HAL open science

A genetically modified Hoare logic

G. Bernot, Jean-Paul Comet, Z. Khalis, A. Richard, Olivier Roux

▶ **To cite this version:**

G. Bernot, Jean-Paul Comet, Z. Khalis, A. Richard, Olivier Roux. A genetically modified Hoare logic. Theoretical Computer Science, 2019, 10.1016/j.tcs.2018.02.003 . hal-02051795

HAL Id: hal-02051795

https://hal.science/hal-02051795v1

Submitted on 25 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Version of Record: https://www.sciencedirect.com/science/article/pii/S030439751830080X
Manuscript_24b6515d4a5004d15b90df79a6c2212b

A Genetically Modified Hoare Logic

G. Bernot[a], J.-P. Comet[a], Z. Khalis[a], A. Richard[a], O. Roux[b]

[a] *University Côte d'Azur*
I3S laboratory, UMR CNRS 7271,
CS 40121, 06903 Sophia Antipolis Cedex, France
[b] *LS2N UMR CNRS 6004, BP 92101,*
1 rue de la Noë, 44321 Nantes Cedex 3, France

Abstract

An important problem when modelling gene networks lies in the identification of parameters, even when using a discrete framework such as the one of René Thomas. We present in this article a new approach based on Hoare logic to generate constraints on parameter values. Specifications of observed behaviours play a role comparable to programs in the classical Hoare logic, and deduced weakest preconditions characterize the sets of all compatible parameterizations, expressed as constraints on parameters. Besides being natural and simple, our Hoare logic approach is remarkably powerful and, among others, it allows one to express external interventions of the biologist *during* experiments such as knockouts. In supplementary materials, we give a proof of soundness of our Hoare logic for gene networks as well as a proof of completeness and decidability based on the notion of the weakest precondition.

Keywords: Hoare logic, Gene regulatory networks, Thomas networks, Parameter identification, Soundness and completeness

1. Introduction

Different frameworks for studying the behaviour of gene networks in a systematic way have been proposed. Among them, ordinary differential equations played an important role, which however mostly lead to numerical simulations. Besides, the abstraction procedure of René Thomas [1], approximating sigmoid functions by step functions, makes it possible to describe the qualitative dynamics of gene networks as paths in a finite state

© 2018 published by Elsevier. This manuscript is made available under the CC BY NC user license
https://creativecommons.org/licenses/by-nc/4.0/

space. Nevertheless this qualitative description of the dynamics is still governed by a set of parameter values, which, although becoming small integers, remain difficult to deduce from classical experimental knowledge. In this context, we are interested in the exhaustive search of parameter values that are consistent with specifications formalizing the experimentally observed behaviours of gene regulatory networks. In addition an important quality of our approach, which is not addressed by other formalisms, is to take into account external interventions of the biologists *during* the experiments (*e.g.* knockouts).

Several works were undertaken with the objective to identify the parameters. The application of temporal logic to gene regulatory networks was presented in [2, 3], then constraint programming was used in [4, 5]. In this paper, we present a somewhat unexpected application of formal methods to biology through a new approach based on Hoare logic [6] and its associated weakest precondition calculus [7] that generates constraints on parameters. The formalism on which we decided to apply this idea is the one of René Thomas because it is now universally recognized as the reference framework for discrete modelling of gene networks. The key point of our proposal is to define a language able to capture the actual traces observed by molecular biologists during a set of experiments (either at the transcriptomic or proteomic level [8]). We have designed a language which is expressive enough to *specify* sets of observed traces as well as external interventions during the biological experiments, while preserving the completeness of a corresponding extended Hoare logic. Since this method avoids building the complete state graph, it results in a powerful technique to find out the constraints representing the set of consistent parameterizations with a tangible gain for computation time. Indeed, the weakest precondition proof strategy which extracts the constraints, goes through the trace specification syntax but is independent of the size of the gene network.

The paper is organized as follows. The basic concepts of classical Hoare logic and its associated Dijkstra weakest precondition are quickly reminded in Section 2. The classical formal definitions for Thomas discrete gene regulatory networks are reminded in Section 3. Section 4 gives our definition of (genetically modified) Hoare triples, including the assertion language and the trace specification language. In Section 5, an extended Hoare logic for gene networks is defined for Thomas discrete models. In Section 6, the small example of the *incoherent feedforward loop of type 1* (made popular by Uri Alon in [9, 10]) highlights the whole process of our approach to find out the

suitable parameter values. Section 7 sketches the previously existing methods for formal identification of discrete parameters in gene network models. We conclude in Section 8. Supplementary materials provide the mathematical semantics of these extended Hoare triples, a proof of soundness of our Hoare logic for gene networks, and a proof of completeness and decidability.

2. Basics of Hoare logic

The Hoare logic is a formal system for reasoning about the correctness of imperative programs. In [6], Tony Hoare introduced the notation "$\{P\}\ p\ \{Q\}$" to mean "If the assertion P (precondition) is satisfied before performing the program p and if the program terminates, then the assertion Q (postcondition) will be satisfied afterwards." This constitutes *de facto* a specification of the program under the form of a triple, called the Hoare triple. In [7], Edsger Dijkstra has defined an algorithm taking the postcondition Q and the program p as input and computing the *weakest precondition* P_0 that ensures Q if p terminates. In other words, *weakest* means that the Hoare triple $\{P_0\}\ p\ \{Q\}$ is satisfied and that for any precondition P, $\{P\}\ p\ \{Q\}$ is satisfied if and only if $P \Rightarrow P_0$ is semantically satisfied. *Notice that weakest precondition means that it does not contain any useless condition, so, it means that the set of states that satisfy the weakest precondition is the largest one.* The basic idea is to stamp the sequential steps of a program with assertions that are infered according to the instruction they surround.

Within the following inference rules, p, p_1 and p_2 stand for programs, P, P_1, P_2, I and Q stand for first-order assertions on the variables of the program, v stands for a variable of the imperative program, and $Q[v \leftarrow expr]$ means that $expr$ is substituted to each free occurrence of v in Q:

Assignment: $\dfrac{}{\{Q[v{\leftarrow}expr]\}\ v{:=}expr\ \{Q\}}$

Sequential composition: $\dfrac{\{P_2\}\ p_2\ \{Q\} \qquad \{P_1\}\ p_1\ \{P_2\}}{\{P_1\}\ p_1;p_2\ \{Q\}}$

Conditional branching: $\dfrac{\{P_1\}\ p_1\ \{Q\} \qquad \{P_2\}\ p_2\ \{Q\}}{\{(e{\wedge}P_1){\vee}(\neg e{\wedge}P_2)\}\ \texttt{if}\ e\ \texttt{then}\ p_1\ \texttt{else}\ p_2\ \{Q\}}$

Iteration: $\dfrac{\{e{\wedge}I\}\ p\ \{I\} \qquad \neg e{\wedge}I{\Rightarrow}Q}{\{I\}\ \texttt{while}\ e\ \texttt{with}\ I\ \texttt{do}\ p\ \{Q\}}$

Empty program: $\dfrac{P \Rightarrow Q}{\{P\}\ \varepsilon\ \{Q\}}$ (where ε stands for the empty program)

The *Iteration* rule deserves some comments. The assertion I is called the *loop invariant* and it is well known that finding the weakest loop invariant (if any) is undecidable in general [11, 12]. So, Tony Hoare asks the programmer to give a loop invariant explicitely (`with` I). There are approaches to help finding loop invariants such as the iterative approach adopted in ASTREE [13] (abstract interpretation [14]).

Some authors prefer the following iteration rule $\dfrac{\{e \wedge I\}\ p\ \{I\}}{\{I\}\ \texttt{while}\ e\ \texttt{with}\ I\ \texttt{do}\ p\ \{\neg e \wedge I\}}$ that requires the application of the empty program rule to become equivalent to our version. By doing so, these authors put the light on the fact that within a program, each `while` instruction carries its own (sub)specification and it can consequently be proved apart from the rest of the program.

From the standard set of Hoare logic rules, the following proof strategy builds a proof tree that computes the weakest precondition [7].

Definition 2.1. (Dijkstra Backward strategy). *Let $\{P\}\ p\ \{Q\}$ be a Hoare triple. We call* backward strategy *the proof strategy defined inductivelly on p as follows:*

1. *If p is of the form $p_1; p_2$ where p_2 is made of a single instruction, then apply the* Sequential composition *rule.*
2. *If p is a single instruction, then apply the corresponding rule (*Iteration *rule,* Conditional branching *rule or* assignment *rule).*
3. *Only after steps 1 and 2 have fully treated p, i.e. when all instructions have been treated, apply the* Empty program *rule.*

Notice that, these three items being mutually exclusive, the backward strategy generates a unique proof tree. (In addition, the remaining leafs of the proof tree must be handled using first order logic and arithmetic knowledge.)

By doing so, the precondition P_0 obtained just before applying the last *Empty program* rule is the weakest precondition. According to Stephen Cook [15], the Hoare logic is complete assuming that each loop invariant in the program is the weakest loop invariant with respect to the condition computed just at the right of its `while` statement. More technically, a program with a `while` statement is of the form: "p_1 ; `while` e `with` I `do` p ; p_2." The Dijkstra backward strategy computes inductively the weakest precondition Q_2 such that, after the execution of p_2, the postcondition is satisfied. So

Q_2 becomes the postcondition of the `while` statement. Cook's result is then valid when the invariant I is the weakest condition that ensures Q_2 if the program exits from the `while` statement. All in all, Cook's result means that the Hoare triple $\{P\}\ p\ \{Q\}$ is correct if and only if $P \Rightarrow P_0$ is semantically satisfied. So the full completeness of the Hoare logic depends on two things: a sufficient expressive power to express all the previously mentionned weakest loop invariants and the existence of a first-order proof tree for $P \Rightarrow P_0$ whenever it is semantically satisfied. Technically, this relies on the expressiveness of the chosen underlying assertion language [16].

The most striking feature of the backward strategy for Hoare logic is that, owing to very simple sequences of syntactic formula manipulations, we capture the mathematical semantics of a program within first order logic. Nevertheless, it is worth noticing that we only address *partial* correctness since Hoare logic does not give any proof of the termination of the program (`while` instructions may induce infinite loops).

3. Basics of discrete gene regulatory network models

This section presents the formal framework based on the discrete modelling method of René Thomas [17, 18] and introduced in [19]. As shown in

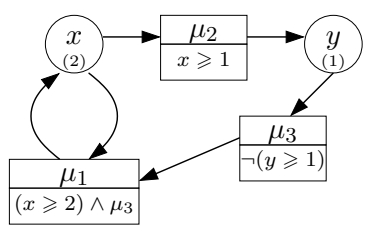

Figure 1: The graphical representation of a gene regulatory graph $R = (V, M, E_V, E_M)$ with $V = \{x, y\}$, the bounds of x and y are respectively 2 and 1, $M = \{\mu_1, \mu_2, \mu_3\}$, φ_{μ_1} is $((x \geqslant 2) \wedge \mu_3)$, φ_{μ_2} is $(x \geqslant 1)$, φ_{μ_3} is $\neg(y \geqslant 1)$.

Figure 1, a gene regulatory graph is visualized as a labelled directed graph 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 simple multiplex μ_2 expresses that the variable x can help the activation of the variable y when its state 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 simply contain a conjunction). In the figure, the situation of μ_1 is a little bit more elaborated: It reflects an auto-activation of x at level 2 which is controlled by μ_3. Because μ_3 contains a negation, μ_1 does not model a positive cooperation of x and y: The auto-activation of x is *inhibited* by y.

