors\ of\ v\}} (\Phi_v^\omega \Rightarrow K_{v,\omega} < v)$$

Our genetically modified Hoare logic is defined by the following inference rules, where v is a variable of the gene network.

Rules encoding Thomas' discrete dynamics:

$$\frac{}{\{\ \Phi_v^+\ \wedge\ Q[v{\leftarrow}v{+}1]\ \}\ v{+}\ \{Q\}}\text{Incrementation} \qquad \frac{}{\{\ \Phi_v^-\ \wedge\ Q[v{\leftarrow}v{-}1]\ \}\ v{-}\ \{Q\}}\text{Decrementation}$$

Rules for quantifiers:

$$\frac{\{P_1\}\ p_1\ \{Q\} \qquad \{P_2\}\ p_2\ \{Q\}}{\{P_1{\wedge}P_2\}\ \forall(p_1{,}p_2)\ \{Q\}}\text{Universal} \qquad \frac{\{P_1\}\ p_1\ \{Q\} \qquad \{P_2\}\ p_2\ \{Q\}}{\{P_1{\vee}P_2\}\ \exists(p_1{,}p_2)\ \{Q\}}\text{Existential}$$

Other rules, directly inspired by Hoare Logic:

$$\frac{}{\{\ \Phi\ \wedge\ Q\ \}\ assert(\Phi)\ \{\ Q\ \}}\text{Assert} \qquad \frac{}{\{Q[v{\leftarrow}k]\}\ v{:=}k\ \{Q\}}\text{Assignment}$$

$$\frac{\{P_1\}\ p_1\ \{P_2\} \qquad \{P_2\}\ p_2\ \{Q\}}{\{P_1\}\ p_1{;}p_2\ \{Q\}}\text{Sequential} \qquad \frac{\{P_1\}\ p_1\ \{Q\} \qquad \{P_2\}\ p_2\ \{Q\}}{\{(e{\wedge}P_1){\vee}(\neg e{\wedge}P_2)\}\ if\ e\ then\ p_1\ else\ p_2\ \{Q\}}\text{Conditional}$$

$$\frac{\{e{\wedge}I\}\ p\ \{I\} \qquad \neg e{\wedge}I{\Rightarrow}Q}{\{I\}\ while\ e\ with\ I\ do\ p\ \{Q\}}\text{Iteration} \qquad \frac{P\ \Rightarrow\ Q}{\{P\}\ \varepsilon\ \{Q\}}\text{Empty trace}$$

We have proved that this modified Hoare logic is correct, and complete assuming a proper choice of the loop invariants [BCK+15]. More precisely, the classical *backward strategy* of Dijkstra (where the Empty trace rule is never applied) computes the weakest precondition P_0 such that $\{P_0\}\ p\ \{Q\}$. Similarly to classical Hoare logic which reflects a partial correctness of imperative programs, the previous definition does not imply termination of *while* loops.

4 Example

In [Mt02] Uri Alon and co-workers have studied the most common *in vivo* patterns involving three genes. Among them, the *incoherent feedforward loop of type 1* is composed by a transcription factor a that activates a second transcription factor c, and a is an activator of b whereas c is an inhibitor of b (Figure 2). Uri Alon and many biologists consider that if a, b and c are

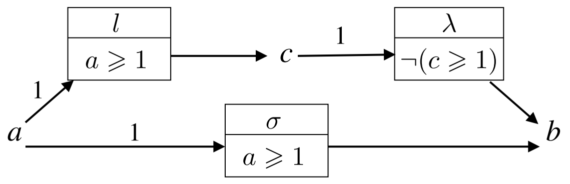

Figure 2: Boolean variables: $\{a, b, c\}$. Multiplexes: $\{l, \lambda, \sigma\}$ with $\phi_l{\equiv}(a \geqslant 1)$, $\phi_\lambda{\equiv}(\neg(c \geqslant 1))$, $\phi_\sigma{\equiv}(a \geqslant 1)$. Unknown parameters: $K_{a,\emptyset}$, $K_{c,\varnothing}$, $K_{c,\{l\}}$, $K_{b,\varnothing}$, $K_{b,\{\sigma\}}$, $K_{b,\{\lambda\}}$ and $K_{b,\{\sigma,\lambda\}}$.

equal to 0, the function of this feedforward loop is to ensure a *transitory activity* of b that signals when a has *switched* from 0 to 1: The idea is that a activates the productions of b and c, and then c stops the production of b. This is specified by the Hoare triple $\{P\}\ p\ \{Q_0\}$ where $P \equiv (a = 1\ \wedge\ b = 0\ \wedge\ c = 0)$, $p \equiv (b{+}; c{+}; b{-})$ and $Q_0 \equiv (b = 0)$. The backward strategy using our genetically modified Hoare logic on this example gives the following successive conditions.

