Identification of Dynamic Parameters for Gene Networks

Jonathan Behaegel
Université Côte d'Azur, CNRS, I3S
Nice, France
behaegel@i3s.unice.fr

Jean-Paul Comet
Université Côte d'Azur, CNRS, I3S
Nice, France
comet@unice.fr

Marie Pelleau
Université Côte d'Azur, CNRS, I3S
Nice, France
marie.pelleau@i3s.unice.fr

***Abstract*—The study of gene networks allows us to better understand some biological processes such as the adaptation of the organism to a disturbance of the environment. In a discrete modelling framework of gene networks, it has been shown that the Hoare logic can help the modeller to identify the parameters of the model, so that the latter exhibits the observed biological traces. In this paper we present a hybrid modelling of gene networks which pays particular attention to the time spent in each state and we introduce an extension of the Hoare logic in this hybrid framework. The weakest precondition calculus associated with this modified Hoare logic makes it possible to determine the minimal constraints on the dynamic parameters of a gene network from an observed biological trace. These constraints form a continuous CSP that can be solved using the AbSolute continuous solver. The first experimental results show that the obtained solutions are in agreement with the specification of the Hoare triple coming from biological expertise.**

***Index Terms*—Genetic networks, Hoare logic, Continuous constraints.**

I. Introduction

The purpose of genetic regulatory network modelling is to study and understand the molecular mechanisms that enable the body to perform essential functions ranging from metabolism to environmental disturbance adaptation. Gene networks are described by rules that can be classified into two types: activations and inhibitions. The combination of these regulations allows many behaviors and it can be shown that the complexity of these systems comes from the positive and negative feedbacks commonly observed which lead to multistationnarity and homeostasis (ability to maintain a balance). The study of the dynamics of these systems opens new perspectives with applications in pharmacology, medicine, or toxicology. Different modelling frameworks (qualitative, continuous, stochastic, hybrid) are possible but regardless of the chosen framework, a crucial point of the modelling process relies on the determination of the parameters that govern the dynamics of the model and their identification remains the limiting step.

René Thomas' discrete modelling framework [17] only seeks to represent the sequence of qualitative events, and formal methods (model checking, constraint programming, Hoare logic) have been successfully used to automate the identification of the parameters governing the dynamics [4], [7], [3]. However, some biological phenomena involve a temporal component playing a primordial role: For example, the circadian cycle allows the body to adapt to day/night alternation. To model such timed phenomena requires *at least* a hybrid modelling framework that adds to René Thomas' approach a measure of the time spent in each of the states. The parameter identification step remains the bottleneck of the modelling process but one can seek in such a hybrid framework for an automation of this step to build a model in agreement with the experimental observations (variations of protein concentration of a biological system during the day which can be very hard to obtain for many proteins). These observations are represented by a biological trace which expresses an irregularly spaced time series of qualitative events. For this, we modified the Hoare logic as well as its associated weakest precondition calculus to build a constraints system solved by the solver AbSolute [14].

Constraint programming has already been used to solve problems related to biological networks. In the discrete approach, it has been used to determine kinetic parameters [7], or bifurcations [10]. It has also been used for the optimization of flow modes in metabolic networks [13]. In the continuous framework, the constraints approach has been used to verify temporal properties of the model [5], or to identify the dynamic parameters of a differential system in the context of the synthetic biology [15]. Our goal is here to address, for hybrid models, the parameter identification problem whereas addressed problems are generally some verification problems [1], [12] or model approximation ones [11].

This paper is organized as follows. Section II presents the hybrid modelling framework allowing the description of the dynamics of gene networks. Section III introduces the Hoare logic and its associated weakest precondition calculus adapted for this hybrid modelling framework. The AbSolute solver is sketched in Section IV with an emphasis on symbolic rewriting and the notion of a sure solution. Section V is dedicated to the presentation of our experimental results. Finally, section VI is devoted to conclusion.

II. Modelling a Gene Network

Many modelling frameworks dedicated to gene networks can be used to represent their dynamics. One of the first modelling frameworks that has been fully formalized was introduced by René Thomas [17], [4] but the time separating

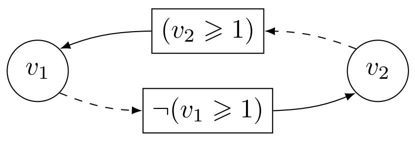


Fig. 1. Interaction graph representing a simple negative feedback loop.

two qualitative events has been totally abstracted. Here, we sketch the hybrid modelling framework based on the Thomas' discrete approach presented in [2].

A. Representation of a Gene Network

The gene networks are represented by labelled graphs, see Figure 1. The two types of vertices allow the description on the one hand of the entities (abstraction of genes and their associated proteins, represented by circles) and on the other hand of the regulations between these entities (represented by rectangles) called multiplexes. The entities, together, form a finite set denoted V. Arcs pointing to a multiplex (dashed arcs) specify the entities involved in the regulation and all predecessors of an entity $v \in V$ are denoted $R^-(v)$. Arcs coming out of a multiplex (solid arcs) specify the entities that are *positively* regulated by the multiplex. Since the regulations are not always active, each multiplex is associated with a formula describing the conditions under which the regulation associated with the multiplex takes place.

Figure 1 shows an interaction graph with 2 entities v_1 and v_2: v_2 activates v_1 above the threshold 1 and v_1 inhibits v_2 under the threshold 1 (the inhibition is represented by the negation of the formula). This graph forms a negative feedback loop since each of both entities has an indirect negative action on itself.

B. Dynamics of a Gene Network

The hybrid modelling framework [2] relies on the discretization of the concentrations of the biological entities, while including the time spent between two successive events. Let us suppose that the entity v acts on several entities at different thresholds and that the total number of different thresholds is b_v. Then the ordering of these thresholds leads to the definition of the qualitative levels of the entity v. Level 0 represents the interval of concentrations lower than the first threshold, level b_v represents the range of concentrations higher than the largest threshold, and the level k represents the range of concentrations between the k-th and the $(k+1)$-th threshold ($k \in]\!]0, b_v[\![$). A qualitative state of the system is a set of data giving a qualitative level to each entity.