So, in this example, there are three qualitatively interesting intervals of expression levels for x: an interval called 0, where x can neither act on y nor on itself, an interval called 1, where x can act on y and never on itself, and an interval called 2, where x can act on y as well as on itself provided that μ_3 is satisfied. From the biological point of view, there is a threshold (*i.e.* a given number of intracellular molecules produced by x) such that x is unable (resp. able) to act on its target gene if its expression level is under (resp. over) the threshold.

We say that the bound of x is $b_x = 2$ and similarly there are only 2 qualitatively interesting intervals for y, so the bound of y is $b_y = 1$.

In general, this labelled directed graph is formally defined as follows.

Definition 3.1. *A* gene regulatory graph with multiplexes *is a tuple* $R = (V, M, E_V, E_M)$ *satisfying the following conditions:*

- *V and M are disjoint sets, whose elements are called* variables *and* multiplexes *respectively.*
- *$G = (V \cup M, E_V \cup E_M)$ is a labelled directed graph such that:*
  - *Edges of E_V start from a variable and end to a multiplex, and edges of E_M start from a multiplex and end to either a variable or a multiplex.*
  - *Every directed cycle of G contains at least one variable.*
  - *Every variable v of V is labelled by a positive integer b_v called the* bound *of v.*
  - *Every multiplex m of M is labelled by a formula φ_m belonging to the language $\mathcal{L}_m$ inductively defined by:*
    - *If $v \to m$ belongs to E_V and $s \in \mathbb{N}$, then $v \geqslant s$ is an atom of $\mathcal{L}_m$.*
    - *If $m' \to m$ belongs to E_M then m' is an atom of $\mathcal{L}_m$.*

– *If φ and ψ belong to $\mathcal{L}_m$ then $\neg\varphi$, $(\varphi \wedge \psi)$ and $(\varphi \vee \psi)$ also belong to $\mathcal{L}_m$.*

All in all, the discrete values of a variable x abstract intervals of quantity of molecules produced by x within the cell. These intervals are obtained by sorting the activation thresholds of x on its targets. Consequently only the knowledge of the thresholds order is useful and not their actual values. The multiplexes use these abstract levels in order to encode peculiar biological knowledge into formulas that define the conditions under which the regulation positively acts on its targets. If there is no peculiar knowledge about cooperation over a given target, there is one multiplex per regulating gene acting on this target, whose formula is reduced to an atom.

Successive multiplexes can be combined by flattening their formulas:

Definition 3.2. *The* flaten version *of a formula φ_m, denoted $\overline{\varphi_m}$, is obtained by recursively substituting each occurrence of a multiplex m' in φ_m by its formula $\varphi_{m'}$ (this recursive process of substitutions is well defined because G has no directed cycle with only multiplexes).*

In Figure 1, the flatten formula $\overline{\varphi_{\mu_1}}$ is $(x \geqslant 2) \wedge \neg(y \geqslant 1)$.
As a result of the flattening transformation, all the atoms of a flatten formula are of the form $v \geqslant s$.

A *state* is obviously an assignment of integer values to the variables v of V within the intervals $[0, b_v]$. According to a given state, by replacing variables by their values, $\overline{\varphi_m}$ becomes a propositional formula whose atoms are the results of the integer inequalities.

Definition 3.3. (States η, satisfaction relation $\models_N$ and resources ρ). *Let N be a* GRN *and V be its set of variables. A* state *of N is a function $\eta : V \to \mathbb{N}$ such that $\eta(v) \leqslant b_v$ for all $v \in V$. Let $\mathcal{L}$ be the set of propositional formulas whose atoms are of the form $v \geqslant s$ with $v \in V$ and $s \in \mathbb{N}^*$. The satisfaction relation $\models_N$ between a state η of N and a formula φ of $\mathcal{L}$ is inductively defined by:*

- *If φ is an atom of the form $v \geqslant s$, then $\eta \models_N \varphi$ if $\eta(v) \geqslant s$.*
- *If $\varphi \equiv \psi_1 \wedge \psi_2$ then $\eta \models_N \varphi$ if $\eta \models_N \psi_1$ and $\eta \models_N \psi_2$; and we proceed similarly for the other connectives.*

Given a variable $v \in V$, a multiplex $m \in N^-(v)$ (where $N^-(v)$ is the set of multiplexes m such that $m \to v$ belongs to the interaction graph of N) is a resource *of v at state η if $\eta \models_N \overline{\varphi_m}$.*
The set of resources *of v at state η is $\rho(\eta, v) = \{m \in N^-(v) \mid \eta \models_N \overline{\varphi_m}\}$.*

According to Figure 1, at the state where $\eta(x) = 2$ and $\eta(y) = 1$, $\overline{\varphi_{\mu_2}}$ is satisfied and consequently μ_2 is the only resource of y. On the contrary $\overline{\varphi_{\mu_1}}$ is false and consequently the set of resources of x is empty.

The equilibrium toward which the expression level of a gene v is attracted only depends on its set ω of resources. The interval number between 0 and b_v containing this equilibrium is classically denoted $K_{v,\omega}$ [20, 21, 17, 22, 2, 19].

Definition 3.4. *A* gene regulatory network *(*GRN *for short) is a couple $N = (V, M, E_V, E_M, \mathcal{K})$ satisfying the following conditions:*

- *$R = (V, M, E_V, E_M)$ is a gene regulatory graph with multiplexes,*
- *$\mathcal{K} = \{K_{v,\omega}\}$ is a family of integers indexed by $v \in V$ and $\omega \subset N^-(v)$, where $N^-(v)$ is the set of multiplexes m such that $m \to v$ is an edge of E_M. Each $K_{v,\omega}$ must satisfy $0 \leqslant K_{v,\omega} \leqslant b_v$.*

A usual notation abuse is the following: we write K_v instead of $K_{v,\varnothing}$ and we write $K_{v,m_1m_2\ldots}$ instead of $K_{v,\{m_1,m_2,\ldots\}}$.

At a given state η, each variable v tries to evolve in the direction of parameter $K_{v,\rho(\eta,v)}$. 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 v is stable if $\eta(v) = K_{v,\rho(\eta,v)}$.

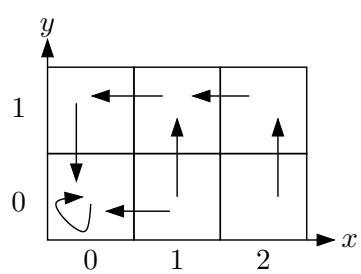

Figure 2: State graph obtained according to Definition 3.5, following Figure 1 and arbitrarily assuming that $K_x = 0$, $K_{x,\mu_1} = 2$, $K_y = 0$ and $K_{y,\mu_2} = 1$.

In Figure 2, at the state $(2, 1)$, we have $K_x = 0 < \eta(x) = 2$ and $K_{y,\mu_2} = \eta(y) = 1$, but $(0, 1)$ is not a successor state of $(2, 1)$ because the

protein degradation occurs one protein after the other and consequently the concentration level of x cannot jump from 2 to 0. Consequently $(1,1)$ is the next state.
At $(1,0)$, both $K_x = 0 < \eta(x) = 1$ and $K_{y,\mu_2} = 1 > \eta(y) = 0$, but the probability for x and y to cross their threshold exactly at the same time is null [20, 21, 17, 22, 2, 19][1]. Consequently, there are two possible next states: $(0,0)$ if x crosses its threshold first and $(1,1)$ if y crosses its threshold first.

So, Thomas method assumes that variables evolve asynchronously and by unit steps toward their respective target levels:

Definition 3.5. (State Graph). *Let $N = (V, M, E_V, E_M, \mathcal{K})$ be a* GRN. *The* state graph *of N is the directed graph $\mathcal{S}$ whose set of vertices is the set of states of N, and such that there exists an edge (called transition) $\eta \to \eta'$ if one of the following conditions is satisfied:*

- *For all variables $v \in V$ we have $\eta(v) = K_{v,\rho(\eta,v)}$, and then $\eta' = \eta$.*
- *There exists $v \in V$ such that $\eta(v) \neq K_{v,\rho(\eta,v)}$, and*

$$\eta'(v) = \begin{cases} \eta(v) + 1 & \text{if } \eta(v) < K_{v,\rho(\eta,v)} \\ \eta(v) - 1 & \text{if } \eta(v) > K_{v,\rho(\eta,v)} \end{cases} \qquad \text{and} \quad \forall u \neq v,\ \eta'(u) = \eta(u).$$

For each variable v such that $\eta(v) \neq K_{v,\rho(\eta,v)}$, there is a transition allowing v to evolve (± 1) toward its focal level $K_{v,\rho(\eta,v)}$. Every outgoing transition of η is supposed to be possible, so that there is an non-determinism as soon as η has several outgoing transitions. Figure 2 represents a complete state graph.

4. Syntax of Hoare triples for gene networks

In order to formalize known information about a gene network, we introduce in this section a language to express properties of states (assertions) and a language to express properties of state transitions (trace specifications).

[1]Indeed, biologically, each threshold corresponds to a precise number of molecules produced by x or y respectively in the cell. So, there is a probability 0 for the degradation to make the number of x-molecules cross the x-threshold *exactly at the same time* as a new molecule produced by y makes the y-threshold crossed (a sufficiently precise time scale will distinguish the two events).

4.1. Assertions for discrete models of gene networks

Definition 4.1. (Terms and Assertions). *Let $N = (V, M, E_V, E_M, \mathcal{K})$ be a* GRN. *The* well formed terms *for N are inductively defined by:*

- *Each integer $n \in \mathbb{N}$ constitutes a well formed term*
- *For each variable $v \in V$, the name of the variable v, considered as a symbol, constitutes a well formed term.*
- *Similarly, for each $v \in V$ and for each subset ω of $N^-(v)$, the symbol $K_{v,\omega}$ constitutes a well formed term.*
- *If t and t' are well formed terms then $(t + t')$ and $(t - t')$ are also well formed terms.*

Let $N = (V, M, E_V, E_M, \mathcal{K})$ be a GRN. *The* assertions *for N are inductively defined by:*

- *If t and t' are well formed terms then $(t = t')$, $(t < t')$, $(t > t')$, $(t \leqslant t')$ and $(t \geqslant t')$ are atomic assertions for N.*
- *If φ and ψ are assertions for N then $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ and $(\varphi \Rightarrow \psi)$ are also assertions for N.*

A state η of the network N satisfies *an assertion φ if and only if its interpretation is valid in $\mathbb{Z}$, after substituting each variable v by $\eta(v)$ and each symbol $K_{v,\omega}$ by its value according to the family $\mathcal{K}$. We note $\eta \models_N \varphi$.*

Moreover, conventionally, we denote "$\top$" the tautology (e.g. "$1 = 1$").

4.2. Trace specifications for discrete models of gene networks

When biologists observe the dynamics of gene expression levels along a set of experiments, they extract, as a direct experimental knowledge, some sets of observed *traces* (see Figure 3). It is consequently of first interest to see these sets of observations as basic elements for the specification of gene networks.

Definition 4.2. (Trace specifications). *Let $N = (V, M, E_V, E_M, \mathcal{K})$ be a* GRN. *The set of* trace specifications *for N is inductively defined by:*

- *For each $v \in V$ and $n \in [0, b_v]$ the expressions $v+$, $v-$ and $v := n$ are atomic trace specifications (respectively increase, decrease or assignment).*

- *If e is an assertion for N, then the expression $assert(e)$ is an atomic trace specification.*

- *If p_1 and p_2 are trace specifications then $(p_1; p_2)$ is also a trace specification (sequential composition). Moreover the sequential composition is associative, so that we can write $(p_1; p_2; \cdots; p_n)$ without intermediate parentheses.*

- *If p is a trace specification and if e and I are assertions for N, 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 $\forall(p_1, p_2)$ and $\exists(p_1, p_2)$ are also trace specifications (quantifiers). Moreover the quantifiers are associative and commutative, so that we can write $\forall(p_1, p_2, \cdots, p_n)$ and $\exists(p_1, p_2, \cdots, p_n)$ as useful abbreviations.*

Conventionally, we denote:

- ε (called the *empty trace*) the trace specification $assert(\top)$.

