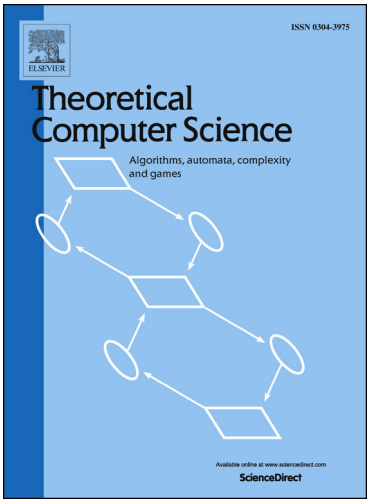
Journal Pre-proof



Constrained Kripke Structure for Identifying Parameters of Biological Models

Jean-Paul Comet, Hélène Collavizza and Laetitia Gibart

PII: S0304-3975(24)00120-8
DOI: https://doi.org/10.1016/j.tcs.2024.114505
Reference: TCS 114505

To appear in: *Theoretical Computer Science*

Received date: 30 June 2023
Revised date: 9 March 2024
Accepted date: 11 March 2024

Please cite this article as: J.-P. Comet, H. Collavizza and L. Gibart, Constrained Kripke Structure for Identifying Parameters of Biological Models, *Theoretical Computer Science*, 114505, doi: https://doi.org/10.1016/j.tcs.2024.114505.

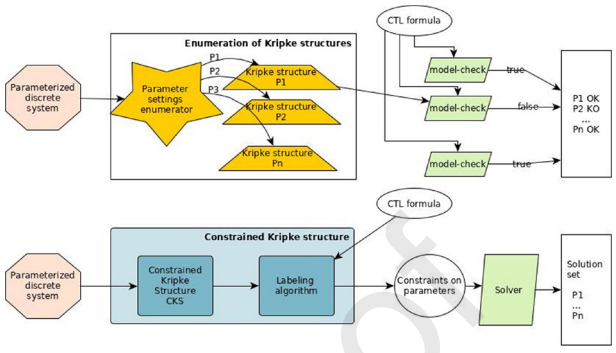
Graphical abstract

Theoretical Computer Science ●●●●, ●●●, ●●●

Constrained Kripke Structure for Identifying Parameters of Biological Models

Jean-Paul Comet , Hélène Collavizza , Laetitia Gibart

Université Côte d'Azur, I3S Laboratory, UMR CNRS 7271, CS 40121, Sophia Antipolis, 06903, France



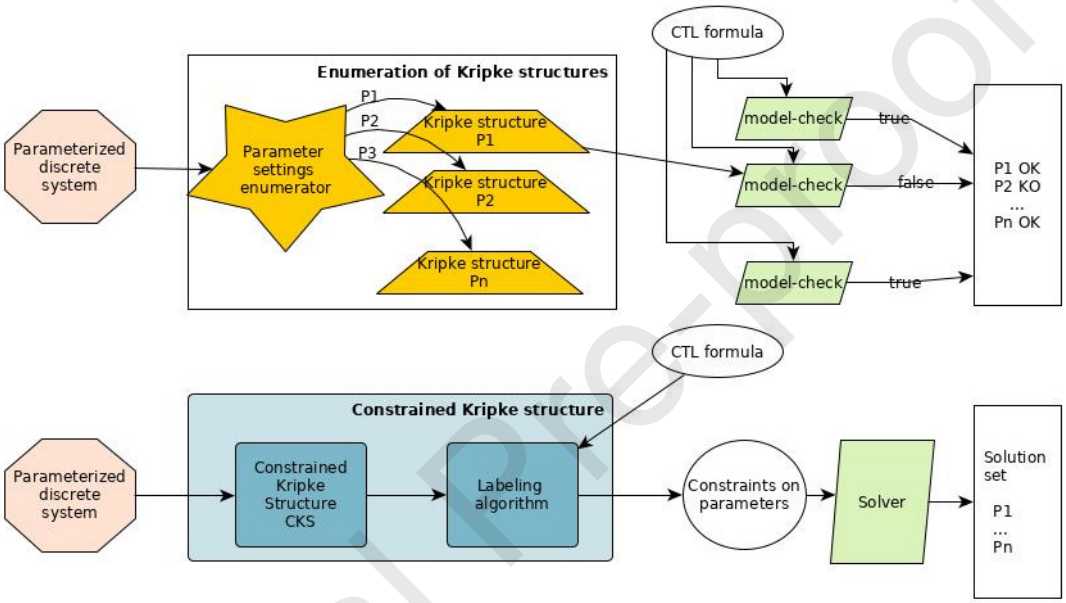
Highlights

- Identification of parameters which control the dynamics of biological systems is one of the major problem of the modelling process.
- The brute-force approach considers each parameter setting, constructs the associated dynamics (Kripke structure), and confronts it, via any model checking procedure, with the known temporal properties.
- The brute-force approach faces the combinatorial explosion of the number of parameter settings (a product of double exponentials).
- A parametric model representing the set of possible transitions (constrained Kripke structure) associated with an ad hoc model checking algorithm can decide, for each state, under which condition on the parameters the temporal properties are true in that state.

Graphical Abstract

Constrained Kripke Structure for Identifying Parameters of Biological Models

Jean-Paul Comet, Hélène Collavizza, Laetitia Gibart



Constrained Kripke Structure for Identifying Parameters of Biological Models

Jean-Paul Comet, Hélène Collavizza, Laetitia Gibart

[a] *Université Côte d'Azur, I3S Laboratory, UMR CNRS 7271, CS 40121, Sophia Antipolis, 06903, France, firstname.lastname@univ-cotedazur.fr*

Abstract

When modelling a complex biological system, the bottleneck of the process is the determination of parameter values that lead to model dynamics that are compatible with observations. Even for discrete modelling frameworks, this step can be limiting. Here we introduce a representation of the whole family of discrete models that can be associated to a biological system, where states are shared by all models and transitions are labelled by constraints on dynamical parameters. A model checking procedure is defined to handle this new representation. This procedure extracts the conditions on parameter settings that are compatible with a given dynamical property expressed in a temporal logic. We prove the correctness of our model checking procedure and illustrate the advantage of such an approach on some different systems of biological interest.

Keywords: Biological system analysis, Model Checking, Constrained Kripke structure, Automated parameter and model synthesis, Frameworks for model verification of biological systems

1. Introduction

Motivation. One of the major problems when modelling complex biological systems lies in the identification of the parameters which control their dynamics. On the one hand, the observational capacity does not allow direct precise measurements of these parameters and, on the other hand, the observations are often global properties of the system (multi-stationarity, cyclical behavior called homoeostasis,...).

For several decades now, a number of observations have been made which show that certain evolutions of biological systems are *qualitative* in nature:

From a functional point of view, each gene of a gene regulatory network can be seen as either switched on or off. For example, during the famous cell cycle, the notion of check-point allows the cell to wait for certain qualitative key events to be carried out before continuing the cycle [1, 2]. With the same idea, the change of functioning of the energetic metabolism (the so-called Warburg/Crabtree effect) seems to result from a qualitative change of the environment [3]. To make the story short, ones of the pioneers of these qualitative modelling frameworks were R. Thomas [4] and S. Kauffman [5, 6]. And after the advent of formal verification methods, these qualitative approaches have been revived due to the addition of formal methods to help the modelling process [7, 8, 9, 10].

Roughly speaking, a biological system is often known by the set of interactions between entities (so-called variables) but the relative strength of the interactions is unknown. It is indeed very difficult to determine the fate of a variable under the control of two regulators: Is a unique activator sufficient to activate the target, or does the target variable need both activators to be boosted? Worse, sometimes the simultaneous presence of both activators leads to inhibition (because they form a complex that has a completely different activity). These relative strengths of influences are generally coded by *parameters*.

The main question is then to identify the parameters that lead to a dynamic that is consistent with all systemic observations. The often used brute-force approach have been firstly implemented: consider each possible valuation of parameters (parameter setting), construct the associated dynamics of the model, and then confront it, via any model checking procedure, with the known temporal properties expressed formally in a temporal logic, see Fig. 1-top[1]. Nevertheless, this approach implemented in `TotemBioNet` [1, 11], faces the combinatorial explosion of the number of parameter settings to consider which is a product of double exponentials. Although this enumerative approach is of great help when the number of parameter settings is reasonable, it is generally too long to be used extensively in routine.