With each qualitative state is associated a "temporal space" which can be seen as the hypercube $[0,1]^n$ where n is the number of entities, see the squares of the Figure 2. The precise position of the system at a particular time is thus characterized not only by the qualitative state, but also by the precise position within the temporal space. These states are thus called hybrid states and are defined by a pair of vectors $h = (\eta, \pi)$ where $\eta \in ([\![0, b_{v_1}]\!] \times \cdots \times [\![0, b_{v_n}]\!])$,

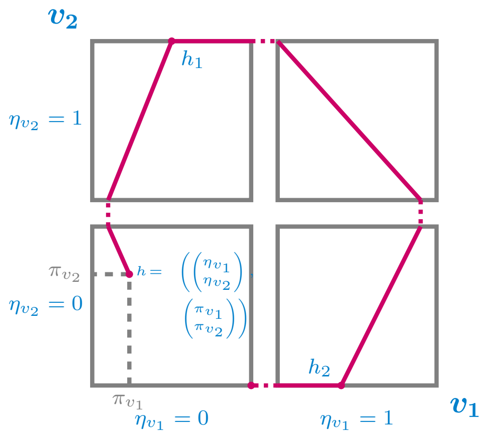


Fig. 2. A possible dynamics associated with the interaction graph of Figure 1.

called discrete state, is the vector of the qualitative levels of entities and $\pi \in [0,1]^n$, called fractional part, is the vector of the exact positions inside the temporal space associated with this discrete state, see Figure 2. By convention, η_v (resp. π_v) denotes the qualitative level (or the fractional part) of entity v. Two discrete states are called neighbours if they differ only by one component and if this component differs only by one.

The evolution within each qualitative state is represented by *continuous transitions* which are governed by celerities to identify, and transitions from one discrete state to another, called *discrete transitions*, are only possible under certain conditions.

Figure 2 shows a possible evolution of a model associated with the interaction graph of Figure 1 starting from the point h inside the discrete state $(\eta_{v_1}, \eta_{v_2}) = (0,0)$. In this discrete state, the concentration of v_2 increases. When v_2 cannot increase anymore, the discrete state changes $(\eta_{v_1}, \eta_{v_2}) = (0,1)$. In this new state, v_1 and v_2 increase until v_2 is saturated. v_1 then keeps on increasing, and the discrete state $(1,1)$ is reached... In Figure 2 the trace is represented only until its entering into the discrete state (0, 0) where $\pi_{v_1} = 1$ and $\pi_{v_2} = 0$ but it could be extended in a similar way.

A multiplex is active at a hybrid state h if its associated formula is evaluated to true after substitution of the entities by their qualitative levels. The active multiplexes are called *resources* and the set of resources of v at state h is denoted $\rho(h, v)$. For example, in Figure 2 we have $\rho(h, v_1) = \emptyset$ because the formula of the unique multiplex regulating v_1 is evaluated to false at this state, and $\rho(h, v_2) = \{v_1\}$ because the formula of the unique multiplex regulating v_2 is evaluated to true at this state. Note that the resources of a hybrid state h depend only on the qualitative levels of the entities: the resources do not vary within a "temporal space".

The different entities do not evolve at the same speed in each of the "temporal spaces". Similarly, the speed of an entity depends on its resources and on its qualitative level. The dynamics of the entities are thus controlled by celerities $C_{v,\omega,k} \in \mathbb{R}$, indexed by entity v, ω the set of resources of

v in the current discrete state, and by the qualitative level k of entity v. For example, in the discrete state $(0, 0)$ of Figure 2, the celerity controlling the entity v_1 is $C_{v_1,\emptyset,0}$ and the celerity which controls the entity v_2 is $C_{v_2,\{v_1\},0}$. A null celerity indicates that the entity has reached an equilibrium. To reach this equilibrium, it is mandatory that the celerities of the same entity having the same regulations head towards this equilibrium, leading to the following constraints:

$$\begin{array}{l} \forall v \in V,\ \forall \omega \subset R^-(v),\ \forall n \in [\![0,\, b_v]\!], \\ C_{v,\omega,n} = 0 \Rightarrow \left\{ \begin{array}{ll} \forall i \in [\![n+1,\, b_v]\!] & C_{v,\omega,i} < 0 \\ \forall i \in [\![0,\, n-1]\!] & C_{v,\omega,i} > 0 \end{array} \right. \end{array} \quad (1)$$

In all cases, the celerities of the same entity with the same resources in neighbouring discrete states move in the same direction and therefore have the same sign. This is transcribed into the following constraints:

$$\begin{array}{l} \forall v \in V,\ \forall \omega \subset R^-(v),\ \forall k \in [\![0,\, b_v - 1]\!], \\ C_{v,\omega,k} \times C_{v,\omega,k+1} \geq 0 \end{array} \quad (2)$$

In a discrete state, if the celerity of v is positive (resp. negative), we say that v faces an external wall when the maximum qualitative level (resp. minimum) is reached $\eta_v = b_v$ (resp. $\eta_v = 0$). Thus, v cannot continue to increase (resp. decrease). In other cases, if $\eta_v < b_v$ (resp. $\eta_v > 0$), the entity v faces an internal wall when the celerities of v in the current state and in the neighbour state where v would tend, are of opposite signs.

Finally, the transitions between two discrete states are possible when the following two conditions are met: an entity v must be on the border of the discrete state ($\pi_v = 0$ or 1) and this entity must be able to change its qualitative level (do not face a wall). The definition 1 introduces the notion of sliding entity:

Definition 1: Let us consider a gene network, an entity $v \in V$ and a hybrid state $h = (\eta, \pi)$.