- *if e then p_1 else p_2* (called *conditional branching*) the trace specification $\exists(\ assert(e); p_1\ ,\ assert(\neg e); p_2\ )$, where p_1 and p_2 are any trace specifications and e is an assertion for N.

Intuitively, $v+$ (resp. $v-$) means that the biologist has observed that the expression level of variable v is increasing by one unit (resp. decreasing by one unit). $v := n$ means that the biologist has set the concentration level for gene v to the value n during the experiment (*e.g.* $v := 0$ for a knockout or $v := b_v$ for a saturation of the product of v). $assert(e)$ allows one to express a property of the current state without change of state. Sequential composition allows one to concatenate two trace specifications. The loop invariant I, as in classical Hoare logic, is a way to handle an unknown number of trace repetitions: It will facilitate proofs of Hoare triples. Finally it becomes possible to group together several trace specifications thanks to the quantifiers $\forall$ and $\exists$. These intuitions are formalized as follows *via* a binary relation between states and *sets* of states.

Notation 4.3. *For a state η, a variable v and $i \in [0, b_v]$, we note $\eta[v \leftarrow i]$ the state η' such that $\eta'(v) = i$ and for all $u \neq v$, $\eta'(u) = \eta(u)$.*

Definition 4.4. (Mathematical semantics of a trace specification). *Let $N = (V, M, E_V, E_M, \mathcal{K})$ be a* GRN*, let $\mathcal{S}$ be the state graph of N whose set of vertices is denoted S and let p be a trace specification for N. The binary relation $\overset{p}{\rightsquigarrow}$ is the smallest subset of $S \times \mathcal{P}(S)$ such that, for any state η:*

1. *If p is the atomic expression $v+$, then let us consider the state $\eta' = \eta[v \leftarrow (\eta(v) + 1)]$: If $\eta \rightarrow \eta'$ is a transition of $\mathcal{S}$ then $\eta \overset{p}{\rightsquigarrow} \{\eta'\}$.*
2. *If p is the atomic expression $v-$, then let us consider the state $\eta' = \eta[v \leftarrow (\eta(v) - 1)]$: If $\eta \rightarrow \eta'$ is a transition of $\mathcal{S}$ then $\eta \overset{p}{\rightsquigarrow} \{\eta'\}$.*
3. *If p is the atomic expression $v := i$, then $\eta \overset{p}{\rightsquigarrow} \{\eta[v \leftarrow i]\}$.*
4. *If p is of the form $assert(e)$, if $\eta \models_N e$, then $\eta \overset{p}{\rightsquigarrow} \{\eta\}$.*
5. *If p is of the form $\forall(p_1, p_2)$: If $\eta \overset{p_1}{\rightsquigarrow} E_1$ and $\eta \overset{p_2}{\rightsquigarrow} E_2$ then $\eta \overset{p}{\rightsquigarrow} (E_1 \cup E_2)$.*
6. *If p is of the form $\exists(p_1, p_2)$: If $\eta \overset{p_1}{\rightsquigarrow} E_1$ then $\eta \overset{p}{\rightsquigarrow} E_1$,* and *if $\eta \overset{p_2}{\rightsquigarrow} E_2$ then $\eta \overset{p}{\rightsquigarrow} E_2$.*
7. *If p is of the form $(p_1; p_2)$: If $\eta \overset{p_1}{\rightsquigarrow} F$ and if $\{E_e\}_{e \in F}$ is a F-indexed family of state sets such that $e \overset{p_2}{\rightsquigarrow} E_e$, then $\eta \overset{p}{\rightsquigarrow} (\bigcup_{e \in F} E_e)$.*
8. *If p is of the form (while e with I do p_0):*
   - *If $\eta \not\models_N e$ then $\eta \overset{p}{\rightsquigarrow} \{\eta\}$.*
   - *If $\eta \models_N e$ and $\eta \overset{p_0;p}{\rightsquigarrow} E$ then $\eta \overset{p}{\rightsquigarrow} E$.*

Detailed comments about this definition can be found in supplementary materials Appendix A.

4.3. Hoare triples

Similarly to Section 2, two assertions and one trace specification are used to constitute a Hoare triple for gene networks.

Definition 4.5. *A* Hoare triple *for a* GRN *N is an expression of the form $\{P\}\ p\ \{Q\}$ where P and Q are assertions for N, called pre- and post-condition respectively, and p is a trace specification for N.*

In practice P can describe a set of states where cells have been synchronised at the beginning of the experiment, for example all states for which the variable v has value zero ($P \equiv (v = 0)$), the trace specification p describes biologically

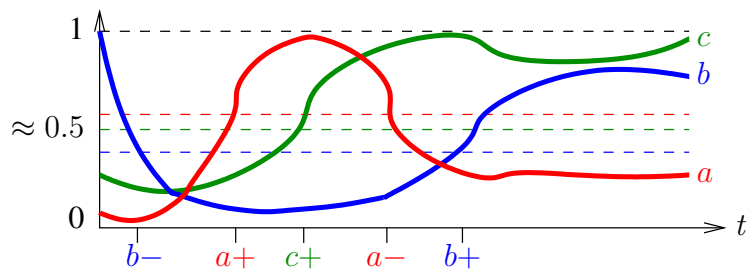

Figure 3: A classical example of normalised expression profiles for three Boolean genes a, b and c resulting from an experimental campaign. Thresholds for each gene are tuned according to biological knowledge. Then the trace specification for this figure is $b-; a+; c+; a-; b+$.

observed dynamic processes, for example increase of the expression level of v ($p \equiv v+$), and the postcondition also describes observations at the end of the experiment, for example all states for which the variable v has value one ($Q \equiv (v = 1)$), and so on.

More precisely we show in Figure 3 a classical representation of expression profiles obtained after an experimental campaign. From our numerous case studies, it is a good heuristics to consider by default equidistributed thresholds (*e.g.* a threshold of 0.5 for Boolean genes). If necessary, some thresholds are tuned after discussing with biologists. Then, successive crossings between a gene profile and its threshold give directly the trace specification. In practice when two crossings are very close, a $\exists$ statement is used ($\exists(x+; y+\ ,\ y+; x+)$) and the other primitives of trace specifications are often introduced in order to mix together and generalise several observed trace specifications.

Whether or not the triple is satisfied by a given gene network N, *will depend on its state transition graph*, thus it will depend on the parameter values in $\mathcal{K}$.

Definition 4.6. (Semantics of a Hoare triple). *Let $N = (V, M, E_V, E_M, \mathcal{K})$ be a* GRN *and let $\mathcal{S}$ be the state graph of N whose set of vertices is denoted S. A Hoare triple $\{P\}\ p\ \{Q\}$ is satisfied if and only if:*

For all $\eta \in S$ satisfying P, there exists E such that $\eta \overset{p}{\rightsquigarrow} E$ and for all $\eta' \in E$, η' satisfies Q.

See supplementary materials Appendix A for more details.

5. A Hoare logic for discrete models of gene networks

In this section, we define our *genetically modified Hoare logic* by giving the rule for each constructor of trace specifications (Definition 4.2). First, let us introduce a few conventional names to denote formulas that will be intensively used.

Notation 5.1. *For each variable v of a* GRN *N, we conventionally use the following notations:*

1. *For each subset ω of $N^-(v)$ we denote by Φ_v^ω the following formula*
$$\Phi_v^\omega \quad \equiv \quad (\bigwedge_{m \in \omega} \overline{\varphi_m}) \wedge (\bigwedge_{m \in N^-(v) \setminus \omega} \neg\overline{\varphi_m})$$
where $N^-(v) \setminus \omega$ stands for the complementary subset of ω in $N^-(v)$. From Definition 3.3, for all states η, $\eta \models_N \Phi_v^\omega$ if and only if $\omega = \rho(\eta, v)$, that is, ω is the set of resources of v at state η. Consequently, for each v, there exists a unique *ω such that $\eta \models_N \Phi_v^\omega$.*
2. *We denote by Φ_v^+ the following formula*
$$\Phi_v^+ \quad \equiv \quad \bigwedge_{\omega \subset N^-(v)} (\Phi_v^\omega \Longrightarrow K_{v,\omega} > v)$$
From Definition 3.5, we have $\eta \models_N \Phi_v^+$ if and only if there is a transition $(\eta \to \eta[v \leftarrow v+1])$ in the state graph $\mathcal{S}$, that is, if and only if the variable v can increase.
3. *We denote by Φ_v^- the following formula*
$$\Phi_v^- \quad \equiv \quad \bigwedge_{\omega \subset N^-(v)} (\Phi_v^\omega \Longrightarrow K_{v,\omega} < v)$$
Similarly, $\eta \models_N \Phi_v^-$ if and only if the variable v can decrease from the state η in the state graph $\mathcal{S}$.

See Section 6 where examples of these formulas are given.

Our Hoare logic for discrete models of gene networks is then defined by the following inference rules, where v is a variable of the GRN and $k \in [0, b_v]$.

1. Rules encoding Thomas discrete dynamics.

Increase: $$\frac{}{\{\ \Phi_v^+ \ \wedge\ Q[v{\leftarrow}v{+}1]\ \}\ v{+}\ \{Q\}}$$

Decrease: $$\frac{}{\{\ \Phi_v^- \ \wedge\ Q[v{\leftarrow}v{-}1]\ \}\ v{-}\ \{Q\}}$$

2. Rules coming from Hoare logic. These rules are similar to the ones given in Section 2. Obvious rules for the expression $assert(\Phi)$, and for the quantifiers, are added:

   Assert: $$\frac{}{\{\ \Phi \wedge Q\ \}\ assert(\Phi)\ \{\ Q\ \}}$$

   Universal quantifier: $$\frac{\{P_1\}\ p_1\ \{Q\} \qquad \{P_2\}\ p_2\ \{Q\}}{\{P_1 \wedge P_2\}\ \forall(p_1,p_2)\ \{Q\}}$$

   Existential quantifier: $$\frac{\{P_1\}\ p_1\ \{Q\} \qquad \{P_2\}\ p_2\ \{Q\}}{\{P_1 \vee P_2\}\ \exists(p_1,p_2)\ \{Q\}}$$

   Assignment: $$\frac{}{\{Q[v \leftarrow k]\}\ v{:=}k\ \{Q\}}$$

   Sequential composition: $$\frac{\{P_1\}\ p_1\ \{P_2\} \qquad \{P_2\}\ p_2\ \{Q\}}{\{P_1\}\ p_1;p_2\ \{Q\}}$$

   Iteration: $$\frac{\{e \wedge I\}\ p\ \{I\} \qquad \neg e \wedge I \Rightarrow Q}{\{I\}\ while\ e\ with\ I\ do\ p\ \{Q\}}$$

   Empty trace: $$\frac{P \Rightarrow Q}{\{P\}\ \varepsilon\ \{Q\}}$$

3. **Boundary axiom** asserting that all values stay between their bounds, for each $v \in V$ and $\omega \subset N^-(v)$:
$$0 \leqslant v \quad \wedge \quad v \leqslant b_v \quad \wedge \quad 0 \leqslant K_{v,\omega} \quad \wedge \quad K_{v,\omega} \leqslant b_v$$

Remark 5.2.

- *$(\Phi_v^+ \Rightarrow v < b_v)$ can be deduced from the boundary axioms: Φ_v^+ implies that for ω corresponding to the current set of resources, $K_{v,\omega} > v$ and, using the boundary axiom $K_{v,\omega} \leqslant b_v$, we get $v < b_v$.*
- *Similarly, we have $(\Phi_v^- \Rightarrow v > 0)$.*

These implications will be used in Section 6.