When the modeller is interested in only one parameter setting, leading to a model satisfying a particular specified temporal property, several approaches has been proposed. In [12] the authors propose an approach which

[1]This graph, and others presented in the paper were designed using the tool yEd (www.yworks.com/products/yed).

exploits the capability of symbolic model checkers to efficiently manipulate implicit descriptions of the state and parameter space. In order to find a parameterisation satisfying a formula φ, the main idea is to test whether $\neg\varphi$ holds. Thus one verifies the absence of a parametrization satisfying φ and a counterexample to $\neg\varphi$ thus directly returns a valid parametrization for φ. Whereas such an approach already proposed in [13] allows one to find a first parameter setting which lead to a model satisfying the specification formula, it does not address the issue of finding exhaustivelly the set of parameter settings validating it.

When the temporal specification concern only long-term behavior, one can also use state-of-the-art methods for attractor detection [14] to extract the models satisfying the temporal specification.

When the temporal specification is more general and exhaustiveness is desired, the diversity of behaviors may be due only to the valuation of parameters. In that case, the models obtained by valuating the parameters share the structure of the regulations, leading often to a set of models having the same states. It then seems natural to reason about sets of models. This remark has been highlighted by several authors. For example, in [15], the different models share the same states and are represented in a single structure where each transition is labelled by the parameter settings which are compatible with the considered transition. The LTL model checking has also been adapted to this structure. Then, the CTL model checking has been adapted to this combined structure [16] with a parallel implementation in order to get the results in an acceptable time. Finally the model checking has been improved to manipulate symbolically the parameter setting sets [17].

In this article, we take up this idea of a single structure to represent the different models, adding a symbolic representation of the parameter settings within the structure itself. The intuition is to consider each parameter setting (leading to a certain dynamics) as a kind of *variant* of a unique parametric model. This parametric model contains all the transitions that exist in at least one concrete model. Considering this complete set of transitions, the usual model checking procedures cannot be directly used for verifying such a parametric model because of the superposition of all possible dynamics. However, one can go further if the actual transitions in a given parameterised model (model resulting from a parameter setting) can be derived from the parametric model and the parameter setting under consideration. In such a case, we can imagine a model checking algorithm that takes as input a parametric model and a temporal logic formula, and decides for each state

under which condition on the parameters the temporal formula is true in that state. In other words, the algorithm does not answer whether the formula is true or not, but under what conditions on the parameters it is true. (see Fig. 1-bottom).

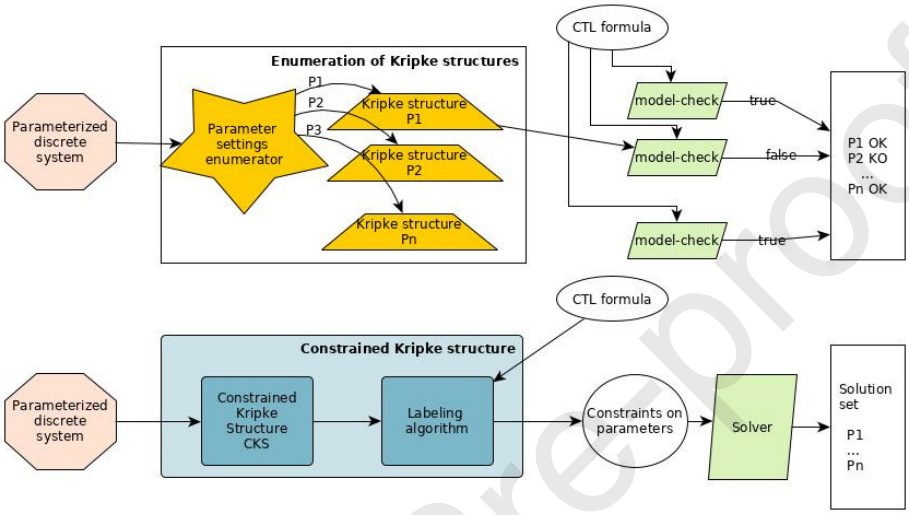


Figure 1: Top: brute force approach where all parameter settings P are enumerated, a Kripke structure KS_P (see below) is built for each one, and a model-checker is called to check the formula on each KS_P. Bottom: a constrained Kripke structure CKS which aggregate all the parameter settings is built, a new labelling algorithm is used to compute the constraints which characterize the family of KS_P satisfying the formula.

Related works. Model checking a family of variants has already been proposed in a totally different application domain. A decade ago, A. Classen and co-workers [18, 19] proposed the concept of Variability-intensive systems (VIS) which form a very large and heterogeneous class of systems whose behavior can be adapted by activating or suppressing some predefined features. Applied to our case, all models in the family would share the same sets of states and transitions, but the transitions would be labelled by parameter constraints that would have to be validated for the transition to exist. Variability would then be controlled by the parameter setting. In the context of software engineering, variability mechanisms allow the adaptation of software to the needs of a particular class of users and the environment. This kind of systems needs also automated processing for verification and validation which becomes challenging: as in our case, one has to face the

combinatorial explosion of the number of possible behaviors. To tackle them, authors proposed *Featured Transitions Systems (FTS)* to model and verify the behaviors of VIS. In a FTS, each transition is annotated with a combination of features determining which variants can execute it. They defined a feature-oriented version of LTL [20] or CTL [18], allowing the verification of properties on a subset of configuration parameter settings. Our goal is to exhibit exhaustively the parameter settings which make true a particular CTL formula without any knowledge on this subset of parameter settings. The model checking of VIS as well as extensions based on abstract interpretation of such variational models [21] or using μ-calculus extensions [22] does not fulfil this aim of exhaustivity.
When identifying parameters, it may also be relevant to consider information from properties that cannot be expressed in classical CTL. Thus, different authors proposed extensions of CTL in order to identify attractors or manipulate them as specifications [23]. Let us finally mention that the use of fully symbolic hybrid CTL model checking for parameter/model inference has been also revisited in [24] for inference of Boolean networks.

Paper structure. In Section 2, the R. Thomas' framework used to model biological systems is introduced. The influence graph defines the individual influence of a variable on others, while the parameter settings define the overall dynamics of the system. Moreover, temporal logic (CTL) allows the known behavior of the system to be expressed formally. In Section 3, we present a brute force approach where parameter settings are enumerated: A Kripke structure is built for each setting and the temporal property is checked on each one of these structures. This exhaustivelly collects all settings that validate the property. In Section 4, we propose a new representation of a family of Kripke structures, called *Constrained Kripke Structure* (CKS) which agregates all the Kripke structures built for each particular parameter setting. This aggregation was possible because (1) the set of states as well as basic state properties are shared by all Kripke structures, and because (2) one can check whether a transition actually exists in a particular Kripke structure by checking the satisfaction of a constraint expressed on a common set of parameters. We also present in this section the labelling algorithm for model checking a CTL formula in a CKS (presented connective by connective). In Section 5 we present our main results: the proof of the validity of the labelling algorithm, some implementation considerations about our first prototype, and last, the application of this new approach on several biological case

studies. And finally, we conclude in Section 6.

2. R. Thomas' Framework for Modelling Biological Systems

The framework of R. Thomas is a qualitative approach where the concentration values of biological substances are abstracted as discrete levels. These levels are derived from thresholds at which certain changes occur.

Given a biological system, and some assumptions on the dynamics of this system, our modelling steps are to:

1. Create an *influence graph* (with multiplexes) that *statically* describes the individual influences between variables. This step can be really tricky, and can need a lot of discussion with the specialist of the system.
2. Deduce from this influence graph the set of *parameter* symbols which represent the relative strength of influences on their common target. These parameters are used to define the *global dynamics* of the system via a *parameter setting* which assigns a value to each parameter.
3. Find an appropriate translation of the biological knowledge in terms of a temporal logic formula.
4. Exhaustively find any parameter setting that makes the dynamics, based on both the influence graph and the considered parameter setting, consistent with this temporal formula.

Step 4 is the so-called *parameter identification problem* in systems biology, which is one of the major challenge in R. Thomas' framework. The set of parameter settings that are identified provides crucial information about the relevance of the modelling:

- If this set is empty, then either the influence graph or the formula must be revised.
- If this set is huge, then the formula needs to be revised as it is not meaningful.
- Otherwise, the set of parameter settings can be analyzed and used in a process of model enrichment [25, 26].