1) v is said *to face an external wall* at state h if: $\big((C_{v,\rho(h,v),\eta_v} < 0) \wedge (\eta_v = 0)\big) \vee \big((C_{v,\rho(h,v),\eta_v} > 0) \wedge (\eta_v = b_v)\big)$.
2) Let $h' = (\eta', \pi')$ be another hybrid state s.t. $\eta'_v = \eta_v + \mathrm{sgn}(C_{v,\rho(h,v),\eta_v})$ and $\eta'_u = \eta_u$ for all $u \in V, u \neq v$. v is said *to face an internal wall* at state h if $\mathrm{sgn}(C_{v,\rho(h,v),\eta_v}) \times \mathrm{sgn}(C_{v,\rho(h',v),\eta'_v}) = -1$, where sgn is the classical sign function, and $\rho(h, v)$ the set of resources of v at h.

We note $\mathsf{sv}(h)$ the set of *sliding entities*, that is, entities which face an internal or external wall at h.

In Figure 2, the entity v_2 faces an external wall, it slides in the discrete states $(0, 1)$ and $(1, 0)$.

For each entity, the time to reach a border is called the *contact delay*, which depends on the speed of the entity and the distance to be travelled in the temporal zone of h (definition 2) . Let us note that when the speed of an entity is null, the contact time is infinite.

Definition 2 (Touch delay): Let us consider a gene network, an entity $v \in V$ and a hybrid state $h = (\eta, \pi)$. We note $\delta_h(v)$ the *touch delay* of v at h for reaching the border of the discrete state. More precisely δ_h is the function from V to $\mathbb{R}^+ \cup \{+\infty\}$ defined by:

- if $C_{v,\rho(h,v),\eta_v} = 0$ then $\delta_h(v) = +\infty$;
- if $C_{v,\rho(h,v),\eta_v} > 0$ (*resp.* < 0) then $\delta_h(v) = \frac{1-\pi_v}{C_{v,\rho(h,v),\eta_v}}$ (*resp.* $\frac{-\pi_v}{C_{v,\rho(h,v),\eta_v}}$).

We also introduce the notion of first changing entities. For each hybrid state h, the set $first(h)$ is the set of entities reaching first their border (*i.e.* entities whose touch delay is minimum starting from the current hybrid state), excluding the *sliding* entities. Thus, the set *first* represents the set of entities that are able to change their qualitative level and lead to discrete transitions.

The previous definitions combined with the equations 1 and 2 can help to determine constraints on celerities when one wants the model to exhibit a particular observed trace. For example, along the continuous transition starting from the current hybrid state, the celerity of the entity that first changes its qualitative level is constrained by the time spent by the system in the current discrete state, in other words by its touch delay. Moreover and naturally because of the discrete transition, this entity v doesn't face a wall (otherwise it could not be able to change its qualitative level) whereas the other entities reaching their border before have to face their own walls. The complete definition of the dynamics of a gene network (continuous and discrete transitions) is given in [2]. The next part introduces a way to construct the constraints on the dynamic parameters of a gene network.

III. Hoare Logic

The Hoare logic was developped to reason rigorously about the correctness of computer programs [9]. It is based on the Hoare triple of the form $\{Pre\}\ p\ \{Post\}$ where Pre and $Post$ represent conditions on the state of the system and p is a imperative program. From an intuitive point of view, a triple is said correct if the execution of the program p, from a state where the precondition Pre is satisfied, is possible and leads to a state where the postcondition $Post$ is satisfied. The inference rules associated with the assignment, the conditional statement, the program composition and with the *while* statement, lead to a proof of the partial correctness of any program: Whenever Pre holds at the state before the execution of p, then $Post$ will hold afterwards, or p does not terminate.

In this paper, we modify the Hoare logic to be able to synthesize the necessary constraints on the dynamic parameters of the hybrid model ensuring that the model exhibits an observed biological trace. This approach was developed in the discrete modelling framework [3] and we present here the extension to the hybrid framework presented in [2].

A. Modified Hoare Logic

As part of gene network modelling, the p imperative program is replaced by an observed biological trace formally written in the path language, and the conditions Pre and $Post$ express properties on departure and arrival hybrid states.

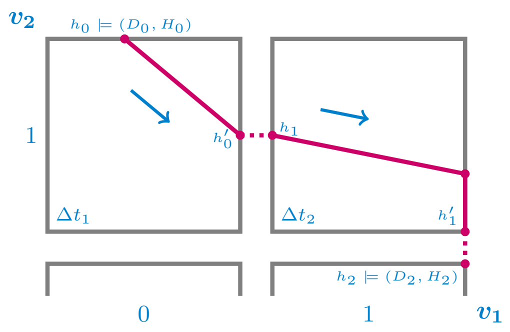


Fig. 3. Biological trace with continuous and discrete transitions. The pre and postconditions are (D_0, H_0) and (D_2, H_2). Δt_1 and Δt_2 represent the time spent in the qualitative states and the vectors represent the celerity vectors.

The pre and postconditions are expressed in the language of property : a property is a pair of the form (D, H), where the discrete condition D describes exclusively the discrete state η of each biological entity, and the H hybrid condition deals with the fractional parts π of these same entities. If no knowledge about the condition D (resp. H) is available, the latter is represented by the tautology True. Figure 3 shows a path whose starting point h_0 is defined by the precondition $(\eta(v_1) = 0 \wedge \eta(v_2) = 1, \pi_{v_1} = 0.4 \wedge \pi_{v_2} = 1)$.