The conditional branching rule of the standard Hoare logic has not been reproduced here because the trace specification (if e then p_1 else p_2) is a shorthand for $\exists(\ assert(e); p_1\ ,\ assert(\neg e); p_2\ )$. The conditional branching rule remains sound.

We prove in Supplementary Materials Appendix B that this modified Hoare logic is sound and complete and we show that the weakest loop invariants can always be computed. This implies the decidability of the (partial)

correctness of any genetically modified Hoare triple. More precisely, the proof strategy called *backward strategy*, already described at the end of Section 2, also applies here: It automatically computes the loop invariants and the weakest precondition W of the Hoare triple $\{P\}\ p\ \{Q\}$, and the implication $P \Rightarrow W$ is decidable.

Similarly to classical Hoare logic which reflects a *partial* correctness of imperative programs, the previous definition does not imply termination of *while* loops.

6. Illustrative examples

6.1. Alon's interpretation of the incoherent feedforward loop of type 1.

In [9, 10] Uri Alon and co-workers have studied the most common *in vivo* patterns involving at most four genes. Among them, even without considering feedback loops such as in [23], there are interesting patterns whose dynamics is less obvious than it seems. In particular they have emphasized the *incoherent feedforward loop of type 1.* It is composed by a transcription factor a that activates a second transcription factor c, and both a and c regulate a gene b. The gene a is an activator of b whereas the gene c is an inhibitor of b. There is a "short" positive action of a on b and a "long" negative action *via* c: a activates c which inhibits b. The left hand side of Figure 4 shows such a feedforward loop. Supposing that both thresholds of actions of a are equal leads to a Boolean network since, in that case, the variable a can take only the value 0 (a has no action) or 1 (a activates both b and c). The right hand side of the figure shows the corresponding GRN with

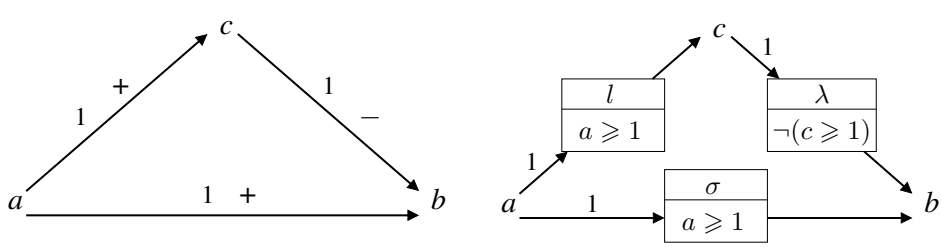

Figure 4: (Left) Boolean "incoherent feedforward loop of type 1" according to Uri Alon. (Right) Corresponding GRN $N{=}(V, M, E_V, E_M, \mathcal{K})$. $V{=}\{a, b, c\}$ with $b_a{=}b_b{=}b_c{=}1$. $M{=}\{l, \lambda, \sigma\}$, $\phi_l \equiv (a \geqslant 1)$, $\phi_\lambda \equiv (\neg(c \geqslant 1))$, $\phi_\sigma \equiv (a \geqslant 1)$. $\mathcal{K}{=}\{K_a, K_c, K_{c,l}, K_b, K_{b,\sigma}, K_{b,\lambda}, K_{b,\sigma\lambda}\}$.

multiplexes: σ encodes the "short" action of a on b, whilst l followed by λ constitutes the "long" action.

Classical interpretation: *Uri Alon and many biologists have in mind that if a is equal to 0 for a sufficiently long time, both b and c will also be equal to 0, because b and c need a as a resource in order to reach the state 1. They also have in mind that 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.*

In the following subsections, we revisit this affirmation *via* four different trace specifications, and we prove formally that the affirmation is only valid under some constraints on the parameters of the network, and only under the assumption that b starts its activity before c.

6.2. Is a transitory production of b possible?

The simple popular idea that b is activated and then the activation of c inhibits b is specified by the Hoare triple $\{P\}\ \mathcal{P}_1\ \{Q_0\}$ where $P \equiv (a = 1\ \wedge\ b = 0\ \wedge\ c = 0)$, $\mathcal{P}_1 \equiv (b{+}; c{+}; b{-})$ 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 obtained through the last expression "$b-$" is $\Phi_b^- \wedge Q_0[b \leftarrow b{-}1]$ (Decrease rule):

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

$$\text{which simplifies as } Q_1 \quad \equiv \quad \begin{cases} b = 1 \\ ((c \geqslant 1) \wedge (a < 1)) \Longrightarrow K_b = 0 \\ ((c \geqslant 1) \wedge (a \geqslant 1)) \Longrightarrow K_{b,\sigma} = 0 \\ ((c < 1) \wedge (a < 1)) \Longrightarrow K_{b,\lambda} = 0 \\ ((c < 1) \wedge (a \geqslant 1)) \Longrightarrow K_{b,\sigma\lambda} = 0 \end{cases}$$

- Then, the weakest precondition obtained through the expression "$c+$" is $\Phi_c^+ \wedge Q_1[c \leftarrow c+1]$:

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

$$Q_2 \quad \equiv \quad \begin{cases} c = 0 \\ a < 1 \Rightarrow K_c = 1 \\ a \geqslant 1 \Rightarrow K_{c,l} = 1 \\ b = 1 \\ a < 1 \Rightarrow K_b = 0 \\ a \geqslant 1 \Rightarrow K_{b,\sigma} = 0 \end{cases}$$ using the boundary axioms and Remark 5.2.

- Lastly, the weakest precondition obtained through the first "$b+$" of the trace is $\Phi_b^+ \wedge Q_2[b \leftarrow b+1]$ which simplifies as $Q_3 \equiv \begin{cases} a < 1 \Rightarrow K_{b,\lambda} = 1 \\ a \geqslant 1 \Rightarrow K_{b,\sigma\lambda} = 1 \\ c = 0 \\ a < 1 \Rightarrow K_c = 1 \\ a \geqslant 1 \Rightarrow K_{c,l} = 1 \\ b = 0 \\ a < 1 \Rightarrow K_b = 0 \\ a \geqslant 1 \Rightarrow K_{b,\sigma} = 0 \end{cases}$

Then, using the Empty trace rule, it follows that $P \Longrightarrow Q_3$ *i.e.* $(a = 1 \wedge b = 0 \wedge c = 0) \Longrightarrow Q_3$. After simplification we get 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.

6.3. Is a transitory production of b possible without increasing c?

The previous trace specification $\mathcal{P}_1$ is not the only one reflecting a transitory production of b, there may be other realisations of this property. For example one can consider the trace specification

$$\mathcal{P}_2 \equiv (b+; b-).$$

With respect to this trace specification, the weakest precondition obtained through the last expression "$b-$" is of course Q_1 as previously. Then, the weakest precondition obtained through "$b+$" is

$$Q_4 \quad \equiv \quad \begin{cases} b = 0 \\ ((c \geqslant 1) \wedge (a < 1)) \Longrightarrow ((K_b = 1) \wedge (K_b = 0)) \\ ((c \geqslant 1) \wedge (a \geqslant 1)) \Longrightarrow ((K_{b,\sigma} = 1) \wedge (K_{b,\sigma} = 0)) \\ ((c < 1) \wedge (a < 1)) \Longrightarrow ((K_{b,\lambda} = 1) \wedge (K_{b,\lambda} = 0)) \\ ((c < 1) \wedge (a \geqslant 1)) \Longrightarrow ((K_{b,\sigma\lambda} = 1) \wedge (K_{b,\sigma\lambda} = 0)) \end{cases}$$

Q_4 is not satisfiable: It implies that each parameter associated with b is both equal to 0 and 1. The trace $(b+; b-)$ is not realisable (inconsistent weakest precondition).

6.4. The existence of the trace $(b+, c+, b-)$ does not imply a transitory production of b for all traces in the same gene network.

When $K_{b,\sigma\lambda} = 1$, $K_{c,l} = 1$ and $K_{b,\sigma} = 0$, that is when trace $(b+, c+, b-)$ is realisable, this does not prevent from some other traces that *do not* exhibit a transitory production of b. For instance the simple trace specification $\mathcal{P}_3 \equiv c+$ leaves b constantly equal to 0, and the Hoare triple

$$\left\{ \begin{array}{l} a = 1 \ \wedge \ b = 0 \ \wedge \ c = 0 \ \wedge \\ K_{b,\sigma\lambda} = 1 \ \wedge \ K_{c,l} = 1 \ \wedge \ K_{b,\sigma} = 0 \end{array} \right\} \ c+ \ \ \{ \ b = 0 \ \}$$

is satisfied, as the corresponding weakest precondition Q_5 is clearly implied by the precondition.

$$Q_5 \quad \equiv \quad \Phi_c^+ \wedge Q_0[c \leftarrow c+1] \quad \equiv \quad \begin{cases} c = 0 \\ a = 0 \Longrightarrow K_c = 1 \\ a = 1 \Longrightarrow K_{c,l} = 1 \\ b = 0 \end{cases}$$

6.5. Once a constantly equals 1, if c reaches level 1 before b, even transitorily, then no production of b is possible anymore.

We prove this property by showing that the following triple is inconsistent, whatever the loop invariant I:

$$\left\{ \begin{array}{l} a = 1 \ \wedge \ b = 0 \ \wedge \\ c = 1 \ \wedge K_{b,\sigma\lambda} = 1 \ \wedge \\ K_{c,l} = 1 \ \wedge \ K_{b,\sigma} = 0 \end{array} \right\} \underbrace{\textit{while } b < 1 \textit{ with } I \textit{ do } \exists(b+, b-, c+, c-)}_{\mathcal{P}_4} \{b = 1\}$$

The sub-trace specification $\exists(b+, b-, c+, c-)$ reflects the fact that a stays constant but b or c evolves. Thus, the *while* statement allows b and c to evolve freely until b becomes equal to 1.
Applying the Iteration rule, I has to satisfy $\neg(b < 1) \wedge I \Longrightarrow (b = 1)$: This property is trivially satisfied whatever the assertion I, due to the boundary axioms. I has also to satisfy $\{b < 1 \wedge I\}\ \exists(b+, b-, c+, c-)\ \{I\}$ which gives *via* the existential quantifier rule:

$$Q_6 \quad \equiv \quad \left\{ \begin{array}{l} (\Phi_b^+ \wedge I[b \leftarrow b+1]) \ \vee \ (\Phi_b^- \wedge I[b \leftarrow b-1]) \ \vee \\ (\Phi_c^+ \wedge I[c \leftarrow c+1]) \ \vee \ (\Phi_c^- \wedge I[c \leftarrow c-1]) \end{array} \right.$$

Consequently I must be any assertion such that

$$(b = 0 \wedge I) \Longrightarrow Q_6$$

Let us denote P the precondition of the trace specification $\mathcal{P}_4$. Applying the Empty trace rule, it results that I must also satisfy $P \Longrightarrow I$. So, because $P \Longrightarrow (b = 0)$, we have $P \Longrightarrow (b = 0 \wedge I)$, which, in turn implies Q_6. Moreover, let us remark that $Q_6 \Longrightarrow (\Phi_b^+ \vee \Phi_b^- \vee \Phi_c^+ \vee \Phi_c^-)$. Consequently, if the Hoare triple of $\mathcal{P}_4$ is correct, then $P \Longrightarrow (\Phi_b^+ \vee \Phi_b^- \vee \Phi_c^+ \vee \Phi_c^-)$ which is impossible because, if P is satisfied then

- Φ_b^+ is false, as $a = 1$, $c = 1$ and $K_{b,\sigma} = 0$
  (indeed,Φ_b^+ implies $a = 1 \wedge c = 1 \Rightarrow K_{b,\sigma} > 0$)
- Φ_b^- is false, as $b = 0$ (Φ_b^- implies $b > 0$)
- Φ_c^+ is false, as $c = 1$ (Φ_c^+ implies $c < 1$)
- Φ_c^- is false, as $a = 1$, $c = 1$ and $K_{c,l} = 1$
  (Φ_c^- implies $a = 1 \wedge c = 1 \Rightarrow K_{c,l} < 1$).