Our previous approach successively built a Kripke stucture to model-check the property for *one* parameter setting. The challenge of the work

presented in this paper is to use a *global Kripke structure* which encompasses all the parameter settings.

In this section, we first define the influence graph with multiplexes, then introduce parameters and the Computational Tree Logic (CTL) we use to express biological properties.

2.1. Influence Graph with Multiplexes

Multiplexes were introduced in [27], as an extension of R. Thomas' modelling framework [28]. They express, via a logic formula, some conditions under which an influence occurs. This reduces the number of parameters, which directly depends on the number of predecessors (see Def. 2).

Definition 1 (Influence Graph with multiplexes)**.** *An* influence graph with multiplexes $IG = (V, M, A)$ *is a directed graph such that:*

- *Vertices are variables in* V *or multiplexes in* M *(*$V \cap M = \emptyset$*).*
- *With each variable* $v_i \in V$ *is associated a discrete domain* $D_i = [l_i..u_i]$ *where* $0 \leq l_i \leq u_i$*.*
- *Arcs in* A *go from multiplexes to variables (*$A \subset M \times V$*).*
- *With each* multiplex $m \in M$ *is associated a formula* φ_m *which expresses the condition under which* m *influences its target variable(s). The language of multiplex formulae is defined by:*
  - *Atoms are atomic formulae* $(v_i \geqslant n)$ *with* $v_i \in V$ *and* $n \in D_i$*.*
  - *If* φ*,* φ_1 *and* φ_2 *are multiplex formulae, then* $\neg\varphi$*,* $\varphi_1 \Box \varphi_2$ *are also multiplex formulae, where* $\Box$ *is either* $\wedge$*,* $\vee$ *or* $\Rightarrow$*.*

Running example : Pseudomonas æriginosa Influence Graph. In [7] we introduced formal methods for selecting the parameter settings compatible with all the available knowledge on a little system: production of mucus in *Pseudomonas æriginosa*. This system will be used as running example throughout this article.

Pseudomonas æriginosa is an opportunistic bacterium that can secrete mucus. Mucus is composed by alginate, a protein named $AlgU$. $MucB$ is another protein that is co-expressed with $AlgU$ and they are both co-activated by a genetic element called an *operon*. Moreover $MucB$ inhibits the operon and the operon activates itself through several molecules. When

these bacteria affect the lung, they cause serious infections *via* accumulation of mucus, particularly for cystic fibrosis diagnosed patient [29].

Fig. 2 presents the influence graph of *Pseudomonas æriginosa*. Orange nodes are variables and blue nodes are multiplexes. Two levels are used for $MucB$ (presence or absence) while three are used for *Operon* to express a non-mucoid state (level 0), a mucoid state (level 2) and an intermediate state (level 1) which can switch to mucoid or non-mucoid state. Thus there exists 6 possible states for this influence graph: 2 distinct values for $MucB$ and 3 distinct values for *Operon*. The set of states will be denoted S in the following of this article. An arrow from a multiplex to a variable expresses an influence, according to a threshold property. For example, the arrow from *prod* to $MucB$ expresses that when the level of *Operon* is sufficient (more than level 1), then $MucB$ is activated. Conversely, the arrow from $free$ to *Operon* expresses that when the level of $MucB$ is sufficient (more than level 1), then *Operon* is inhibited: `!(MucB>=1)` is the concrete syntax of our tool `TotemBioNet`[2] to denote $\neg(MucB \geq 1)$. Thus `!(MucB>=1)` expresses the absence of an activation, seen as an inhibition. Last, the arrow from *alg* to *Operon* expresses a self-loop: when the level of *Operon* is maximum (more than level 2) then *Operon* is activated.

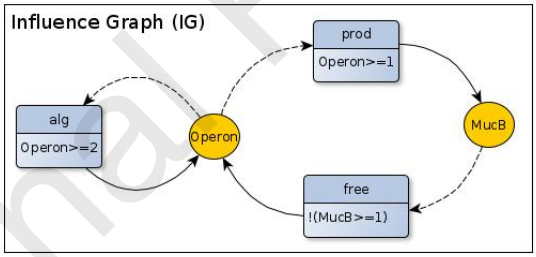


Figure 2: Influence graph of the *Pseudomonas æriginosa* mucus production system. The dashed arrows are there to make easier to understand the interactions: they can be deduced from formulae of multiplexes. `!(MucB>=1)` stands for $\neg(MucB \geq 1)$ in the concrete syntax of `TotemBioNet`.

[2]The complete concrete syntax can be found at `https://gitlab.com/totembionet/cks-bionet/-/blob/main/CKS-BioNet/doc/userManual.pdf`

2.2. Regulatory Networks

Given an influence graph, *parameters* represent the relative strength of influences on a variable.

Definition 2 (Parameters)**.** *The* parameters *of a variable v are denoted $K_{v,\omega}$ where ω is a subset of the predecessors of v in IG. $K_{v,\omega}$ represents the direction of evolution of variable v when it is controlled according to the multiplexes in ω.*

- *A variable v has $2^{d^-(v)}$ parameters where $d^-(v)$ is the number of predecessors of v in IG.*
- *$\mathcal{K}(v)$ denotes the set of parameters of variable v.*
- *$\mathcal{K}$ is the set of all parameters: $\mathcal{K} = \cup_{v \in V} \mathcal{K}(v)$.*

Definition 3 (Regulatory network)**.** *A* Regulatory network *is a couple $\mathcal{N} = (IG, \mathcal{K})$.*

The dynamics of a regulatory network $\mathcal{N} = (IG, \mathcal{K})$ is determined by a valuation of its parameters in $\mathcal{K}$, which is called a *parameter setting*.

Definition 4 (Parameter Setting)**.** *Given a regulatory network $\mathcal{N} = (IG, \mathcal{K})$, a* parameter setting *$P$ assigns to each parameter $K_{v_i,\omega}$ associated with variable v_i a value in its discrete domain D_i:*

$$\begin{array}{rccc} P : & \mathcal{K} & \to & \mathbb{N} \\ & K_{v_i,\omega} & \mapsto & k_{v_i,\omega} \in D_i \end{array}$$

The set of parameter settings of $\mathcal{N}$ is denoted $\mathcal{P}$.

Note that the space of parameter settings is huge. Indeed, for a regulatory network $\mathcal{N}$, the number of parameter settings is the product of all possible values of the parameters associated to each variable. Each parameter of a variable $v_i \in V$ can take its value in integer interval $D_i = [l_i, u_i]$. So there are $(u_i - l_i + 1)^{|\mathcal{K}(v_i)|}$ possible parameter settings for the parameters of v_i where $|\mathcal{K}(v_i)| = 2^{d^-(v_i)}$ is the number of parameters of v_i (i.e. the cardinal of the set of parts of the set of predecessors of v_i, see Def. 2). So, the number of parameter settings is: $|\mathcal{P}| = \prod_{v_i \in V} (u_i - l_i + 1)^{2^{d^-(v_i)}}$.

A parameter setting $P \in \mathcal{P}$ sets the global dynamics of the system. On each state, the possible transitions of the system depend on the *applicable parameters* of the variables which are determined by the combination of multiplexes whose formulae are true in that state.

Definition 5 (Applicable parameter on a state)**.** *The* applicable parameter *of variable v on a state $s \in S$, is the only parameter $k_{v,\omega}$ such that:*

- *$\forall m \in \omega$, φ_m is true on state s.*
- *$\forall m \in pred(v), m \notin \omega$, φ_m is false on state s (where $pred(v)$ is the set of predecessors of v).*

The applicable parameter of v in state s is denoted $k_v[s]$. Each multiplex m, predecessor of v, such that φ_m is true on s is called a resource *of v in s.*

Given a parameter setting P, a variable v and a state s, the value of the applicable parameter for v on s indicates if, on state s, v tends to increase, decrease or stay stable.

Example : Pseudomonas æriginosa parameters. Variable $MucB$ has one predecessor thus two parameters are associated with it: $K_{MucB,\{\}}$ when formula associated with *prod* is false and $K_{MucB,\{prod\}}$ when formula associated with *prod* is true (see Fig. 2). In the same way, variable *Operon* has two predecessors thus four parameters are associated with it: $K_{Operon,\{\}}$, $K_{Operon,\{free\}}$, $K_{Operon,\{alg\}}$ and $K_{Operon,\{alg,free\}}$.