The path language allows the description of the biological traces : it includes the empty path ϵ, the elementary paths which are of the form $(\Delta t, a, dpa)$ and the non-elementary paths composed of a finite sequence of elementary paths. An elementary path is defined by a triple $(\Delta t, a, dpa)$:

- The discrete path atom dpa, of the form $v+$ (resp. $v-$), expresses the next transition towards a neighbouring discrete state : the qualitative level of v is incremented (resp. decremented).
- The a element allows the modeller to specify properties related to the dynamics within the current discrete state. The modeller can characterize behaviors using the following assertion language : $a ::= \top \mid C_v \Box c \mid (\text{no})\text{slide}(v) \mid (\text{no})\text{slide}^+(v) \mid (\text{no})\text{slide}^-(v) \mid \neg a \mid a \wedge a \mid a \vee a$ where $\Box$ belongs to the set of comparison symbols. The term $\text{slide}^+(v)$ (resp. $\text{slide}^-(v)$) indicates that v will slide along its upper (resp. lower) border, while $\text{noslide}^+(v)$ (resp. $\text{noslide}^-(v)$) prevents this sliding. The term $\text{slide}(v)$ (resp. $\text{noslide}(v)$) denotes the disjunction (resp. conjunction) of $\text{slide}^+(v)$ and $\text{slide}^-(v)$ (resp. $\text{noslide}^+(v)$ and $\text{noslide}^-(v)$). Let us note that, in an elementary path, the same entity cannot appear in the dpa and in a slide term. To express that the trace of Figure 3 goes through a sliding of entity v_1 in the discrete state where $\eta_{v_1} = 1$ and $\eta_{v_2} = 1$, we can write $\text{slide}^+(v_1)$. This assertion language amends the language defined in [2] by supplementing it with the atoms $\text{noslide}^\pm(v)$ which allow a better description of experimental data.
- Finally, Δt indicates the exact time spent in the current discrete state.

Thus, the Hoare triple representing the trace of Figure 3 is written :

$$\begin{Bmatrix} D_0 \\ H_0 \end{Bmatrix} \begin{pmatrix} \Delta t_1 \\ \top \\ v_1+ \end{pmatrix} ; \begin{pmatrix} \Delta t_2 \\ \text{slide}^+(v_1) \\ v_2- \end{pmatrix} \begin{Bmatrix} D_2 \\ H_2 \end{Bmatrix} \tag{3}$$

We say that a continuous transition $h \longrightarrow h'$ satisfies the assertion couple $(\Delta t, a)$ if the transition exists, if this transition lasts Δt and if it respects the assertion a ($C_v \Box c$ is satisfied if $C_{v,\rho(h,v),\eta_v} \Box c$, $\text{slide}^+(v_1)$ is satisfied if π_{v_1} reaches 1 and v_1 faces a wall, *etc.*)

Each elementary instruction of the p path corresponds to a continuous transition of a duration Δt defined in the current discrete state, followed by a discrete transition allowing the entity of the dpa to increment or decrement its qualitative level. Figure 3 shows that from the hybrid state h_0 satisfying the precondition of Equation 3, there is a continuous transition $h_0 \longrightarrow h'_0$ with a duration of Δt_1 leading to a discrete transition $h'_0 \longrightarrow h_1$. The process is repeated for the next elementary instruction where a sliding of v_1 ($\text{slide}^+(v_1)$) occurs, until it reaches the hybrid state h_2 satisfying the postcondition (D_2, H_2).

B. *Weakest Precondition*

Edsger Dijkstra introduced a *predicate transformer semantics* [9] for imperative programming languages : with each instruction of the considered programming language is associated a predicate transformer. For each elementary instruction, the weakest precondition is a function that associates a Pre precondition with each $Post$ postcondition. By iterating the process, the weakest precondition is built for a complete imperative program. We similarly define the weakest precondition for each elementary path :

Definition 3 (Weakest Precondition): Let p be a path representing a biological trace and $Post = (D_f, H_f)$ be a postcondition indexed by a final index f. The *weakest precondition* attributed to p and $Post$ is a property : $\text{WP}^i_f(p, Post) \equiv (D_{i,f}, H_{i,f})$, indexed by a fresh initial index i and by f and whose value is recursively defined by:

- If $p = \epsilon$ is the empty path, then $D_{i,f} \equiv D_f$ and $H_{i,f} \equiv H_f$;
- If $p = (\Delta t, a, v\pm)$, then :

$$\begin{aligned} D_{i,f} &\equiv D_f[\eta_v \backslash \eta_v \pm 1], \\ H_{i,f} &\equiv H_f[\eta_v \backslash \eta_v \pm 1] \wedge \Phi^\pm_v(\Delta t) \wedge \mathcal{F}(\Delta t) \\ &\qquad \wedge \neg \mathcal{W}^\pm_v \wedge \mathcal{A}(\Delta t, a) \wedge \mathcal{J}_v; \end{aligned}$$

- If $p = p_1; p_2$ is a concatenation of paths :

$$\text{WP}^i_f(p_1; p_2, Post) \equiv \text{WP}^i_m(p_1, \text{WP}^m_f(p_2, Post))$$

which is parameterized by a fresh intermediate index m;

where $\Phi^\pm_v(\Delta t)$, $\mathcal{W}^\pm_v$, $\mathcal{F}(\Delta t)$, $\mathcal{A}(\Delta t, a)$ and $\mathcal{J}_v$ are sub-properties detailed in [2].

Intuitively, each sub-property describes a condition that must be satisfied to allow the continuous and discrete transitions associated with the elementary path $(\Delta t, a, v\pm)$:

- $\Phi_v^+(\Delta t)$ (resp. $\Phi_v^-(\Delta t)$) is the constraint that allows the entity v to reach its upper (resp. lower) border in Δt time, in other words, the touch delay of the entity v is Δt;
- $\mathcal{F}(\Delta t)$ forces all entities except the entity v of the dpa to reach their borders after v, or face a wall;
- $\neg\mathcal{W}_v^\pm$ forbids the presence of a wall for the entity v, thus allowing it to change its qualitative level;
- $\mathcal{A}(\Delta t, a)$ translates the formulas of the assertion a into specific constraints on celerities and fractional parts associated with the current state;
- $\mathcal{J}_v$ makes the junction between two successive states; intuitively the touch delay of entity v is compared with the touch delays of the other entities.

These 5 sub-properties are described in detail in [2].