So, we have formally proved that when a is constantly equal to 1, as soon as c has reached the level 1, it becomes *never possible* for b to increase to 1.

As mentioned in the beginning of this section, this proof contradicts the *universality* of the *classical interpretation* of this incoherent feedforward loop of type 1. We believed interesting to use our genetically modified Hoare logic for synthesising the parameter values for which the presupposed function of the incoherent feedforward loop of type 1 can hold. In [9, 10] the pulse of b in response of the switch of a is meant as a robust property. As formally established here, this robustness does not mean that the property holds for all parameter values, nor for the parameter values where the pulse can arise. As established in Subsections 6.4 and 6.5, it is necessary to ensure, in addition, that b will always increase *before* c in a robust manner.

6.6. About the scalability of the approach

The incoherent feedforward loop of type 1 example is of particularly small size for pedagogical reasons. We used our genetically modified Hoare logic on several examples including the classical epigenetic switch of λ phage [24] and, in cooperation with biologists, other examples of credible size such as the mucus production in *P. aeruginosa* [25], the circadian clock [26] or the cell cycle in mammals [27]. In all examples the computation of the weakest precondition takes less than one tenth of a second on a standard laptop (dual core, 2GHz) [24, 28]. What can take time is the resolution of constraints, varying from ten seconds to one day, depending on the chosen constraint solver and the problem under consideration (CTL based softwares require several days to model check all the possible sets of parameter values).

On the mammal cell cycle example, inspired by the model proposed by John Tyson in [29], we made a discrete model with 5 variables and 11 multiplexes. We obtained a set of 339,738,624 possible valuations, each model with 48 states and 26 parameters. From biological knowledge we extracted 12 trace specifications. After applying our Hoare logic method, 13 parameters were entirely identified (50%) and only 8,192 valuations remained possible according to the generated constraints (0.002%). Lastly additional reachability properties (endoreplication and quiescent phase) have been necessary to identify all parameters by formalizing them into temporal logic. For more details, see [27] in which the obtained discrete model has then been extended into a hybrid model with real time behaviour.

7. Related Works

One of the main motivations for the introduction of formal methods in discrete modelling of gene networks (or any complex system) is the *automation of parameter identification*. Our genetically modified Hoare logic is entirely dedicated to this problem of parameter identification for discrete gene networks. There are other formal methods which address this question, which we summarize briefly in this section.

The first approaches based on Thomas modelling used hand-made identification. They used known mathematical properties on circuits[2] in order to reduce the number of admissible parameter values and then, Ren Thomas and

[2]An observed homeostasy is necessarily generated by a so-called negative circuit and a notion of "characteristic states" provides necessary inequalities on parameter values.

Marcelle Kaufman used simulations on a "trial and error" method [32, 33]. Later on, simulation softwares helped systematic simulations, mainly the Hidde de Jong et al. system GNA [34] and the Denis Thieffry et al. system GINsim [35] that also include some tools for the determination of invariants. On biological systems where sufficient biological knowledge drastically limits the possible parameter values, approaches purely based on simulations remain efficient [36]. See also the article of Jasmin Fisher and Thomas Henzinger [37] for a complementary survey on simulation and mathematical models for biology.

The first use of the power of formal methods really comes with temporal logics and CTL model checking with our software SMBioNet [2]. Later on, GNA also included some aspects of CTL model checking and Alexander Bockmayr and Heike Siebert [38] introduced timed automata using UPPAAL. Mirco Giacobbe et al. [39] proposed a simplified (synchronous and deterministic) dynamics for gene networks, and a modified LTL model checking allowed for efficient generation of constraints on parameters. With respect to the general asynchronous and non deterministic dynamics, constraint solving introduced by Laurent Trilling and co-workers efficiently complemented the CTL temporal logic approach [4, 5] as well as symbolic execution techniques [3] introduced by Pascale Legall and co-workers. More detailed descriptions of these methods and their variants can be found in [40, 25]. These approaches fully take benefit from biological expertise, formalizing knowledge into temporal formulas but they need a large interpretation capacity of the experimental observations. This was our motivation to introduce Hoare Logic which uses trace specifications directly extracted from experiments.

Following the same motivation, Heike Siebert and co-workers [41] encoded time-series measurements into CTL formulas. Their approach is able to take into account partially known time-series measurements using repeatedly encapsulated EF statements. Then, they use softwares such as *SMBioNet* in order to identify the parameters. The price to pay is a huge computation time to identify the parameters, compared to constraint solving. Also, compared to our Hoare Logic, neither assignment, nor quantifier nor iteration are possible. Notice that although Siebert's approach is based on a modal logic,

---

Similarly, an observed multistationarity is necessarily generated by a so-called positive circuit in the gene network and characteristic states of positive circuits play a similar role. For more results about circuits, oscillations and attraction basins, see [17, 30, 31] among others.

a procedure based on tableau semantics [42, 43], does not apply because the objective of using time-series from biological experiments is, similarly to our approach, to extract *constraints* on the Thomas parameters; it is not to prove the satisfiability of the considered time-series[3].

On the semantic side, Definition 4.4 is in fact rather natural and similar ideas have been used by David Peleg and by Matthew Hennessy for concurrent systems in computer science [44, 45] where the authors defined a mathematical semantics for concurrent propositional dynamic logic. Our definition has a slightly different treatment of quantifiers, disjunctions and conjunctions in order to cope with the biological meaning of non-determinism.

Last but not least, whatever the aforementioned formalism, there is no possibility to model an intervention of the biologist *during* the experiment. Knockouts of genes are typical examples of such interventions. In our formalism they are easy to express in trace specifications, using assignment expressions (such as $v := 0$). They are not directly expressible in the other formalisms, including CTL or LTL, because the logic formulas they consider are by definition satisfied (or not) according to the paths *within a given model*. Indeed, a model of any of the aforementioned formalisms is, to some extend, based on the exhaustive set of transitions between states that can be triggered in "normal" conditions, that means without any external intervention. Consequently, such interventions do not correspond to transitions of the model. Because the semantics of temporal logics is defined on paths within the model (sequences of transitions inside the model), these logics cannot directly address external interventions.

Let us additionally remark that Patrick and Radhia Cousot's abstract interpretation [14] subsumes the Hoare logic, so a natural question is *should we use genetically modified abstract interpretation instead of genetically modified Hoare logic?* The technical point is that the dynamics of Thomas networks is formalized in an easy way using Hoare inference rules, whereas abstract interpretation would make things more complicated. The empirical point is that Hoare triples facilitate discussions with biologists because trace specifications cope very well with classical normalised expression profiles obtained

[3]Notice also that, although both Dijkstra weakest precondition algorithm and the tableau procedure for LTL go backwards, they are intrinsically different. In particular, in the Hoare approach as well as ours, the size of the formulas built by the Dijkstra algorithm increases up to the final constraint, contrarily to tableau procedure that builds a sequence of decreasing sub-formulas of the considered formula.

experimentally, see Section 4.3 and Figure 3.

8. Conclusion

In this paper, based on the discrete Thomas framework, we have developed a trace specification language that easily captures experimental observations of biologists when they study a gene network. This language can also take into account the possible interventions of the biologist during the experiments. Based on Hoare logic and Hoare triples as well as Dijkstra weakest precondition calculus, we have developed an automatic extraction of constraints that fully characterizes under which conditions a Thomas model is compatible with these experimental observations. The proposed approach has the advantage of being simple, leading to an efficient algorithm that depends only on the size of the trace specification (and not on the size of the gene network), without requiring simplifications.

As a consequence of our theorems, when a genetically modified Hoare triple is correct, we are always able to automatically generate all the weakest loop invariants and to build a *syntactic proof tree* that establishes the correctness[4]. In other words, the assertion language of Definition 4.1 is expressive enough to ensure the *purely logical* soundness and decidability of our genetically modified Hoare logic with *while* loops and quantifiers. This is an important step towards a systematic exploitation of the numerous gene expression traces available in biological databases.

One may easily imagine similar works for many applications besides gene networks. When modelling any complex system, the cornerstone lies, whatever the application domain, in the identification of the parameters. Hoare logic was initially designed for proofs of imperative programs. In this paper, we divert this approach for exhibiting constraints on parameters of gene network models. One can imagine several other adaptations for several types of discrete complex systems, the key point is to extract from the considered underlying modelling framework, a first order formula that characterizes the conditions under which a transition exists.

[4]assuming that the path specification terminates.

Acknowledgement

The authors thank the French National Agency for Research (ANR-14-CF09-0011 *HyClock* project) for its support. This work has also been partly supported by the ANR-10-BLANC-0218 *BioTempo* project, by the CNRS PEPII project *CirClock* and by the European PHC PROCOPE project *TiGeRNet*.

References

[1] R. Thomas, Regulatory networks seen as asynchronous automata : A logical description, J. theor. Biol. 153 (1991) 1–23.

[2] G. Bernot, J.-P. Comet, A. Richard, J. Guespin, Application of formal methods to biological regulatory networks: Extending Thomas' asynchronous logical approach with temporal logic, Journal of Theoretical Biology 229 (3) (2004) 339–347.

[3] D. Mateus, J.-P. Gallois, J.-P. Comet, P. Le Gall, Symbolic modeling of genetic regulatory networks, Journal of Bioinformatics and Computational Biology 5 (2B) (2007) 627–640.

[4] E. Fanchon, F. Corblin, L. Trilling, B. Hermant, D. Gulino, Modeling the molecular network controlling adhesion between human endothelial cells: Inference and simulation using constraint logic programming, in: CMSB, 2004, pp. 104–118.

[5] F. Corblin, S. Tripodi, E. Fanchon, D. Ropers, L. Trilling, A declarative constraint-based method for analyzing discrete genetic regulatory networks, Biosystems 98 (2) (2009) 91–104.

[6] C. Hoare, An axiomatic basis for computer programming, Communications of the ACM 12 (10) (1969) 576–585.

[7] E. W. Dijkstra, Guarded commands, nondeterminacy and formal derivation of programs, Commun. ACM 18 (1975) 453–457.

[8] A. Bernot, Genome transcriptome and proteome analysis, John Wiley & Sons, 2004.

[9] S. Shen-Orr, R. Milo, S. Mangan, U. Alon, Network motifs in the transcriptional regulation network of escherichia coli, Nature Genetics 31 (2002) 64–68.

[10] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon, Network motifs: Simple building blocks of complex networks, Science 298 (2002) 824–827.

[11] W. Hatcher, A semantic basis for program verification, J. of Cybernetics 4 (1) (1974) 61–69.

[12] A. Blass, Y. Gurevich, Inadequacy of computable loop invariants, ACM Transactions on Computational Logic 2 (1) (2001) 1–11.

[13] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Min, D. Monniaux, X. Rival, The ASTREE analyser., in: M. Sagiv (Ed.), ESOP 2005 The European Symposium on Programming, no. 3444 in LNCS, Springer, 2005, pp. 21–30.

[14] P. Cousot, R. Cousot, Basic concepts of abstract interpretation., in: R. Jacquard (Ed.), Building the Information Society, Kluwer Academic, 2004, pp. 359–366.

[15] S. A. Cook, Soundness and completeness of an axiom system for program verification, SIAM Journal on Computing 7 (1) (1978) 70–90.