On state 21 ($Operon = 2$ and $MucB = 1$), $K_{Operon,\{alg\}}$ is the applicable parameter for *Operon* (because formula *alg* is true while formula $free$ is false), and $K_{MucB,\{prod\}}$ is the applicable parameter for $MucB$ (because formula *prod* is true).

2.3. CTL formula

We focus on biological properties which express global behavior such as the existence of an attraction basin or a sustained oscillation. For such long-term properties, we need to express some relations:

- About values of variables and combinations of values for several variables.
- About values of variables in the future, depending on successive transitions induced by a parameter setting.

The first item leads to the need of logical connectives such as logical *or*, logical *and*, etc. whereas the second item involves the use of temporal connectives. In other terms, *one needs a temporal logic.*

Among the different temporal logics we have chosen the *Computational Tree Logic* (CTL). Because the states are shared by all dynamics (derived from all possible parameter settings), for a given state, whatever the parameter setting, the value of each variable will remain the same. Thus we consider CTL formulae inductively built over the set of variables V in the usual way, using logical connectives, modalities on paths (E Exists, A All) and modalities on time (X neXt, F Future, U Until, G Generally).

Definition 6 (ϕ_{CTL} : set of CTL formulae over V)**.** *ϕ_{CTL}, the set of CTL formulae over V, is defined by :*

- *An atomic proposition $a \in AP$ ($AP \subset \Phi_{CTL}$) is either a constant ($\top$, $\bot$) or $v \triangle n$ where $v \in V$ is a variable, n is a natural number and $\triangle$ is a comparison operator among $<, >, \leq, \geq, =$.*
- *A CTL formula $\varphi \in \phi_{CTL}$ is either an atomic proposition, or*
  - *$\neg\varphi$ or $\varphi_1 \nabla \varphi_2$ where ∇ is a logical connective among $\wedge, \vee, \Rightarrow, \Leftrightarrow$*
  - *$E[\varphi_1 U \varphi_2]$ or $A[\varphi_1 U \varphi_2]$*
  - *$\odot\varphi_1$ where $\odot$ is either EX, EF, EG, AX, AF or AG*

  with φ, φ_1 and φ_2 in ϕ_{CTL}.

Example : Operon stable states. From literature we know that a non-mucoid *Pseudomonas æriginosa* (which does not produce mucus) will never create mucus, and that when *Pseudomonas æriginosa* is mucoid (which does produce mucus), it cannot turn off again. This can be translated by the following CTL formula:

$$\begin{array}{rccl} \varphi \equiv & ((Operon = 0) & \Rightarrow & AG[\neg(Operon = 2)]) \quad \wedge \\ & ((Operon = 2) & \Rightarrow & AG[\neg(Operon = 0)]) \end{array}$$

It expresses that starting from a state where $Operon = 0$ (non-mucoid) then there is no path leading to a state where $Operon = 2$ (mucoid), and vice versa.

Given a regulatory network and a CTL formula φ which expresses some global behavior of the biological system, the *parameter identification problem* consists in choosing the parameter settings such that φ is true taking into account the transitions induced by the values of the parameters.

3. Identification by Enumeration of Kripke Structures

We first formally define here the Kripke structure for *one setting* that was used in our previous enumerative approach and present our associated tool `TotemBioNet`. This Kripke structure will be generalized in the next section, in order to take into account *all parameter settings*, in a new global approach.

3.1. Kripke Structure for One Parameter Setting

The Kripke structure representing the dynamics of the model associated with a given parameter setting P is defined as follows.

Definition 7 (Kripke structure associated with parameter setting P)**.** *Given a regulatory network $\mathcal{N} = (IG, \mathcal{K})$ whose variables are denoted $v_1, \ldots, v_n$ and $P \in \mathcal{P}$ a parameter setting for $\mathcal{N}$, the associated* Kripke structure *$KS_P = (S, R_P, L)$ is defined as follows:*

- *S is the set of states: $\Pi_{i=1}^{n} D_i$.*
- *R_P is the set of transitions computed as follows:*
  1. **Loops***: there is an arc from s to itself if $P(k_{v_i}[s]) = s_i, \forall i = 1 \ldots n$.*
  2. **Arcs***: there is an arc from $s^p = (s_1^p, ..., s_n^p)$ to $s^q = (s_1^q, ..., s_n^q)$ if there exists one and only one index i s.t. $s_i^p \neq s_i^q$ with either : $s_i^q = s_i^p + 1$ and $P(k_{v_i}[s^p]) > s_i^p$ or $s_i^q = s_i^p - 1$ and $P(k_{v_i}[s^p]) < s_i^p$.*
- *L is a labelling function $L : S \to 2^{AP}$ where $L(s)$ is the set of atomic propositions which are true in s.*

Remember that $P(k_{v_i}[s])$, the applicable parameter of variable v_i on state s (see Def. 5), is the value towards which v_i evolves from s. Thus item 1 (Loops) and item 2 (Arcs) express how the variable v_i evolves:

- Stability (item 1): For the variable v_i to be stable, v_i must have reached its focal value, thus v_i must be equal to the value of its applicable parameter.
- Transition towards a neighbour (item 2): In order for variable v_i to increase (or decrease) by one level, the value of v_i must be less (or greater) than the value of its applicable parameter.

Let us remark that whatever the valuation $P \in \mathcal{P}$, the Kripke structure KS_P deduced from the parameter setting P is serial (i.e. every state has at least one successor). In fact, two cases can arise. (i) There is an applicable parameter whose value is different from the current value of the associated variable, and in that case there is a transition to another state. (ii) All the parameters applicable in this state are equal to the current value of the associated variable, and in that case, there is a loop of s on itself.

Example : Kripke structures for Pseudomonas æriginosa. Fig. 3 provides two Kripke structures obtained from two different parameter settings P_1 and P_2 (given in concrete syntax, see right side of Fig. 3). Remember that in state 21, $K_{Operon,\{alg\}}$ is the applicable parameter for $Operon$ and $K_{MucB,\{prod\}}$ is the applicable parameter for $MucB$. Thus for example, transition from node 21 to node 20 exists in KS_{P_1} because $K_{Operon,\{alg\}} = 2$ ($Operon$ stays stable) and $K_{MucB,\{prod\}} = 0$ ($MucB$ tends towards 0). Transition from node 21 to node 11 exists in KS_{P_2} because $K_{Operon,\{alg\}} = 0$ ($Operon$ tends towards 0 by one step) and $K_{MucB,\{prod\}} = 1$ ($MucB$ stays stable). Note that self loops allow the Kripke structure to be serial.

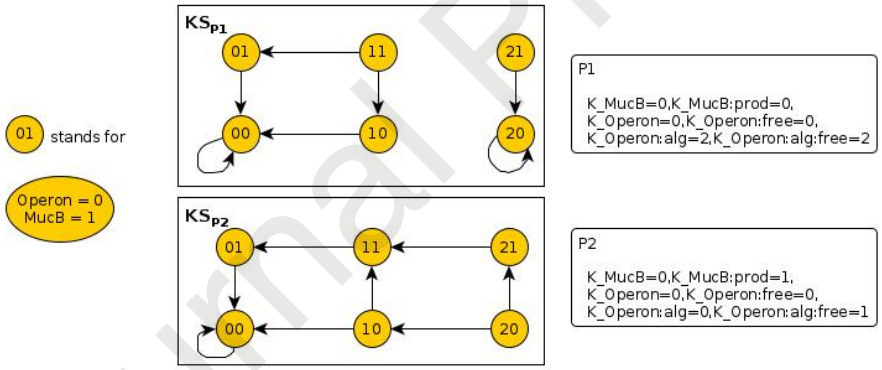


Figure 3: Two Kripke structures associated with the parameter settings P_1 and P_2. Formula φ (see Subsection 2.3) is true on KS_{P_1} and false on KS_{P_2} because starting from state 21 or 22 the state 01 can be reached. Parameters are written in concrete syntax: `K_VarName:p1:p2:...:pn` denotes the parameter of variable $VarName$ under influence of predecessors $p_1, \ldots, p_n$ (e.g. `K_MucB:prod` stands for $K_{MucB,\{prod\}}$).

Given a parameter setting P and a CTL formula $\varphi \in \Phi_{CTL}$, a model-checker tool can be called to check if φ is true for the KS_P Kripke structure. To solve the previously described *parameter identification problem*, a brute

force approach consists then in enumerating the parameter settings, build the associated Kripke structure KS_P and call a model-checker tool. This enumeration approach is implemented in our tool *TotemBioNet*.