In Figure 3, in the discrete state $(1,1)$, v_1 reaches a wall first and a sliding is observed until v_2 reaches its lower border. The weakest precondition associated with the path $p = \begin{pmatrix} \Delta t_2 \\ \mathsf{slide}^+(v_1) \\ v_2- \end{pmatrix}$ and the postcondition $post = \begin{Bmatrix} D_2 \\ H_2 \end{Bmatrix}$ is given by:

$$\begin{aligned} D_1 &\equiv D_2[\eta_{v_2} \backslash \eta_{v_2} - 1], \\ H_1 &\equiv H_2[\eta_{v_2} \backslash \eta_{v_2} - 1] \wedge \Phi_{v_2}^-(\Delta t_2) \wedge \mathcal{F}(\Delta t_2) \\ &\quad \wedge \neg\mathcal{W}_{v_2}^- \wedge \mathcal{A}(\Delta t_2, a) \wedge \mathcal{J}_{v_2}; \end{aligned}$$

where the sub-properties involve the fractional parts of the input state h_1 ($\pi_{v_1}^1$ and $\pi_{v_2}^1$) as well as the fractional parts from the output state h_1' ($\pi_{v_1}^{1'}$ and $\pi_{v_2}^{1'}$), and are expressed after simplification as follows:

- v_2 reaches its lower border:
  $\Phi_{v_2}^-(\Delta t_2) \equiv (\pi_{v_2}^{1'} = 0) \wedge (C_{v_2,\rho(h_1',v_2),1} < 0)$
  $\wedge(\pi_{v_2}^1 = \pi_{v_2}^{1'} - C_{v_2,\rho(h_1',v_2),1} \times \Delta t_2)$
- v_2 does not reach a border: $\neg\mathcal{W}_{v_2}^- \equiv C_{v_2,\rho(h_2,v_2),0} \leq 0$
- v_1 reaches a wall because of $\mathsf{slide}^+(v_1)$:
  $\mathcal{F}(\Delta t_2) \equiv C_{v_1,\rho(h_1',v_1),1} > 0$
  $\mathcal{A}(\Delta t_2, a) \equiv (\pi_{v_1}^1 > \pi_{v_1}^{1'} - C_{v_1,\rho(h_1',v_1),1} \times \Delta t_2)$
  $\wedge(\pi_{v_1}^{1'} = 1)$
- Junction between the states h_2 and h_1':
  $\mathcal{J}_{v_2} \equiv (\pi_{v_2}^2 = 1 - \pi_{v_2}^{1'}) \wedge (\pi_{v_1}^2 = \pi_{v_1}^{1'})$
  $\wedge(C_{v_1,\rho(h_1',v_1),1} > 0)\wedge$
  $(\pi_{v_1}^1 \geq \pi_{v_1}^{1'} - C_{v_1,\rho(h_1',v_1),1} \times \Delta t_2)$

The set of these sub-properties makes it possible to describe the constraints on the celerities and possibly to compare the celerities of different entities between them. For the current example, the assertions $\mathsf{slide}^\pm(v_1)$ and $\mathsf{noslide}^\pm(v_1)$ lead to constraints between the celerities of entities v_1 and v_2 because they are both connected to the time spent in the current discrete state.

The strategy chosen for computing the weakest precondition is the same as the one proposed by Dijkstra [9], starting from the postcondition and going up the elementary paths until the first one (Definition 3). The proof of soundness of the Hoare logic adapted to the Hybrid modelling framework and its implementation are given in [2].

C. Example of Constraints Obtained

From Figure 1, we can build the constraints on the celerities allowing the model of the negative loop to be coherent with an experimental biological trace. For this, let us consider the following path and postcondition :

$$\begin{pmatrix} 5.0 \\ \mathsf{noslide}(v_1) \\ v_2+ \end{pmatrix};\begin{pmatrix} 7.0 \\ \mathsf{slide}^+(v_2) \\ v_1+ \end{pmatrix};\begin{pmatrix} 8.0 \\ \mathsf{noslide}(v_1) \\ v_2- \end{pmatrix};$$
$$\begin{pmatrix} 4.0 \\ \mathsf{slide}^-(v_2) \\ v_1- \end{pmatrix}\ \begin{Bmatrix} D_4 \\ H_4 \end{Bmatrix}$$

avec $D_4 \equiv (\eta_{v_1} = 0) \wedge (\eta_{v_2} = 0)$ and $H_4 \equiv True$. We apply the chosen strategy of the weakest precondition calculus on the last elementary path $ep = \begin{pmatrix} 4.0 \\ \mathsf{slide}^-(v_2) \\ v_1- \end{pmatrix}$ using the postcondition $\begin{Bmatrix} D_4 \\ H_4 \end{Bmatrix}$ and we compute $\mathsf{WP}(ep, (D_4, H_4)) \equiv (D_3, H_3)$.

The time spent in the current state is 4.0 hours. The dpa v_1- indicates that the entity v_1 reaches its lower border and crosses it. The statement $\mathsf{slide}^-(v_2)$ indicates that the entity v_2 reaches its lower border before v_1 reaches its one.

According to the definition 3, we can easily compute the qualitative levels of all entities of the discrete condition from the qualitative levels observed in postcondition :

$$D_3 \equiv D_4[\eta_{v_1} \backslash \eta_{v_1} - 1] \equiv \eta_{v_1} = 1 \wedge \eta_{v_2} = 0$$

The celerities used in the current state are then identifiable thanks to the knowledge of these qualitative levels. None of the multiplexes act on its target entity, so the celerities are $C_{v_1,\emptyset,1}$ and $C_{v_2,\emptyset,0}$.

According to the definition of the weakest precondition, the hybrid condition H_3 is obtained by the following formula :

$$\begin{aligned} H_3 \equiv\ & H_4[\eta_{v_1} \backslash \eta_{v_1} - 1] \wedge \Phi_{v_1}^-(4.0) \wedge \mathcal{F}(4.0) \wedge \neg\mathcal{W}_{v_1}^- \\ & \wedge \mathcal{A}(4.0, \mathsf{slide}^-(v_2)) \wedge \mathcal{J}_{v_1} \end{aligned}$$

After simplification of the constraints obtained for each sub-property, we obtain :

$$\begin{aligned} H_3 \equiv\ & (C_{v_1,\emptyset,1} < 0) \wedge (C_{v_2,\emptyset,0} < 0) \wedge \neg(C_{v_1,\emptyset,0} > 0) \\ & \wedge (\pi_{v_1}^{3'} = 0) \wedge (\pi_{v_2}^{3'} = 0) \wedge (\pi_{v_1}^4 = 1 - \pi_{v_1}^{3'}) \\ & \wedge (\pi_{v_2}^4 = \pi_{v_2}^{3'}) \wedge (\pi_{v_1}^3 = \pi_{v_1}^{3'} - 4.0 \times C_{v_1,\emptyset,1}) \\ & \wedge (\pi_{v_2}^3 < \pi_{v_2}^{3'} - 4.0 \times C_{v_2,\emptyset,0}) \end{aligned}$$

The same process is repeated successively for the three remaining elementary paths in order to compute the constraints on all the celerities of the model.