[16] D. Kozen, J. Tiuryn, On the completeness of propositional Hoare logic, Information Sciences 139 (3) (2001) 187–195.

[17] R. Thomas, R. d'Ari, Biological Feedback, CRC Press, 1990.

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

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

[20] R. Thomas, A. Gathoye, L. Lambert, A complex control circuit. regulation of immunity in temperate bacteriophages., Eur. J. Biochem. 71 (1) (1976) 211–27.

[21] R. Thomas, Logical analysis of systems comprising feedback loops., J. Theor. Biol. 73 (4) (1978) 631–56.

[22] E. Snoussi, R. Thomas, Logical identification of all steady states : the concept of feedback loop caracteristic states, Bull. Math. Biol. 55 (5) (1993) 973–991.

[23] B. Yordanov, G. Batt, C. Belta, Model checking discrete-time piecewise affine systems: application to gene networks, in: Control Conference (ECC), 2007 European, IEEE, 2007, pp. 2619–2626.

[24] Z. Khalis, Logique de Hoare et identification formelle des paramtres d'un réseau génétique, Ph.D. thesis, University of Evry-Val d'Essonne (2010).

[25] G. Bernot, J.-P. Comet, H. Snoussi, Formal methods applied to gene network modelling, in: L. Farinas del Cerro, K. Inoue (Eds.), Logical Modeling of Biological Systems, Bioengineering and health science series, ISTE & Wiley, ISBN 978-1-84821-680-8, 2014, pp. 245–289.

[26] E. Cornillon, Modles qualitatifs de réseaux génétiques: réduction de modles et introduction d'un temps continu, Ph.D. thesis, Université Cote d'Azur (2017).

[27] J. Behaegel, J.-P. Comet, G. Bernot, E. Cornillon, F. Delaunay, A hybrid model of cell cycle in mammals, Journal of Bioinformatics and Computational Biology 14 (1) (2016) In press.

[28] M. Folschette, Application de la logique de Hoare aux réseaux de régulation génétique avec multiplexes, Master's thesis, ECN, Nantes, France (2011).

[29] J. Tyson, B. Novak, Temporal organization of the cell cycle, Current Biology 18 (17) (2008) R759–R768.

[30] A. Richard, Negative circuits and sustained oscillations in asynchronous automata networks, Advances in Applied Mathematics 44 (4) (2010) 378–392.

[31] A. Richard, J.-P. Comet, Necessary conditions for multistationarity in discrete dynamical systems, Discrete Applied Mathematics 155 (18) (2007) 2403–2413.

[32] M. Kaufman, J. Urbain, R. Thomas, Towards a logical analysis of the immune response, Journal of theoretical biology 114 (4) (1985) 527–561.

[33] R. Thomas, M. Kaufman, Multistationarity, the basis of cell differentiation and memory. I. & II., Chaos 11 (2001) 170–195.

[34] H. de Jong, J. Geiselmann, C. Hernandez, M. Page, Genetic network analyzer: qualitative simulation of genetic regulatory networks., Bioinformatics 19 (3) (2003) 336–44.

[35] A. Gonzalez, A. Naldi, L. Sanchez, D. Thieffry, C. Chaouiya, Ginsim: a software suite for the qualitative modelling, simulation and analysis of regulatory networks, Biosystems 84 (2) (2006) 91–100.

[36] R. Khoodeeram, G. Bernot, J.-Y. Trosset, An ockham razor model of energy metabolism, in: P. Amar, F. Kps, V. Norris (Eds.), Proc. of the Thematic Research School on Advances in Systems and Synthetic Biology, EDP Science, 2017, pp. 81–101.

[37] J. Fisher, T. Henzinger, Executable cell biology, Nature biotechnology 25 (11) (2007) 1239.

[38] H. Siebert, A. Bockmayr, Temporal constraints in the logical analysis of regulatory networks, Theoretical Computer Science 391 (3) (2008) 258–275.

[39] M. Giacobbe, C. Guet, A. Gupta, T. Henzinger, T. Paixao, T. Petrov, Model checking gene regulatory networks, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2015, pp. 469–483.

[40] G. Bernot, J.-P. Comet, C. Risso-de Faverney, Regulatory networks, in: B. Reisfeld, A. Mayeno (Eds.), Computational Toxicology, Vol. II, Humana Press, ISBN 978-1-62703-058-8, USA, 2013, pp. 215–234.

[41] H. Klarner, A. Streck, D. Šafránek, J. Kolčák, H. Siebert, Parameter identification and model ranking of Thomas networks, in: Computational Methods in Systems Biology, Springer, 2012, pp. 207–226.

[42] M. Reynolds, A traditional tree-style tableau for LTL, CoRR abs/1604.03962.
URL `http://arxiv.org/abs/1604.03962`

[43] M. Bertello, N. Gigante, A. Montanari, M. Reynolds, Leviathan: A new LTL satisfiability checking tool based on a one-pass tree-shaped tableau., in: IJCAI, 2016, pp. 950–956.

[44] D. Peleg, Concurrent dynamic logic, Journal of the ACM (JACM) 34 (2) (1987) 450–479.

[45] M. Hennessy, Algebraic Theory of Processes, MIT Press, 1988.

Supplementary materials

Appendix A. Semantics of Hoare triples for gene networks

We define the semantics of a trace specification *via* a binary relation between states and *sets* of states. This relation characterises all the possible realisations of the trace specification. The general ideas that motivate our definition are the following:

- Starting from an initial state η, a trace specification without existential or universal quantifier is either realised by associating with η another state η', or is not realisable and η' does not exist. For example, the atomic expression $v+$ associates η' with η (where $\forall u \neq v, \eta'(u) = \eta(u)$ and $\eta'(v) = \eta(v) + 1$) if and only if the transition $\eta \to \eta'$ exists in the state space. If, on the contrary, this transition does not exist, the trace specification is not realisable.

- Existential quantifiers open a sort of space of possibilities for η': According to the chosen trace specification under each existential quantifier one may get different associated states. Consequently, one cannot define the semantics as a partial function that associates a unique η' with η; a binary relation is a more suited mathematical object (denoted $\rightsquigarrow$ in the sequel).

- A universal quantifier induces a sort of unity/solidarity between all the states η' that can be obtained through each trace specification under its scope. All these states have to satisfy the postcondition (Definition Appendix A.2) . For this reason, we define a binary relation that associates a *set of states* E with the initial state η: "$\eta \rightsquigarrow E$". Such a set E can be understood as grouping together the states it contains in preparation for checking the forthcoming post condition.

- When the trace specification p contains both existential and universal quantifiers, we may consequently get several sets $E_1, \cdots, E_n$ such that $\eta \overset{p}{\rightsquigarrow} E_i$, each of the E_i being a *possibility* through the existential quantifiers of p and all the states belonging to a given E_i being together through the universal quantifiers of p. On the contrary, if p is not realisable, then there is no set E such that $\eta \overset{p}{\rightsquigarrow} E$ (not even the empty set).

Definition Appendix A.1. (Mathematical semantics of a trace specification). *Let $N = (V, M, E_V, E_M, \mathcal{K})$ be a* GRN*, let $\mathcal{S}$ be the state graph of N whose set of vertices is denoted S and let p be a trace specification for N. The binary relation $\overset{p}{\rightsquigarrow}$ is the smallest subset of $S \times \mathcal{P}(S)$ such that, for any state η:*

1. *If p is the atomic expression $v+$, then let us consider the state $\eta' = \eta[v \leftarrow (\eta(v) + 1)]$: If $\eta \rightarrow \eta'$ is a transition of $\mathcal{S}$ then $\eta \overset{p}{\rightsquigarrow} \{\eta'\}$.*
2. *If p is the atomic expression $v-$, then let us consider the state $\eta' = \eta[v \leftarrow (\eta(v) - 1)]$: If $\eta \rightarrow \eta'$ is a transition of $\mathcal{S}$ then $\eta \overset{p}{\rightsquigarrow} \{\eta'\}$.*
3. *If p is the atomic expression $v := i$, then $\eta \overset{p}{\rightsquigarrow} \{\eta[v \leftarrow i]\}$.*
4. *If p is of the form $assert(e)$, if $\eta \models_N e$, then $\eta \overset{p}{\rightsquigarrow} \{\eta\}$.*
5. *If p is of the form $\forall(p_1, p_2)$: If $\eta \overset{p_1}{\rightsquigarrow} E_1$ and $\eta \overset{p_2}{\rightsquigarrow} E_2$ then $\eta \overset{p}{\rightsquigarrow} (E_1 \cup E_2)$.*
6. *If p is of the form $\exists(p_1, p_2)$: If $\eta \overset{p_1}{\rightsquigarrow} E_1$ then $\eta \overset{p}{\rightsquigarrow} E_1$,* and *if $\eta \overset{p_2}{\rightsquigarrow} E_2$ then $\eta \overset{p}{\rightsquigarrow} E_2$.*
7. *If p is of the form $(p_1; p_2)$: If $\eta \overset{p_1}{\rightsquigarrow} F$ and if $\{E_e\}_{e \in F}$ is a F-indexed family of state sets such that $e \overset{p_2}{\rightsquigarrow} E_e$, then $\eta \overset{p}{\rightsquigarrow} (\bigcup_{e \in F} E_e)$.*
8. *If p is of the form (while e with I do p_0):*
   - *If $\eta \not\models_N e$ then $\eta \overset{p}{\rightsquigarrow} \{\eta\}$.*
   - *If $\eta \models_N e$ and $\eta \overset{p_0;p}{\rightsquigarrow} E$ then $\eta \overset{p}{\rightsquigarrow} E$.*

This definition calls for several comments.

The relation $\overset{p}{\rightsquigarrow}$ exists because (i) the set of all relations that satisfy the properties 1–8 of the definition is not empty (the relation which links all states to all sets of states satisfies the properties) and (ii) the intersection of all the relations that satisfy the properties 1–8, also satisfies the properties.

A simple atomic expression such as $v+$ may be not realisable in a state η (if $\eta \rightarrow \eta'$ is not a transition of $\mathcal{S}$). In this case, there is no set E such that $\eta \overset{v+}{\rightsquigarrow} E$. The same situation happens when the trace specification is an assertion that is not satisfied at the current state η.

Universal quantifiers propagate non-realisable trace specifications: If one of the p_i is not realisable then $\forall(p_1, \cdots, p_n)$ is not realisable. *It is not the case for existential quantifiers*: If $\eta \overset{p_i}{\rightsquigarrow} E_i$ for one of the p_i then $\eta \overset{\exists(p_1 \cdots p_n)}{\rightsquigarrow} E_i$ even if one of the p_j is not realisable.

When a *while* loop does not terminate, there is no set E such that $\eta \overset{while...}{\rightsquigarrow} E$. This is due to the minimality of the binary relation $\overset{p}{\rightsquigarrow}$. On

the contrary, when the *while* loop terminates, it is equivalent to a trace specification containing a finite number of occurrences of the sub-trace p_0 in sequence, starting from η.