The weakest precondition through the last instruction "$b-$" is (Decrementation rule):

$$\left\{\begin{array}{l} \Phi_b^{\varnothing} \Rightarrow K_b < b \\ \Phi_b^{\sigma} \Rightarrow K_{b,\sigma} < b \\ \Phi_b^{\lambda} \Rightarrow K_{b,\lambda} < b \\ \Phi_b^{\sigma,\lambda} \Rightarrow K_{b,\sigma\lambda} < b \\ b - 1 = 0 \end{array}\right. \equiv \left\{\begin{array}{l} (\neg\neg(c \geqslant 1) \wedge \neg(a \geqslant 1)) \Rightarrow K_b < b \\ (\neg\neg(c \geqslant 1) \wedge (a \geqslant 1)) \Rightarrow K_{b,\sigma} < b \\ (\neg(c \geqslant 1) \wedge \neg(a \geqslant 1)) \Rightarrow K_{b,\lambda} < b \\ (\neg(c \geqslant 1) \wedge (a \geqslant 1)) \Rightarrow K_{b,\sigma\lambda} < b \\ b - 1 = 0 \end{array}\right. \Leftrightarrow \left\{\begin{array}{l} b = 1 \\ ((c \geqslant 1) \wedge (a < 1)) \Rightarrow K_b = 0 \\ ((c \geqslant 1) \wedge (a \geqslant 1)) \Rightarrow K_{b,\sigma} = 0 \\ ((c < 1) \wedge (a < 1)) \Rightarrow K_{b,\lambda} = 0 \\ ((c < 1) \wedge (a \geqslant 1)) \Rightarrow K_{b,\sigma\lambda} = 0 \end{array}\right.$$

Then, the weakest precondition through "c+" is (Incrementation rule):

$$\begin{cases} \neg(a \geqslant 1) \Rightarrow K_c > c \\ a \geqslant 1 \Rightarrow K_{c,l} > c \\ b = 1 \\ ((c+1 \geqslant 1) \wedge (a < 1)) \Rightarrow K_b = 0 \\ ((c+1 \geqslant 1) \wedge (a \geqslant 1)) \Rightarrow K_{b,\sigma} = 0 \\ ((c+1 < 1) \wedge (a < 1)) \Rightarrow K_{b,\lambda} = 0 \\ ((c+1 < 1) \wedge (a \geqslant 1)) \Rightarrow K_{b,\sigma\lambda} = 0 \end{cases} \Leftrightarrow \begin{cases} c = 0 \\ a < 1 \Rightarrow K_c = 1 \\ a \geqslant 1 \Rightarrow K_{c,l} = 1 \\ b = 1 \\ a < 1 \Rightarrow K_b = 0 \\ a \geqslant 1 \Rightarrow K_{b,\sigma} = 0 \end{cases}$$

Lastly, through the first "b+" (Incrementation rule):

$$\begin{cases} (\neg\neg(c \geqslant 1) \wedge \neg(a \geqslant 1)) \Rightarrow K_b > b \\ (\neg\neg(c \geqslant 1) \wedge (a \geqslant 1)) \Rightarrow K_{b,\sigma} > b \\ (\neg(c \geqslant 1) \wedge \neg(a \geqslant 1)) \Rightarrow K_{b,\lambda} > b \\ (\neg(c \geqslant 1) \wedge (a \geqslant 1)) \Rightarrow K_{b,\sigma\lambda} > b \\ c = 0 \\ a < 1 \Rightarrow K_c = 1 \\ a \geqslant 1 \Rightarrow K_{c,l} = 1 \\ b+1 = 1 \\ a < 1 \Rightarrow K_b = 0 \\ a \geqslant 1 \Rightarrow K_{b,\sigma} = 0 \end{cases} \Leftrightarrow P_0 \equiv \begin{cases} a < 1 \Rightarrow K_{b,\lambda} = 1 \\ a \geqslant 1 \Rightarrow K_{b,\sigma\lambda} = 1 \\ c = 0 \\ a < 1 \Rightarrow K_c = 1 \\ a \geqslant 1 \Rightarrow K_{c,l} = 1 \\ b = 0 \\ a < 1 \Rightarrow K_b = 0 \\ a \geqslant 1 \Rightarrow K_{b,\sigma} = 0 \end{cases}$$

Then, using the Empty trace rule to finish the correctness proof of the Hoare triple, we have to ensure $P \Rightarrow P_0$ and, after simplification, we get the correctness if and only if $K_{b,\sigma\lambda} = 1$ and $K_{c,l} = 1$ and $K_{b,\sigma} = 0$. So, under these three hypotheses and whatever the values of the other parameters, the system can exhibit a transitory production of b in response to a switch of a from 0 to 1.

References

[BCK+15] G. Bernot, J.-P. Comet, Z. Khalis, A. Richard, and O. Roux. A genetically modified hoare logic. *ArXiv: 1506.05887*, June 2015.

[KCRB09] Z. Khalis, J.-P. Comet, A. Richard, and G. Bernot. The SMBioNet method for discovering models of gene regulatory networks. *Genes, Genomes and Genomics*, 3(special issue 1):15–22, 2009.

[Mt02] R. Milo *et al.* Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[TK01] R. Thomas and M. Kaufman. Multistationarity, the basis of cell differentiation and memory. II. logical analysis of regulatory networks in terms of feedback circuits. *Chaos*, 11:180–195, 2001.