The obtained CSP includes 26 variables : 8 celerities (2 celerities for each of the 4 discrete states) defined on $\mathbb{R}$ and 18 fractional parts defined on $[0,1]$. Indeed, the path has 4 elementary paths which defines 4 couples $(D_i, H_i)_{i\in[0..3]}$ and a final couple (D_4, H_4). For each of the 4 non final couples, there are 2 fractional parts for the entry hybrid state and 2 others for the exit hybrid state, and for the final couple, there are only 2 fractional parts for the output hybrid state. The simplifications made on the fly make it possible to determine 6 exit fractional parts that correspond to the 4 exit positions of each of the dpa and to the 2 slidings imposed by the biological trace. Finally, the CSP contains 38 constraints including 6 assignments. On the processed examples, our approach deleted trivial constraints and redundant constraints which represent nearly half of the generated constraints.

IV. Solver

In order to solve this CSP we use AbSolute [14], a continuous constraints solver based on abstract domains. For efficiency reasons, we added in the solver a preprocessing phase in which a symbolic rewriting of the constraints is performed.

A. Symbolic Rewriting

In the same way as CPLEX, AbSolute performs a preprocessing step. This step can simplify the constraints and reduce the size of the problem by eliminating variables by rewriting. During this step, only the linear equality constraints are considered. In this section, the term equality refers to a linear equality constraint.

First, we identify all the constants and keep them in a list. A constant corresponds to a unary equality. The constants are replaced throughout the CSP by their values, and this process is repeated until reaching a fixed point.

In a second step, the binary equalities are removed from the problem and kept in a list as variable views [16]. One of the variables is chosen and replaced throughout the CSP by its rewriting, and constraints on the bounds are added. This process is also repeated until reaching a fixed point.

Let us consider the CSP with $V = \{v_1, v_2, v_3\}$ the continuous variables taking their values in $D = \{D_1 = D_2 = D_3 = [-5, 5]\}$, and $C = \{c_1 : v_1 = 2, c_2 : v_1 = v_2 + v_3, c_3 : v_3 = 3 \times (v_2)^2\}$. We can see that v_1 is a constant, so we replace it everywhere by its value. The second constraint becomes a view, so we can replace v_2 with $2 - v_3$. We thus obtain the following CSP : $V' = \{v_3\}$, $D' = \{D_3 = [-5, 5]\}$ et $C' = \{c_1 : v_3 = 3 \times (2 - v_3)^2, c_2 : 2 - v_3 \geq -5, c_3 : 2 - v_3 \leq 5\}$. On this small example, we transformed a CSP with three variables, into a CSP with a constant, a view and a variable.

Let us now consider the example of Section III-C in which the CSP contains 26 variables and 38 constraints. The rewriting step leads to a new CSP which contains 16 constants, 5 views, 5 variables and 26 constraints (including 10 for the bounds of views).

This step is transparent to the user, and if they exist, the solutions are given for the initial variables of the problem.

B. Representation of a Solution

In a continuous solver, a solution is usually a Cartesian product of intervals, called a box. Using the concept of contractors [6], a solution is either a box containing only solutions of the CSP, or a small enough box (for a given precision parameter) that can contain a solution of the CSP. Figure 4 shows an example of solutions of a problem, the green boxes contain only solutions and the pink boxes may contain a solution.

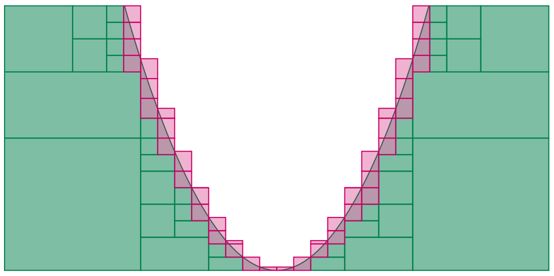


Fig. 4. Comparison between sure and unsure solutions.

AbSolute can return, if they exist, two types of solutions: sure solutions, containing only solutions, and unsure solutions that may contain a solution.

Every solution is represented by two lists. The first one contains the domain of sure variables (for any value taken in these domains, there is a solution). By abuse of language, we call it the *invariants* list. The second list contains domains that may contain a solution. By opposition, we call it *variants* list.

A solution is called *sure* if the variants list is empty and if all the variables are in the invariants list. In contrast, an *unsure* solution has at least one element in the variants list.

When a solution of the problem is found, the unknowns are substituted by their values in the views. These new assignments are added to the invariants or variants list depending on whether the unknown was invariant or variant.

V. Experimental Results

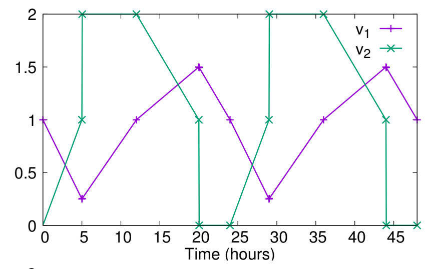
A. General Approach

The use of Hoare logic to constrain the parameters of the model we are trying to build, aims to automate the parameter identification and to minimize the intervention of the modeller in the building of the model. The input data are on the one hand the network of genes, and on the other hand a postcondition and an experimental trace expressed in the path language. Three main steps are to be distinguished : construction of minimal constraints on the dynamic parameters of the gene network, identification of a set of parameter values satisfying these constraints, and simulation of the model which can then be compared to the experimental biological trace.