The semantics of sequential composition may seem unclear for readers not familiar with commutations of quantifiers. We give an example to explain the construction of $\overset{p_1;p_2}{\rightsquigarrow}$ (see Figure A.5):

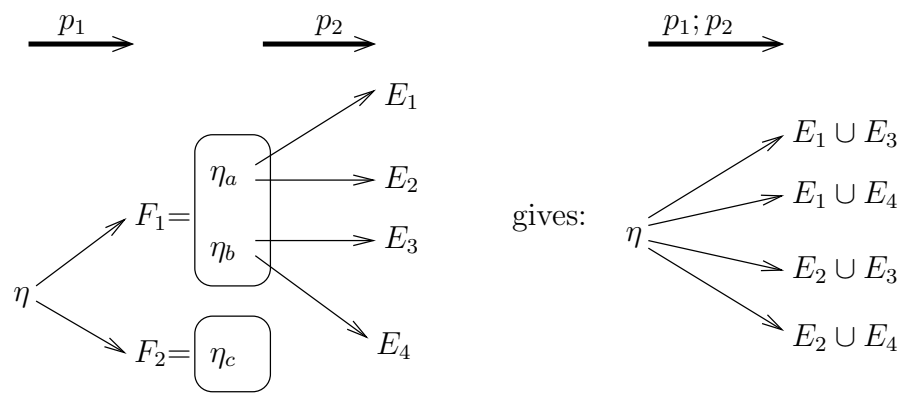

Figure A.5: An example for the semantics of sequential composition

- Let us assume that starting from state η, two sets of states are possible *via* p_1: $\eta \overset{p_1}{\rightsquigarrow} F_1 = \{\eta_a, \eta_b\}$ and $\eta \overset{p_1}{\rightsquigarrow} F_2 = \{\eta_c\}$. It intuitively means that p_1 permits a choice between F_1 and F_2 through some existential quantifier and that the trace specification leading to F_1 contains a universal quantifier grouping together η_a and η_b.

- Let us also assume that

  - starting from the state η_a, two sets of states are possible *via* p_2: $\eta_a \overset{p_2}{\rightsquigarrow} E_1$ and $\eta_a \overset{p_2}{\rightsquigarrow} E_2$,
  - starting from the state η_b, two sets of states are possible *via* p_2: $\eta_b \overset{p_2}{\rightsquigarrow} E_3$ and $\eta_b \overset{p_2}{\rightsquigarrow} E_4$,
  - and there are no set E such that $\eta_c \overset{p_2}{\rightsquigarrow} E$.

When focusing on the traces of $(p_1; p_2)$ that encounter F_1 after p_1, the traces such that p_1 leads to η_a must be grouped together with the ones that lead to η_b. Nevertheless, for each of them, p_2 permits a choice of possibilities:

between E_1 or E_2 for η_a and between E_3 or E_4 for η_b. Consequently, when grouping together the possible futures of η_a and η_b, one needs to consider the four possible combinations: $\eta \overset{p_1;p_2}{\rightsquigarrow} (E_1 \cup E_3)$, $\eta \overset{p_1;p_2}{\rightsquigarrow} (E_1 \cup E_4)$ $\eta \overset{p_1;p_2}{\rightsquigarrow} (E_2 \cup E_3)$ and $\eta \overset{p_1;p_2}{\rightsquigarrow} (E_2 \cup E_4)$.
Lastly, when focusing on the traces of $(p_1; p_2)$ that encounter F_2 after p_1, since η_c has no future *via* p_2, there is no family indexed by F_2 as mentioned in the definition and consequently it adds no relation into $\overset{p_1;p_2}{\rightsquigarrow}$.

Let us remark that, if $\eta \overset{p}{\rightsquigarrow} E$ then E cannot be empty; it always contains at least one state. The proof is easy by structural induction of the trace specification p (using the fact that a *while* loop which terminates is equivalent to a trace specification containing a finite number of occurrences of the sub-trace p_0).

Definition Appendix A.2. (Semantics of a Hoare triple). *Given a* GRN $N = (V, M, E_V, E_M, \mathcal{K})$*, let* $\mathcal{S}$ *be the state graph of* N *whose set of vertices is denoted* S*. A Hoare triple* $\{P\}\ p\ \{Q\}$ *is satisfied if and only if:*

For all $\eta \in S$ *satisfying* P*, there exists* E *such that* $\eta \overset{p}{\rightsquigarrow} E$ *and for all* $\eta' \in E$*,* η' *satisfies* Q*.*

The previous definition implies the consistency of the trace specification p with the state graph: If the specification p is not realisable starting from one of the states satisfying pre-condition P, then the Hoare triple cannot be satisfied. For instance if some $v+$ is required by the trace specification p but the increasing of v is not possible according to the state graph, then the Hoare triple is not satisfied.
As an example, let us consider the GRN in Figure A.6 and its state graph.

1. The Hoare triple $\{(a = 0) \wedge (b = 0)\}\ a+; a+; b+\ \{(a = 2) \wedge (b = 1)\}$ is satisfied, because
   - for all states that do not satisfy the pre-condition, the Hoare triple is satisfied by definition,
   - there is, in this example, a unique state satisfying the precondition $(a = 0) \wedge (b = 0)$ and from *this* state, the trace specification $a+; a+; b+$ is possible and leads to the state $(2, 1)$ and
   - the state $(2, 1)$ satisfies the postcondition $(a = 2) \wedge (b = 1)$.
2. The Hoare triple $\{(a = 2) \wedge (b = 0)\}\ b+; a-; a-\ \{(a = 0) \wedge (b = 1)\}$ is not satisfied because from the state satisfying the precondition, the first

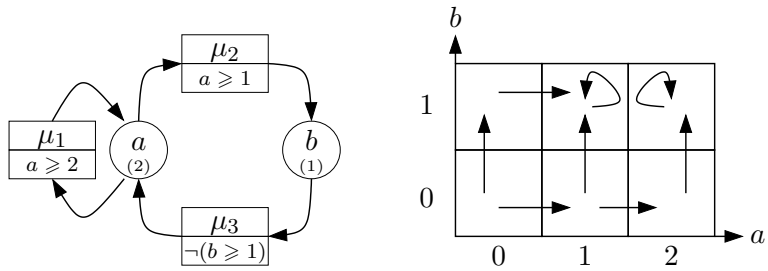

Figure A.6: **(Left)** The graphical representation of the GRN $N = (V, M, E_V, E_M, \mathcal{K})$ with $V = \{a, b\}$, the bounds of a and b are respectively 2 and 1, $M = \{\mu_1, \mu_2, \mu_3\}$, φ_{μ_1} is $(a \geqslant 2)$, φ_{μ_2} is $(a \geqslant 1)$, φ_{μ_3} is $\neg(b \geqslant 1)$. Finally the family of integers is $\{K_a = 1, K_{a,\mu_1} = 2, K_{a,\mu_3} = 2, K_{a,\mu_1\mu_3} = 2, K_b = 1, K_{b,\mu_2} = 1\}$. **(Right)** Representation of its state graph.

expression $b+$ is realisable and necessarily leads to the state $(2, 1)$ from which the next expression $a-$ is not consistent with the state graph.

3. The following Hoare triple contains two existential quantifiers and a universal one:
$\{(a = 0) \wedge (b = 0)\}\ \forall(a+, b+); \exists(a+, b+); \exists(\varepsilon, b+)\ \{(b = 1)\}$ (remember that ε denotes the empty trace and is an abbreviation for $assert(\top)$ where $\top$ stands for a tautology).

   - We have clearly $(0, 0) \overset{\forall(a+,b+)}{\rightsquigarrow} \{(1, 0), (0, 1)\}$
   - Since we have $(1, 0) \overset{\exists(a+,b+)}{\rightsquigarrow} \{(2, 0)\}$ and $(1, 0) \overset{\exists(a+,b+)}{\rightsquigarrow} \{(1, 1)\}$ and $(0, 1) \overset{\exists(a+,b+)}{\rightsquigarrow} \{(1, 1)\}$, we deduce $(0, 0) \overset{\forall(a+,b+);\exists(a+,b+)}{\rightsquigarrow} \{(1, 1), (2, 0)\}$ and $(0, 0) \overset{\forall(a+,b+);\exists(a+,b+)}{\rightsquigarrow} \{(1, 1)\}$.
   - We have trivially $(1, 1) \overset{\exists(\varepsilon,b+)}{\rightsquigarrow} \{(1, 1)\}$
   - Moreover we have both $(2, 0) \overset{\exists(\varepsilon,b+)}{\rightsquigarrow} \{(2, 0)\}$ and $(2, 0) \overset{\exists(\varepsilon,b+)}{\rightsquigarrow} \{(2, 1)\}$
   - We deduce that the considered trace specification p can lead to 3 different sets of states: $(0, 0) \overset{p}{\rightsquigarrow} \{(1, 1), (2, 0)\}$, $(0, 0) \overset{p}{\rightsquigarrow} \{(1, 1)\}$ and $(0, 0) \overset{p}{\rightsquigarrow} \{(1, 1), (2, 1)\}$.

   Because the postcondition is satisfied in both states $(1, 1)$ and $(2, 1)$, the two last sets of states which are in relation with $(0, 0)$, satisfy the postcondition. Consequently although the first set does not, one can deduce that the Hoare triple is satisfied.

Appendix B. Soundness and Completeness

As usual in Hoare logic, The soundness and completeness of the logic can only ensure a *partial* correctness of the Hoare triples because the *while* loops of the trace specifications do not necessarily terminate.

Appendix B.1. Soundness

The soundness of our modified Hoare logic means that: Given a network $N = (V, M, E_V, E_M, \mathcal{K})$, if $\vdash \{P\}\ p\ \{Q\}$ according to the inference rules of Section 5 (and after substituting the symbols $K_{...}$ by their value in N), then for all states η that satisfies P, if there is a set E such that $\eta \overset{p}{\rightsquigarrow} E$, then there is at least a set E' such that $\eta \overset{p}{\rightsquigarrow} E'$ and $\forall \eta' \in E', \eta' \models_N Q$.

The proof is made as usual by induction on the proof tree of $\vdash \{P\}\ p\ \{Q\}$. Hence, we have to prove that each rule of Section 5 is sound. Here we develop only the *Increase* rule and the *Sequential composition* rule since the soundness of the other inference rules is either similar (*Decrease* rule), trivial (*Assert* rule, *Quantifier rules*, *Assignment* rule, *Empty trace* rule and *Boundary axioms*) or standard in Hoare logic (*Iteration* rule). Let us note that the soundness of the *Sequential composition* rule is not trivial because its semantics is enriched to cope with the quantifiers.

Let η be any state of N.

Increase rule: $\dfrac{}{\{\ \Phi_v^+ \ \wedge\ Q[v{\leftarrow}v{+}1]\ \}\ v{+}\ \{Q\}}$ (where v is a variable of N)

From Definition Appendix A.2, the hypothesis is

$\boxed{H}$ $\eta \models_N \Phi_v^+$ and $\eta \models_N Q[v \leftarrow v+1]$

and we have to prove the conclusion

$\boxed{C}$ there exists $E \subset S$ such that $\eta \overset{v+}{\rightsquigarrow} E$ and $\forall \eta' \in E,\ \eta' \models_N Q$

Let us choose $E = \{\eta'\}$ with $\eta' = \eta[v \leftarrow \eta(v) + 1]$. From Notation 5.1, the hypothesis $\eta \models_N \Phi_v^+$ is equivalent to $(\eta \rightarrow \eta') \in \mathcal{S}$, which in turn, according to Definition 4.4, implies $\eta \overset{v+}{\rightsquigarrow} \{\eta'\}$. Hence, it only remains to prove that $\eta' \models_N Q$, which results from the hypothesis $\eta \models_N Q[v \leftarrow v+1]$. □

Sequential composition rule: $\dfrac{\{P_1\}\ p_1\ \{P_2\} \qquad \{P_2\}\ p_2\ \{Q\}}{\{P_1\}\ p_1;p_2\ \{Q\}}$

From Definition Appendix A.2, we consider the following three hypotheses:

$\boxed{H_1}$ for all $\eta_1 \in S$ such that $\eta_1 \models_N P_1$ there exists E_1 such that $\eta_1 \overset{p_1}{\rightsquigarrow} E_1$ and $\forall \eta' \in E_1,\ \eta' \models_N P_2$

$\boxed{H_2}$ for all $\eta_2 \in S$ such that $\eta_2 \models_N P_2$ there exists E_2 such that $\eta_2 \overset{p_2}{\rightsquigarrow} E_2$ and $\forall \eta'' \in E_2,\ \eta'' \models_N Q$

$\boxed{H_3}$ $\eta \models_N P_1$

and we have to prove the conclusion:

$\boxed{C}$ there exists $E \subset S$ such that $\eta \overset{p_1;p_2}{\rightsquigarrow} E$ and $\forall \eta'' \in E,\ \eta'' \models_N Q$

Let us arbitrarily choose a set E_1 such that $\eta \overset{p_1}{\rightsquigarrow} E_1$ and $\forall \eta' \in E_1,\ \eta' \models_N P_2$ (we know that E_1 exists from $\boxed{H_1}$ and $\boxed{H_3}$).