3.2. TotemBioNet: a Tool for Parameter Identification

TotemBioNet has been widely used to model several biological systems, where abstraction can provide new insight on the biological system: Mucus production in *Pseudomonas æruginosa* [7, 11] in different environments, verification of checkpoint properties in cell cycle [1], and Warburg/Crabtree effect in central carbon metabolism [30].

TotemBioNet takes as inputs:

- An influence graph.
- The (potentially reduced) domains of each parameter.
- Some formalised behaviors, expressed as CTL formulae for temporal properties.

TotemBioNet enumerates the parameter settings. Then for each $P \in \mathcal{P}$, it builds the associated Kripke structure KS_P, and calls the model-checker $NuSMV$ [31][3] to check the CTL formula. The *TotemBioNet* output, written in a .csv file, is the set of parameter settings that solves the identification parameter problem.

Note that hopefully the number of parameter settings can often be reduced using biological knowledge. One possibility is *thought experiments*. Since the parameter $K_{v,\omega}$ represents the direction of evolution of variable v when the resources set is ω, the biologist can be able in some cases to identify this value by hand [30]. Another possibility is to take benefit of information about a well known path of the system using a so-called Hoare logic [10]. In that case, some conditions on parameters, that allow the Kripke structure to exhibit this trace, can be automatically collected by TotemBioNet [1]. This also reduces the number of parameter settings which have to be enumerated.

The global *TotemBioNet* process is illustrated in Fig. 4.

[3]The parameters are actually translated in NuSMV by *FROZENVAR* because we set their values by enumeration before building the NuSMV input file. In the first experiments, we tried to use non-instantiated variables, but this approach was not feasible: it did not scale up and moreover, the symbolic CTL Model-checker has problems extracting counterexamples.

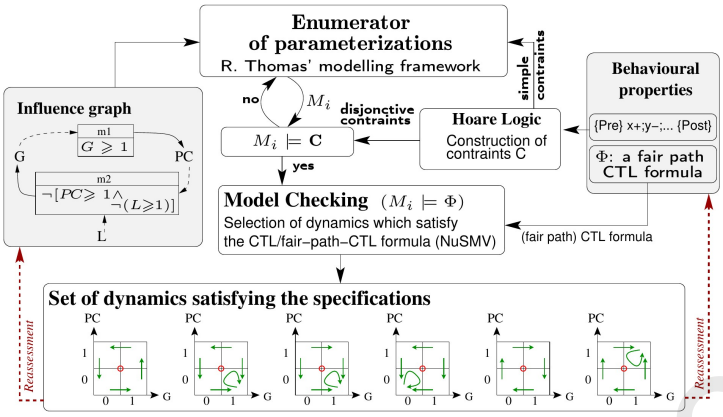


Figure 4: *TotemBioNet* processing flow

3.3. Related Tools for Parameter Identification

While the analysis of a unique Kripke structure associated with a parameter setting is commonly carried out using different tools, the search for all parameter settings that lead to a Kripke structure satisfying certain properties, remains a task with few tools. For example, the colomoto platform [32] provides several tools capable of analyzing the dynamics associated with a parameter setting *via* the analysis of the transition graph (BoolNet [33], GINsim [34]). For verifying the coherence between the traces given by the transition graph and temporal properties, some of these tools even allow the verification of associated temporal formulae through the model checker NuSMV [31]. However, none of these tools allows the enumeration of all possible dynamics of a non-parameterized network, combined with the formal verification of properties in temporal logic as *TotemBioNet* does.

4. Identification based on a Global Constrained Kripke Structure

The aim of this new approach is to aggregate all the Kripke structures obtained for each of the parameter settings, into a global Kripke structure which describes all the possible transitions.

4.1. Contraints on Parameters

To be able to express the conditions for a transition on the global structure to occur, we make an hypothesis: each envisioned transition can be labelled by a constraint on parameters that defines the set of settings sharing

this transition. More precisely, we suppose that for each possible transition $(s, s') \in \cup_{P \in \mathcal{P}} R_P$ in any Kripke structure, there exists a constraint c on parameters of $\mathcal{K}$ that makes possible to decide whether this transition (s, s') belongs to R_P or not: $(s, s') \in R_P$ iff the constraint c is satisfied for the parameter setting P.

Definition 8 ($C_\mathcal{K}$: set of constraints on parameters)**.** *The set $C_\mathcal{K}$ of well-formed constraints over parameters in $\mathcal{K}$ is inductively defined by:*

- *An atom is either a constant ($\top$, $\bot$) or $k \triangle n$ where $k \in \mathcal{K}$ is a parameter symbol, n is a natural number and $\triangle$ is a comparison operator among $<, >, =$.*
- *A constraint on parameter $c \in C_\mathcal{K}$ is either an atom, or $\neg c$ or $c_1 \nabla c_2$ where c, c_1 and c_2 are constraints in $C_\mathcal{K}$ and ∇ is a logical connective among $\wedge$ and $\vee$.*

We define now the set of parameter settings satisfying a constraint.

Notation 4.1 (Satisfaction relation of a constraint)**.** *For each couple $(P, c) \in \mathcal{P} \times C_\mathcal{K}$, we note $P \vDash c$ for saying that constraint c is evaluated to true with the valuation P.*

Definition 9 (Parameter settings of a constraint)**.** *For each $c \in C_\mathcal{K}$, we define $[\![c]\!] = \{P \in \mathcal{P} \mid P \vDash c\}$ the set of parameter settings which satisfy the constraint c.*[4]

4.2. Constrained Kripke Structure for a Regulatory Network

The aggregation of all Kripke structures from the previous section is done within a *constrained Kripke structure*: A constraint on parameters is associated with each transition in order to define the set of parameter settings for which the transition exists.

Definition 10 (Constrained Kripke Structure over $C_\mathcal{K}$)**.** *A Constrained Kripke Structure is a quadruplet $CKS = (S, C_\mathcal{K}, R, L)$ where :*

- *S is the set of states of the system.*

[4] In terms of Constraint Satisfaction Problem, $[\![c]\!]$ is the set of solutions of constraint c on the Cartesian product $\prod_{i=1}^{n} [l_i..u_i]^{|K(v_i)|}$.

- *$C_{\mathcal{K}}$ is the set of constraints over parameters.*
- *R is a relation $R \subseteq S \times C_{\mathcal{K}} \times S$ and each element (s, c, t) of R is denoted $s \xrightarrow{c} t$. This expresses that the relation between s and t exists only under the constraint c. Each element $s \xrightarrow{c} t$ is called a transition.*
- *L is a labelling function $L : S \to \mathcal{P}(AP \times C_{\mathcal{K}})$.*

This definition and the associated labelling algorithm presented in Section 4.3 are a general framework for model checking formulae on Kripke structures with constrained transitions. However, we are interested in the application domain where CKS is obtained from an influence graph $IG = (V, E)$.

Definition 11 (Constrained Kripke structure for an influence graph)**.** *Given an influence graph $IG = (V, E)$, its associated constrained Kripke structure, denoted CKS_{IG}, can be constructed in the following way:*

- *S is the set of states : $\Pi_{i=1}^{n} D_i$.*
- *$C_{\mathcal{K}}$ is the set of constraints over parameters.*
- *R is the set of transitions computed as follows:*
  1. **Loops:** *For each state $s = (s_1, s_2, ..., s_n) \in S$, there exists a transition $s \xrightarrow{c} s$, where the constraint c is : $c \equiv \wedge_{i=1...n} k_{v_i}[s] = s_i$.*
  2. **Arcs:** *For each state $s = (s_1, s_2, ..., s_n) \in S$, for each variable index $i \in [1..n]$:*
     - *If $s_i > l_i$ (lower bound of v_i), then there exists a transition $s \xrightarrow{c} s'$ where $s' = s[s_i \leftarrow s_i - 1]$ and $c \equiv k_{v_i}[s] < s_i$.*
     - *If $s_i < u_i$ (upper bound of v_i), then there exists a transition $s \xrightarrow{c} s'$ where $s' = s[s_i \leftarrow s_i + 1]$ and $c \equiv k_{v_i}[s] > s_i$.*
- *L is a labelling function $L : S \to \mathcal{P}(AP \times C_{\mathcal{K}})$ where $L(s) = \{(a, \top) \mid a$ is an atomic proposition which is true in $s\}$.*

In definition 7, item 1 (Loops) and item 2 (Arcs) expressed how a particular variable v_i evolves. Here, item 1 and 2 express a more general property over a state (i.e. over all the variable states):

- Stability of a state (item 1): For a state s to be stable, it is necessary that no variable is attracted to a different value. In other words, all the applicable parameters have to be equal to the current value of the associated variable.