The first step uses the weakest precondition calculus starting from the given postcondition by going up the path describing the observed biological trace, see subsection III-B. For models with cyclic behaviour, it is possible to add complementary constraints indicating that the initial hybrid state is identical to the final hybrid state. These constraints are simplified and reduced on the fly.

The second step uses the AbSolute constraint solver which helps to identify a set of parameter values satisfying the constraints. A solution given by the solver AbSolute corresponds to the data of a list of invariants and a list of variants. If

A)



B)

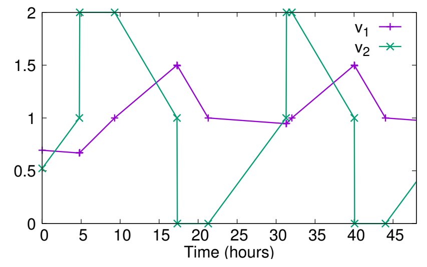


Fig. 5. A) Simulation obtained over 48 hours in cyclic behaviour. B) Simulation obtained over 48 hours without imposing a cyclic behaviour. The curve for each entity v_i corresponds to the sum of its qualitative level and its fractional part $(\eta(v_i) + \pi(v_i))$.

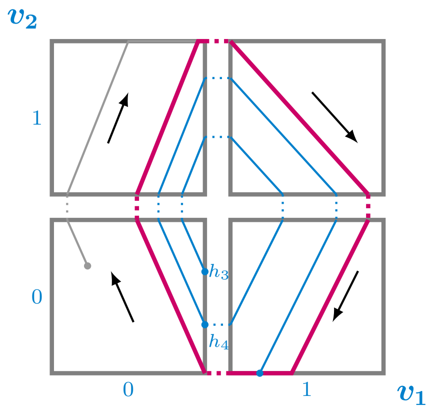


Fig. 6. Convergence of trajectories towards a possible limit cycle associated with the graph of Figure 1.The limit cycle is represented in red, and the grey and blue traces show the convergence of the traces towards the limit cycle.

the variants list is empty, the solution is said to be sure and parameter values can simply be chosen, variable by variable. In most cases, several solutions can be generated, they can be sure or not.

The user specifies a desired number of solutions and the solver displays as many as requested if it is possible. If there are no sure solutions, we randomly choose an unsure solution, a variable among the variants is randomly selected, we assign an arbitrary value to it in its definition domain, and this variable becomes constant (in the list of invariants). This constraint is then added to the CSP, and the solver is called again to compute the new solutions of this new constraint system. This process is repeated until exhausting the list of variants. When a sure solution exists and one of its invariants has a definition domain whose boundaries are distinct, its value is chosen randomly in its definition interval.

Finally, the last step is to simulate the dynamics of the gene network according to the chosen parameter values.

B. Results

For the model of Figure 1, starting from the path and the postcondition of the subsection III-C, the previously explained approach leads to a set of parameter values that we have used to simulate the evolution of the model, see Figure 5. In fact, a sure solution is immediately given by AbSolute.
The entire processing chain, from constraint construction to simulation, runs in 656 milliseconds.

The 4 elementary paths of the observed trace, see the subsection III-C, are easily identifiable in the Figure 5A : the incrementation of the qualitative level of v_2 (resp. v_1), vertically observed on y-axis, occurs at 5.0 hours (resp. 12.0 hours), and the decrementing occurs at 20.0 hours (resp. 24.0 hours). The saturation phenomena $\mathsf{slide}^{+}(v_2)$ and total degradation $\mathsf{slide}^{-}(v_2)$ are quite apparent before the qualitative level change v_1+ and v_1- respectively. Finally, the entity v_1 never slides, which is due to taking into account our $\mathsf{noslide}(v_1)$ constraints.

Simulation of Figure 5A assumes that the biological system has a cyclic behaviour of 24 hours and this constraint has been added to the CSP. Two 24-hour period can be distinguished in the Figure. Nevertheless, at each period, the phase is shifted by 0.53 seconds and this shift is due to the approximation of the reals in the solver. However, this simulation is robust since the time required to observed a phase shift equal to a period exceeds 18 years. This precision is accurate enough for biological problems.

When we don't impose cyclic behaviour, the used approach has some difficulties in precisely identifying the starting point of the simulation : the variables of the CSP corresponding to the beginning of the simulation belong to the variants list. To obtain Figure 5B, we randomly chose values for the fractional parts of the initial point, which considerably impacts the evolution of entities. Nevertheless, the simulations starting from different initializations lead after a certain time to a cyclic behaviour. Indeed, it was shown that any system built on an interaction graph of two entities and whose dynamic contains a sliding on an external wall has a cyclic behaviour called *limit cycle* towards which all traces converge [8].

Intuitively, Figure 6 shows that when we start from the hybrid state h_3, the upper border of v_2 in the current discrete state is reached and crossed, then a new continuous transition is observed in the discrete state $(0, 1)$. This continuous transition arrives at the upper border of v_1 and the alternation between continuous and discrete transitions pursues until it reaches the hybrid state h_4. The trajectory has approached the limit cycle in red, and after several passes into the discrete state $(0, 0)$, the limit cycle is reached.

Finally, we increased the number of entities in the negative feedback loop (up to 6), to compare the CPU time needed to find the 10, 100, and 1000 first solutions.

Figure 7 illustrates the interaction graph for a negative feedback loop for three or more entities.

$$\begin{tikzcd}
 & v_n \geqslant 1 \arrow[dl] & v_n \arrow[l, dashed] & \cdots \arrow[l] & \\
v_1 \arrow[r, dashed] & \neg(v_1 \geqslant 1) \arrow[r] & v_2 \arrow[r, dashed] & v_2 \geqslant 1 \arrow[ul]
\end{tikzcd}$$

Fig. 7. Interaction graph representing a negative feedback loop for three or more entities.