For each $\eta' \in E_1$, we similarly choose a set $E_2^{\eta'}$ such that:
$\eta' \overset{p_2}{\rightsquigarrow} E_2^{\eta'}$ and $\forall \eta'' \in E_2^{\eta'},\ \eta'' \models_N Q$ (we know that the family $\{E_2^{\eta'}\}_{\eta' \in E_1}$ exists from $\boxed{H_2}$ and the fact that $\eta' \models_N P_2$ for all $\eta' \in E_1$)

Let $E = (\bigcup_{\eta' \in E_1} E_2^{\eta'})$, we have: $\eta \overset{p_1;p_2}{\rightsquigarrow} E$ from Definition 4.4 and $\forall \eta'' \in E,\ \eta'' \models_N Q$ (from the way the union is built). □

Appendix B.2. Completeness and weakest precondition

Completeness of Hoare logic is defined as follows. Given a network $N = (V, M, E_V, E_M, \mathcal{K})$, if the Hoare triple $\{P\}\ p\ \{Q\}$ is satisfied in N (according to Definition Appendix A.2) then $\vdash \{P\}\ p\ \{Q\}$ (using the inference rules of Section 5 and after substituting the symbols $K_{\ldots}$ by their value in N). We prove the completeness by establishing that one can compute the weakest invariants of all *while* loops and that the backward strategy gives a proof of $\{P\}\ p\ \{Q\}$.

The main difference with respect to the classical completeness proof is that we navigate into a finite state space, so that we will not have to care about the incompleteness of arithmetic or restrictions about weakest loop invariants. In the following proposition, we see that one can compute the weakest invariant for each *while* occurrence in the trace specification. Only

practical reasons in order to facilitate proofs justify to ask the specifier to include loop invariants into trace specifications: Often, a slightly non minimal invariant considerably simplifies the proof tree.

Proposition Appendix B.1. (Existence of the weakest loop invariant). *Given a* GRN $N = (V, M, E_V, E_M, \mathcal{K})$*, let us consider two assertions* Q *and* e*, and a trace specification* p*. There exists a weakest loop invariant* I *such that the Hoare triple* $\{I\}$ *while* e *with* I *do* p $\{Q\}$ *is partially correct.*

The following proof is <u>constructive</u> and gives a way to compute I (see remark Appendix B.4).
Proof:

1. In the first step of the proof, we build a set $\mathcal{D}$ as a countable union.
   - Let $q_0 = \{\eta \in S \mid \eta \models_N Q \wedge \neg e\}$ be the set of all states that satisfy Q without entering the *while* loop.
   - given q_i, let $q_{i+1} = \{\eta \in S \mid \eta \models_N e$ and $\exists E \subset S, \eta \overset{p}{\rightsquigarrow} E$ and $E \subset q_i\}$. From Definition Appendix A.2, for each i, q_i is the set of states that induce exactly i *while* loops and such that the resulting states satisfy Q.
   - Let $\mathcal{D}_n = \bigcup_{i=0}^{n} q_i$. The sequence of $\mathcal{D}_n$ is increasing and because S is finite, it is stationary. So $\mathcal{D} = \bigcup_{i=0}^{\infty} q_i$ exists and can be inductively computed.
2. In the second step of the proof, we show that the characteristic formula of $\mathcal{D}$ is a loop invariant.
   - Because $\mathcal{D}$ is finite, there is a formula I such that $\eta \models_N I$ iff $\eta \in \mathcal{D}$: $I \equiv \bigvee_{\eta \in \mathcal{D}} \mathbb{1}_\eta$ where $\mathbb{1}_\eta \equiv \bigwedge_{v \in V} v = \eta(v)$
   - I is a loop invariant because for each state η that satisfies I, there is an integer i such that $\eta \in q_i$.
     - If $i > 0$, then η satisfies $I \wedge e$ and by definition, there is a set E such that $\eta \overset{p}{\rightsquigarrow} E$ and $E \subset q_{i-1}$, consequently E satisfies I because every state of q_{i-1} satisfies I.
     - If $i = 0$, then $\eta \models_N \neg e$, thus $\eta \not\models_N e \wedge I$, which implies that $\{e \wedge I\}$ p $\{I\}$ is satisfied for η, according to Definition Appendix A.2 and elementary truth tables.

3. In the last step of the proof, we show that each state of $\mathcal{D}$ satisfies any minimal loop invariant.
   - Let J be a minimal loop invariant. Assume that there is a state $\eta \in \mathcal{D}$ that does not satisfy J. Then $J \vee \mathbb{1}_\eta$ (where $\mathbb{1}_\eta$ is the formula characterizing the state η), is strictly weaker than J. But it is also an invariant since after i iterations of the *while* loop from η, one of the resulting sets of states E satisfies Q. This contradicts the minimality of J.
   - Consequently I is the weakest loop invariant. □

Theorem Appendix B.2. (Completeness theorem on the genetically modified Hoare logic). *Given a* GRN *N, a trace specification p and a post-condition Q, the backward strategy defined at the end of Section 2, with the inference rules of Section 5, computes after steps 1 and 2 the* weakest precondition *P_0 such that $\{P_0\}\ p\ \{Q\}$ is satisfied. In other words, for any assertion P, if $\{P\}\ p\ \{Q\}$ is satisfied, then $P \Rightarrow P_0$ is satisfied (that is, the third step of the backward strategy).*

This theorem has an obvious corollary.

Corollary Appendix B.3. *Given a* GRN *N, our modified Hoare logic is complete.*

Proof of the corollary: if $\{P\}\ p\ \{Q\}$ is satisfied, then, from the theorem above, there is a proof tree that infers the Hoare triple if there is a proof tree for the property $P \Rightarrow P_0$ (which is semantically satisfied because P_0 is the weakest precondition). First order logic being complete and the number of possible substitutions being finite (the state space being finite), the proof tree for $P \Rightarrow P_0$ exists. □

Proof of the completeness theorem:
Under the following two hypotheses

$\boxed{H_1}$ the Hoare triple $\{P\}\ p\ \{Q\}$ is satisfied, i.e., for all η satisfying P, there exists E such that $\eta \overset{p}{\rightsquigarrow} E$ and for all $\eta' \in E$, η' satisfies Q,

$\boxed{H_2}$ for all *while* statements of p, the corresponding loop invariant I is the weakest one (Proposition Appendix B.1),

one has to prove the conclusion:

$\boxed{C}$ $P \Rightarrow P_0$ is satisfied, where P_0 is the precondition computed from p and Q by the steps 1 and 2 of the backward strategy with the inference rules of Section 5.

The proof is done by structural induction according to the backward strategy on p.

- If p is of the form $v+$, then the only set E such that $\eta \overset{v+}{\rightsquigarrow} E$ is $E = \{\eta[v \leftarrow v+1]\}$. The hypothesis $\boxed{H_1}$ becomes:

  $\boxed{H_1}$ for all η satisfying P, $\eta' = \eta[v \leftarrow v+1]$ satisfies Q and $\eta \rightarrow \eta'$ is a transition of $\mathcal{S}$

  and from the *Increase* rule, the conclusion becomes:

  $\boxed{C}$ $P \Rightarrow (\Phi_v^+ \wedge Q[v \leftarrow v+1])$ is satisfied.

  So, $\boxed{H_1} \Rightarrow \boxed{C}$ straightforwardly results from the definition of Φ_{v^+} (Notation 5.1) and we do not use $\boxed{H_2}$.

- If p is of the form $p_1; p_2$, then we firstly inherit the two structural induction hypotheses:

  $\boxed{H_3}$ for all assertions P' and Q', if $\{P'\}\ p_1\ \{Q'\}$ is satisfied then $P' \Rightarrow P_1$ is satisfied, where P_1 is the precondition computed from Q' *via* the backward strategy

  $\boxed{H_4}$ for all assertions P'' and Q'', if $\{P''\}\ p_2\ \{Q''\}$ is satisfied then $P'' \Rightarrow P_2$ is satisfied, where P_2 is the precondition computed from Q'' *via* the backward strategy

  Moreover the hypothesis $\boxed{H_1}$ becomes (Definition 4.4):

  $\boxed{H_1}$ for all η satisfying P, there exists a family of state sets $\mathcal{F} = \{E_e\}_{e \in F}$ such that $\eta \overset{p_1}{\rightsquigarrow} F$ and $e \overset{p_2}{\rightsquigarrow} E_e$ for all $e \in F$ and for all $\eta' \in E = (\bigcup_{e \in F} E_e)$, η' satisfies Q

  Lastly, from the *Sequential composition* rule, the conclusion becomes:

  $\boxed{C}$ $P \Rightarrow P_1$ is satisfied, where P_1 is the weakest precondition of $\{\cdots\}\ p_1\ \{P_2\}$, P_2 being the weakest precondition of $\{\cdots\}\ p_2\ \{Q\}$.

From $\boxed{H_4}$ (with $Q'' = Q$) it results that all the states $e \in F$ of hypothesis $\boxed{H_1}$ satisfy P_2. Consequently $\{P\}\ p_1\ \{P_2\}$ is satisfied. Thus, from $\boxed{H_3}$ (with $Q' = P_2$ and $P' = P$) it comes $P \Rightarrow P_1$, which proves the conclusion.

- If p is of the form *while e with I do* p', then, by construction of the backward strategy, applying the *Iteration* rule, we get $P_0 = I$, and the conclusion results immediately from $\boxed{H_2}$.

- Similarly to the soundness proof, we do not develop here the other cases of the structural induction. They are either similar to already developed cases (*Decrease* rule) or trivial (*Assert* rule, *Quantifier rules*, and *Assignment* rule).

This ends the proof. □

Remark Appendix B.4. *Soundness and completeness being now established, one can extend Proposition Appendix B.1 by giving a purely symbolic computation of the weakest loop invariant I of a while loop. Following the notations of the proof of Proposition Appendix B.1:*

- *The set of states q_0 is characterised by the formula $Q_0 \equiv \neg e \wedge Q$,*

- *In addition,* assuming that the trace specification p terminates*, the set of states q_{i+1} is inductively characterised by the weakest precondition Q_{i+1} obtained via the backward strategy of the proof of $\{Q_{i+1}\}\ p\ \{Q_i\}$ (this is due to the soundness and completeness of our calculus).*

- *From this construction, we deduce that the first integer n such that $q_{n+1} \subset \mathcal{D}_n$ (where $\mathcal{D}_n = \bigcup_{i=0}^{n} q_i$) is the first n such that $Q_{n+1} \Rightarrow \bigvee_{i=0}^{n} Q_i$. This implication is decidable because the set of possible substitutions is finite.*

Proposition Appendix B.1 implies that the integer n mentioned before exists. Consequently $I = \bigvee_{i=0}^{n} Q_i$ can be expressed in a purely symbolic way. And more importantly, this can be done from the solely knowledge *of the interaction graph. The assertion I is then a constraint on states and parameters $K_{\ldots}$, what we used in Section 6.*