- Transitions towards a neighbour (item 2): For a state s to evolve, it is necessary for at least one variable v_i to change. To decrease (resp. increase), it is necessary that the current value of this variable v_i is greater (resp. less) to the lower (resp. upper) bound of that variable and that v_i is attracted to a lower (resp. greater) level than the current one.

Fig. 5 represents the CKS (without labelling function) for the influence graph of Fig. 3.

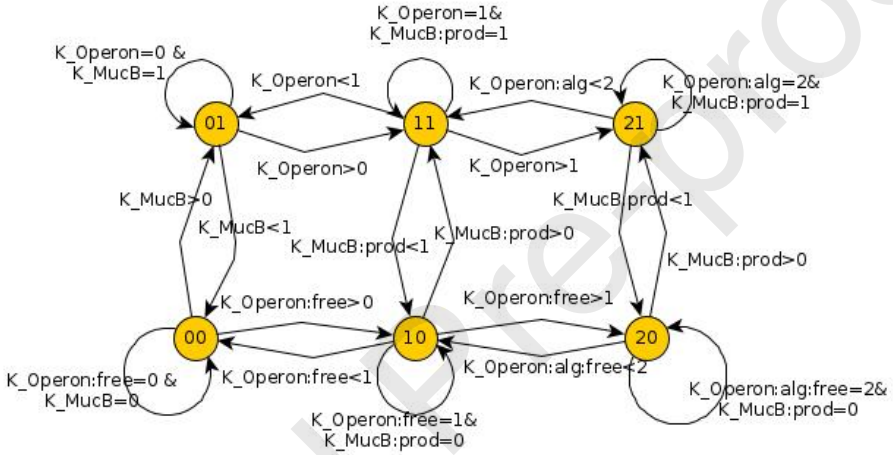


Figure 5: Constrained Kripke structure CKS_{IG} for Mucus production in *Pseudomonas æruginosa*. Character `"&"` stands for operator $\wedge$ (e.g. `K_Operon:alg:free=2 & K_MucB:prod=0` stands for $K_{Operon,\{alg,free\}} = 2 \wedge K_{MucB,\{prod\}} = 0$).

4.3. Labelling Algorithm

The aim of the labelling algorithm applied to CKS is to label each state s by couples consisting of a CTL formula φ and the constraint on the parameters so that s satisfies the formula φ. It aims at extending the labelling function L of the CKS to all Φ_{CTL}. Because the set Φ_{CTL} is infinite, the labelling algorithm focuses on labelling the states s of the CKS with sub-formulae of a target formula φ. At the end of the algorithm, for each sub-formula ψ of this target formula φ (itself included), each state s has to be labelled by (ψ, c_s^ψ) where $c_s^\psi \in C_\mathcal{K}$ is the constraint on parameters which defines *all the*

parameter settings P for which the formula ψ is satisfied in state s in the associated Kripke structure KS_P (we note $s \vDash_P \psi$).

Let us consider that one wants to model check formula $\varphi \in \Phi_{CTL}$. As usual, the labelling algorithm starts with the atomic formulae, and progressively reviews all sub-formulae (from the smallest to the biggest) and finally labels the states of the constrained Kripke structure with target formula φ. The sequel describes each of these steps considering all possible connectives.

4.3.1. Choice of Temporal connectives

The algorithm does not need to be able to handle every CTL connective explicitly, because there are several equivalences between temporal connectives as well as De Morgan's laws for logical connectives. For example, $\neg AF\varphi \equiv EG\neg\varphi$ and $\neg EF\varphi \equiv AG\neg\varphi$. Therefore, as in propositional logic, there is some redundancy among the connectives. It can be proved that only 3 temporal connectives are sufficient to express all 8 ones [35]. Several choices are possible, but here we choose EX, EU and EG (see Appendix A). For non temporal connectives, we choose $\neg$ and $\wedge$.

Notation 4.2. *Labels. For a state $s \in S$, we denote $L(s)$ the set of labels associated with s. Moreover we denote $L_c(s, \varphi)$ the constraint associated with φ in $L(s)$. The labelling function $L_c(s, \varphi)$ gives the label $\bot$ to every state and every formula unless s is already labelled by (φ, c_s^φ), and in such a case $L_c(s, \varphi) = c_s^\varphi$.*

Note that the implementation does not actually need to explicitly label all states with all subformulas of the target formula, it is just important that the labelling function returns the "default label" for states that do not yet have a label containing the subformula in question.

4.3.2. Initialisation Step

Generally the first step of a labelling algorithm consists in labelling each state with atomic properties which are true in that state. Here we also label each state by all the couples $(a, \top)$ such that the atomic proposition $a \in AP$ is satisfied in s (the satisfiability of atom a does not depend on parameter settings). Since the atom $\top$ is true in all states, we also label each states by $(\top, \top)$.

4.3.3. Procedure for Non Temporal Connectives

1. If $\varphi = \neg\psi$, repeat for each state s:

- If $(\psi, c) \in L(s)$, label s with $(\varphi, \neg c)$.
- If s is not already labelled with ψ, label s with $(\varphi, \top)$.

2. If $\varphi = \varphi_1 \wedge \varphi_2$, repeat for each state s:
   - If $(\varphi_1, c_1) \in L(s)$ and $(\varphi_2, c_2) \in L(s)$, label s with $(\varphi, c_1 \wedge c_2)$.
   - Else, do nothing.

4.3.4. Procedure for EX[φ]

Repeat for each state s:

- Seek for each state t such that $s \xrightarrow{c} t \in R$ and t is labelled by (φ, c_t^{φ}).
- Then label state s with :

$$\Big(EX[\varphi] \quad , \quad \bigvee_{t \text{ s.t. } s \xrightarrow{c} t \in R \ \wedge \ L_c(t,\varphi) = c_t^{\varphi}} (c \wedge c_t^{\varphi})\Big).$$

Note that $EX[\varphi]$ is true iff there exists *at least one* successor which is already labelled by φ. Nevertheless, because we are interested in characterising all the parameter settings for which the state s is labelled by $EX[\varphi]$, we have to consider all the successors.

4.3.5. Procedure for E[φ U ψ]

The processing of the connective EU is a little bit more subtle because one has to consider paths of length greater than one. This can be done using an iterative loop whose aim is to track all paths leading to a state where ψ is satisfied.

1. Initialisation: For each state s already labelled by (ψ, c_s^{ψ}), label the state s with $(E[\varphi U \psi], c_s^{\psi})$.
2. While there exists at least one state s such that $[\![L_c(s, E[\varphi U \psi])]\!]$ has changed[5]:
   - For each state s labelled with (φ, c_s^{φ}), one seeks for all states t such that $s \xrightarrow{c} t \in R$ and t is already labelled with $(E[\varphi U \psi], c_t^{EU})$. One labels s with:

$$\Big(E[\varphi U \psi] \quad , \quad \Big(\bigvee_{t \text{ s.t. } s \xrightarrow{c} t \ \wedge \ L_c(t, E[\varphi U \psi]) = c_t^{EU}} (c \wedge c_t^{EU})\Big) \wedge c_s^{\varphi}\Big) \quad (1)$$

[5]that is, the semantics of constraints associated with $E[\varphi U \psi]$ does change for at least one state.

The implementation of such an algorithm depends on the capability of checking a change of the semantics of a constraint. If a normal form of each constraint is maintained, this checking can be done at a syntactic level. But because of an excessive computation cost, one can also use an external tool in order to test the equivalence of both constraints:

$$[\![c_s]\!] \neq [\![c'_s]\!] \quad \Leftrightarrow \quad (c_s \Leftrightarrow c'_s) \text{ is not satisfiable.}$$

We do not detail this point, but in Section 5.2 we give some tips for such implementation mixing syntactic and semantic levels.
The Section 5.1.1 provides a proof of the termination of this algorithm as well as its correctness.