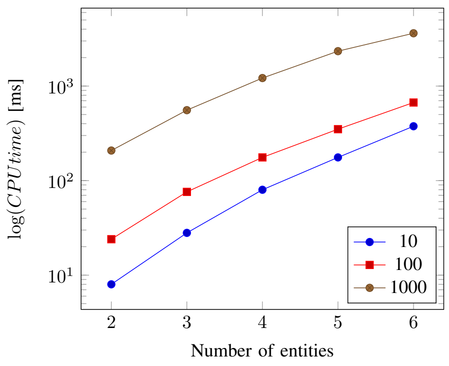


Fig. 8. CPU time to find the 10, 100 and 1000 first solutions given the number of entities in the negative feedback loop.

Figure 8 compares the CPU time (log scale) to find the first 10 (in blue), 100 (in red) and 1000 (in brown) solutions. We can see that the CPU time increases with the number of entities. This shows that the resolution time is exponential with respect to the number of entities in the interaction graph.

VI. Conclusion and Perspectives

In this paper, we propose to combine the weakest precondition calculus of the Hoare logic with the AbSolute constraint solver. This approach allows us to generate parameter values on hybrid models of gene networks leading to a dynamics exhibiting the observed biological trace. From the biological application point of view, the tool developped is very promising because, on our first examples, it makes possible to generate behaviours similar to the biological observations. During our developments, we were confronted with the size of the built formulas. The combinatorial explosion prompted us to find a strategy to simplify and reduce on the fly the constraints of the CSP. Moreover, taking into account symbolic relations between the variables in the solver allows us to reduce the number of variables to identify and thus to limit the problems of uncertainty between dependent variables due to the representation of the real numbers.

The execution time of the AbSolute solver depends on the one hand on the accuracy of the intervals of the variables of the CSP, and on the other hand on the number of solutions that the user wants. To address larger problems, especially complex biological systems, it will be necessary to find the best compromise between solution diversity and computation time to obtain satisfactory results. Finally, our current goal with therapeutic fallout is to model the coupling of the circadian cycle (alternation sleep/wakefulness) with the cell cycle (division of cells) to address questions of chronotherapy of cancer (optimization of the time of drug administration). However, the model of the cell cycle (5 abstract biological entities) leads to a system of constraints containing 178 continuous variables, 65 discrete variables, 267 constraints including certain disjunctions which considerably increases the combinatorics. The coupling of these two cycles will lead to a system of constraints that will require reviewing our strategy of finding solutions. In particular, an improvement of the rewriting system could greatly reduce the resolution time.

References

[1] R. Alur (2011): *Formal verification of hybrid systems.* In: *2011 Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT)*, pp. 273–278, doi:10.1145/2038642.2038685.

[2] J. Behaegel, J.-P. Comet & M. Folschette (2017): *Constraint Identification Using Modified Hoare Logic on Hybrid Models of Gene Networks.* In: *Proceedings of the 24th International Symposium on Temporal Representation and Reasoning (TIME)*, pp. 5:1–5:21.

[3] G. Bernot, J.-P. Comet, Z. Khalis, A. Richard & O. F. Roux (2018): *A Genetically Modified Hoare Logic.* *Theoretical Computer Science.* Https://doi.org/10.1016/j.tcs.2018.02.003.

[4] G. Bernot, J.-P. Comet, A. Richard & J. Guespin (2004): *Application of formal methods to biological regulatory networks: extending Thomas asynchronous logical approach with temporal logic.* *Journal of theoretical biology* 229(3), pp. 339–347.

[5] A. Bockmayr & A. Courtois (2002): *Using hybrid concurrent constraint programming to model dynamic biological systems.* In: *International Conference on Logic Programming*, Springer, pp. 85–99.

[6] G. Chabert & L. Jaulin (2009): *Contractor Programming.* *Artificial Intelligence* 173, pp. 1079–1100.

[7] F. Corblin, E. Fanchon & L. Trilling (2010): *Applications of a formal approach to decipher discrete genetic networks.* *BMC Bioinformatics* 11, p. 385, doi:10.1186/1471-2105-11-385. Available at http://dx.doi.org/10.1186/1471-2105-11-385.

[8] E. Cornillon (2017): *Modèles qualitatifs de réseaux génétiques : réduction de modèles et introduction d'un temps continu.* Ph.D. thesis, Université Côte d'Azur.

[9] E. Dijkstra (1975): *Guarded commands, nondeterminacy and formal derivation of programs.* *Commun. ACM* 18, pp. 453–457, doi:http://doi.acm.org/10.1145/360933.360975. Available at http://doi.acm.org/10.1145/360933.360975.

[10] L. F. Fitime, O. F. Roux, C. Guziolowski & L. Paulevé (2017): *Identification of bifurcation transitions in biological regulatory networks using Answer-Set Programming.* *Algorithms for Molecular Biology* 12(1), pp. 19:1–19:14.

[11] A. Girard, A. A. Julius & G. J. Pappas (2008): *Approximate Simulation Relations for Hybrid Systems.* *Discrete Event Dynamic Systems* 18(2), pp. 163–179.

[12] T. A. Henzinger (2000): *The Theory of Hybrid Automata*, pp. 265–292. Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/978-3-642-59615-5-13. Available at https://doi.org/10.1007/978-3-642-59615-5-13.

[13] M. Martin, P. Dague, S. Pérès & L. Simon (2016): *Minimality of Metabolic Flux Modes under Boolean Regulation Constraints.* In: *12th International Workshop on Constraint-Based Methods for Bioinformatics.*

[14] M. Pelleau, A. Miné, C. Truchet & F. Benhamou (2013): *A constraint solver based on abstract domains.* In: *International Conference on Verification, Model Checking, and Abstract Interpretation*, Springer, pp. 434–454.

[15] G. Rodrigo, J. Carrera & A. Jaramillo (2008): *Combinatorial Optimisation to Design Gene Regulatory Networks.* In: *4th International Workshop on Constraint-Based Methods for Bioinformatics.*

[16] C. Schulte & G. Tack (2014): *View-Based Propagator Derivation - (Extended Abstract).* In: *20th International Conference on Principles and Practice of Constraint Programming*, pp. 938–942.

[17] R. Thomas (1973): *Boolean formalization of genetic control circuits.* *Journal of theoretical biology* 42(3), pp. 563–585.