4.3.6. Procedure for EG[φ]

The subtlety of the processing of this connective remains in the fact that one has to highlight an infinite path along which the formula φ is true everywhere. Because the set of states is finite, the search of infinite paths reduces to the search of cycles sharing the same property. Thus the intuition of the next algorithm is to find the cycles passing only through states satisfying the formula φ.

1. Initialisation : For each state s already labelled by (φ, c_s^{φ}), label the state s with $(EG[\varphi], c_s^{\varphi})$.
2. While there exists at least one state s such that $[\![L_c(s, EG[\varphi])]\!]$ has changed[6]:
   - For each state s labelled by both (φ, c_s^{φ}) and $(EG[\varphi], c_s)$, one seeks for all states t such that $s \xrightarrow{c} t \in R$ and t is already labelled with $(EG[\varphi], c_t^{EG})$. Compute

$$c_s^{EG} = \left(\bigvee_{t \text{ s.t. } s \xrightarrow{c} t \in R \ \wedge \ L_c(t, EG[\varphi]) = c_t^{EG}} c \wedge c_t^{EG} \wedge c_s^{\varphi} \right) \quad (2)$$

     If $[\![c_s^{EG}]\!] \neq [\![c_s]\!]$, do change the current label $(EG[\varphi], c_s)$ by $(EG[\varphi], c_s^{EG})$. In particular if c_s^{EG} is not satisfiable ($[\![c_s^{EG}]\!] = \emptyset$), remove the label $(EG[\varphi], c_s^{EG})$ from state s.

[6] that is, the semantics of constraints associated with $EG[\varphi]$ does change for at least one state.

As for connective EU, the implementation of such an algorithm depends on the capability of checking a change of the semantics of a constraint. The Section 5.1.2 provides a proof of the termination of this algorithm as well as its correctness.

5. Results

In this section we present our main results:

- The validity of the labelling algorithm for constrained Kripke structures.
- A first prototype that implements the labelling algorithm and computes the parameter settings which satisfy the biological property.
- The application of this global approach to some biological case studies.

5.1. Validity of the Labelling Algorithm

This subsection is devoted to the validity of the global labelling algorithm. Given a CKS, two policies are possible for providing the set of parameter settings making true a given CTL formula. The first one is to enumerate each possible parameter setting $P \in \mathcal{P}$, build the associated Kripke structure, and use the classical model checking algorithm. The second one consists in using our labelling algorithm, and then deduce the set of parameter settings consistent with the formula. Intuitively, the constraints on parameters built by the algorithm are a syntactic representation of the set of parameter settings making true each sub-formula. The validity of the algorithm focuses on the semantics of these constraints. The following proofs aim to assert that the diagram in Fig. 6 is commutative.

From a CKS and a parameter setting, it is straightforward to build the Kripke structure associated to P.

Definition 12 (Kripke structure deduced from a parameter setting P)**.** *Let us consider a Constrained Kripke Structure $CKS = (S, C_{\mathcal{K}}, R, L)$ and a parameter setting P. The Kripke structure $CKS(P)$ deduced from CKS and P is the Kripke structure (S, R', L') where:*

- *S is the set of states of the system.*
- *R' is a relation between states which are present for the parameter setting P: $R' = \{(s,t) \mid \exists c \in C_{\mathcal{K}}$ s.t. $(s,c,t) \in R$ and $P \vDash c\}$.*

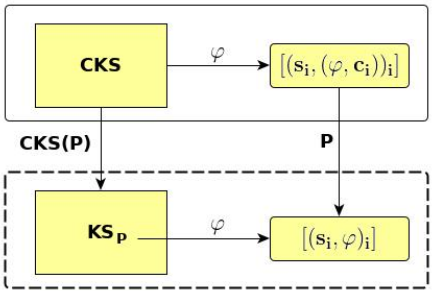


Figure 6: Commutative Diagram. Top: global labelling algorithm on CKS that labels s_i with (φ, c_i) if φ is true at s_i under constraint c_i on parameters. Bottom: classical labelling algorithm on KS_P. $CKS(P)$ is the projection of CKS on the particular parameter setting P. (s_i, φ) appears in the bottom list iff constraint c_i is true for parameter setting P.

- *L' is a labelling function $L' : \rightarrow 2^{AP}$ where $L'(s) = \{a \in AP \mid \exists c \in C_{\mathcal{K}}$ s.t. $(a, c) \in L(s)$ and $P \vDash c\}$.*

Let us remark, that, by construction $CKS(P) = KS_P$ (see Def. 7 and 12), and because of this equality and previously defined hypotheses, the Kripke structure is serial.

In order to prove the validity of the labelling algorithm, one has to show that after termination of the algorithm, if $(\varphi, c_s^\varphi) \in L(s)$, then for each parameter setting $P \in \mathcal{P}$:

- If P satisfies c_s^φ (i.e. $P \vDash c_s^\varphi$), then the formula φ is true in state s when considering the Kripke structure $CKS(P)$ (i.e. $s \vDash_P \varphi$).
- If P does not satisfy c_s^φ (i.e. $P \nvDash c_s^\varphi$), then the formula φ is false in state s when considering the Kripke structure $CKS(P)$ (i.e. $s \nvDash_P \varphi$).

In other words[7]:

$$\forall s \in S \ \Big((\varphi, c_s^\varphi) \in L(s) \quad \Leftrightarrow \quad \Big(\forall P \in \mathcal{P}, (P \vDash c_s^\varphi) \Leftrightarrow (s \vDash_P \varphi) \Big) \ \Big)$$

[7]Remember that in this formula, $\vDash$ covers the semantics of two universes: $\vDash$ denotes semantics of parameters constraints while $\vDash_P$ denotes the semantics of CTL formulae in the Kripke structure KS_P.

5.1.1. Validity of the labelling algorithm for EU connective

Lemma 5.1. *Let us consider that, at a certain stage of the algorithm, s is labelled with $(E[\varphi U\psi], c_s^1)$ and that the next step modifying this label leads to the new label $(E[\varphi U\psi], c_s^2)$. Then* $[\![c_s^1]\!] \subseteq [\![c_s^2]\!]$

Proof. At the beginning of the algorithm, all the states are labelled with $(E[\varphi U\psi], \bot)$. In other words, for each state s, the set of parameter settings satisfying the constraint associated with $E[\varphi U\psi]$ is empty.
During initialisation, each updated state s trivially satisfies the statement.
Let us suppose now that the statement is true until a stage of the algorithm. Let us consider a new update of state s.

- If it is the first time that s is labelled with $E[\varphi U\psi]$: the algorithm associates with this formula a constraint whose solutions constitute a set which can be only larger than the empty set.
- If the label of s $(E[\varphi U\psi], c_s)$ is updated by $(E[\varphi U\psi], c'_s)$:
  - If this is the first time the successor t has been taken into account to label s with the formula $E[\varphi U\psi]$ then in the disjunction of eq. 1, there exists a new option (the one considering this new successor t), leading to a possible increasing of the set of solutions: $[\![c_s]\!] \subseteq [\![c'_s]\!]$.
  - If the update is due to a change of $[\![c_t]\!]$: by recurrence hypothesis, $[\![c_t]\!] \subseteq [\![c'_t]\!]$. So $[\![c_t \wedge c_r \wedge c_s^\varphi]\!] \subseteq [\![c'_t \wedge c_r \wedge c_s^\varphi]\!]$ leading naturally to: $[\![c_s]\!] \subseteq [\![c'_s]\!]$.

This finishes the proof. □

Theorem 1. *The loop of the EU labelling algorithm terminates.*

Proof. Let us consider the vectors of sets of parameter settings $([\![L_c(s_1, E[\varphi U\psi])]\!], [\![L_c(s_2, E[\varphi U\psi])]\!], \ldots, [\![L_c(s_N, E[\varphi U\psi])]\!])$ where N is the total number of states. The set of possible vectors is large but finite.
Let us consider the partial order between 2 possible vectors: $V_1 \leqslant V_2$ iff for each coordinate $i \in [1..N]$, we have $V_1(i) \subseteq V_2(i)$.
The vector associated with the starting point of the algorithm (before the initialisation step) is the vector of empty sets. And each update of this algorithm leads to an increasing of the vector. Thus because the algorithm increases the vector, and that the set of vectors is finite, the algorithm reaches a fixed point, and the algorithm terminates. □

Lemma 5.2. *Let us consider a CKS, a parameter setting P and a given formula $E[\varphi U \psi]$. For each state s, let $(E[\varphi U \psi], c_s)$ be the label of state s computed by the EU procedure such that $P \vDash c_s$ for the first time (no previously computed $L_c(s, E[\varphi U \psi])$ are satisfied by P). Then, there exists a finite path (without cycle) in $CKS(P)$ which makes $E[\varphi U \psi]$ true in s (this path starts at s and terminate in a state validating ψ).*

Proof. The proof consists in constructing a cycle-free path in $CKS(P)$ that validates the formula $E[\varphi U \psi]$. This path is constructed inductively, starting from the end of the path, i.e. from a state validating ψ.

- Trivial during initialisation: if $P \vDash c_s^{\psi}$, the path (in $CKS(P)$) with no transition starting from state s where ψ is true ($P \vDash c_s^{\psi}$) makes true the formula $E[\varphi U \psi]$ in state s. This path does not contain any cycles.

- Let us assume the proposition is true up to a certain number of iterations and let us now consider the updating c'_s of c_s: $c'_s = L_c(s, E[\varphi U \psi])$ such that $P \vDash c_s$ for the first time (no previously computed $L_c(s, E[\varphi U \psi])$ are satisfied by P). The new label is

$$\Big(E[\varphi U \psi] \quad , \quad \Big(\bigvee_{t \text{ s.t. } s \xrightarrow{c} t \ \wedge \ t \text{ labelled by } (E[\varphi U \psi], c_t)} (c \wedge c_t \wedge c_s^{\varphi}) \Big)\Big)$$

  Since $P \vDash c'_s$, there exists at least one conjunction such that $P \vDash c \wedge c_t \wedge c_s^{\varphi}$. Thus, there exists a transition from s to t_1 in $CKS(P)$ and t_1 is labelled with $(E[\varphi U \psi], c_{t_1})$.

  - If t_1 is also labelled by $(\psi, c_{t_1}^{\psi})$ and if $P \vDash c_{t_1}^{\psi}$, then the path $s \to t_1$ is a path in $CKS(P)$ (since $P \vDash c_r$) and the path makes true the formula $E[\varphi U \psi]$ in $CKS(P)$ (since $P \vDash c_s^{\varphi}$ and $P \vDash c_{t_1}^{\psi}$).
  - In all other cases:
    * $P \vDash c$ implies that $s \to t_1$ is a path in $CKS(P)$.
    * $P \vDash c_s^{\varphi}$ implies that, inside the Kripke structure $CKS(P)$, the formula φ is satisfied in s.
    * Since $P \vDash c_{t_1}$, we can use the recurrence hypothesis which states that, when $P \vDash L_c(t_1, E[\varphi U \psi]))$ for the first time, one builts a finite path without cycle in $CKS(P)$ which makes $E[\varphi U \psi]$ true in t_1.

These three information pieces allow one to build a finite path without cycle in $CKS(P)$ which makes true $E[\varphi U \psi]$ in s.

□

Theorem 2 (Validity of the EU labelling algorithm)**.** *Let us consider a state s labelled by $(E[\varphi U \psi], c_s)$. Then:*

1. *For all $P \vDash c_s$, $s \vDash_P E[\varphi U \psi]$.*
2. *For all $P \nvDash c_s$, $s \nvDash_P E[\varphi U \psi]$.*

Proof. 1. This proof is done by the previous lemma.

2. By contradiction. Let us consider $P \nvDash c_s$ and $s \vDash_P E[\varphi U \psi]$. Then, there exists at least a path $s, t_1, t_2, \ldots, t_n$ in $CKS(P)$ such that $t_n \vDash_P \psi$ and all previous states satisfy φ. Among all these paths, we choose the shortest one which is cycle-free. Since all transitions of $CKS(P)$ are transitions of CKS, the exhibited path is also in CKS.
   After the first `while` loop of the algorithm, t_{n-1} has been labelled by $(E[\varphi U \psi], c_{n-1})$ and $P \vDash c_{n-1}$, after the second loop, t_{n-2} has been labelled by $(E[\varphi U \psi], c_{n-2})$ and $P \vDash c_{n-2}$, ... and after n loops, s has been labelled by $(E[\varphi U \psi], c_0)$ and $P \vDash c_0$.
   Since $[\![c_0]\!] \subseteq [\![c_s]\!]$ (Lemma 5.1), one deduces: $P \vDash c_0 \quad \Rightarrow \quad P \vDash c_s$.
   Contradiction.

□

5.1.2. Validity of the labelling algorithm for EG connective

Lemma 5.3. *Let us consider that, at a certain stage of the algorithm, s is labelled with $(EG[\varphi], c_s^1)$ and that after the next step (first update of a label), the same state s is labelled with $(EG[\varphi], c_s^2)$. Then* $[\![c_s^2]\!] \subseteq [\![c_s^1]\!]$.

Proof. The proof is similar to the one for Lemma 5.1. After the initialisation step of the algorithm, all states satisfying φ are labelled with formula $(EG[\varphi], c_s^\varphi)$ where $c_s^\varphi = L_c(s, \varphi)$. In other words, for each state s satisfying φ, the set of parameter settings satisfying the constraint associated with this formula is all possible parameter settings making φ true in s. Note that this labelling step is by excess: the algorithm labels states with some constraints that will be more restrictive at a later stage (perhaps until they become unsatisfiable).
Let us suppose now that the statement is true until a stage of the algorithm.

Let us consider a new update of state s.
If the label of s $(EG[\varphi], c_s)$ is updated by $(EG[\varphi], c_s^{EG})$: The update is due at least to a change of $[\![c_t]\!]$. By recurrence hypothesis, $[\![c'_t]\!] \subseteq [\![c_t]\!]$. So $[\![c'_t \wedge c \wedge c_s^{\varphi}]\!] \subseteq [\![c_t \wedge c \wedge c_s^{\varphi}]\!]$ leading naturally to: $[\![c_s^{EG}]\!] \subseteq [\![c_s]\!]$. This finishes the proof. □

Theorem 3. *The loop of the EG labelling algorithm terminates.*

Proof. Let us consider the vectors of sets of parameter settings $([\![L_c(s_1, EG[\varphi])]\!], [\![L_c(s_2, EG[\varphi])]\!], \ldots, [\![L_c(s_N, EG[\varphi])]\!])$ where N is the total number of states. As previously, the set of possible vectors is finite and we consider the partial order between 2 possible vectors: $V_1 \leqslant V_2$ iff for each state s_i, we have $V_1(s_i) \subseteq V_2(s_i)$.
The vector after the initialisation step is the vector where each coordinate corresponding to a state satisfying φ is equal to the set of parameter settings Q such that $s \vDash_Q \varphi$. And each update of this algorithm leads to a decreasing of the vector. Thus because each step of the algorithm decreases the vector, and that the set of vectors is finite, the algorithm reaches a fixed point, and the algorithm terminates. □

Theorem 4 (Validity of the EG labelling algorithm)**.** *Let us consider a state s labelled by $(EG[\varphi], c_s)$. Then:*

1. *For all $P \vDash c_s$, $s \vDash_P EG[\varphi]$.*
2. *For all $P \nvDash c_s$, $s \nvDash_P EG[\varphi]$.*

Proof. 1. Let us first prove that for all $P \vDash c_s$, $s \vDash_P EG[\varphi]$. First of all we know that $P \vDash c_s^{\varphi}$ (c_s^{φ} is in all options of the disjunction c_s) and thus $s \vDash_P \varphi$.
Since $P \vDash c_s$, there exists at least a sub-formula of the form $c \wedge c_t \wedge c_s^{\varphi}$ which is true in s for the parameter setting P. Thus there exists a successor t_1 (in $CKS(P)$) which is labelled by $(EG[\varphi], c_{t_1})$ and $P \vDash c_{t_1}$. We deduce $t_1 \vDash_P \varphi$.
With the same reasoning, we can construct a sequence of states $s, t_1, t_2, \ldots, t_n$ corresponding to a path in P and where each state satisfies φ.
Because the set of states is finite, we can exhibit a circuit with the same properties. Thus $s \vDash_P EG[\varphi]$.

2. By contradiction. Let us consider $P \nvDash c_s$ and $s \vDash_P EG[\varphi]$.
By the semantic definition of EG connective, there exists in $CKS(P)$ a circuit $s = s_0, s_1, s_2, \ldots, s_n$ such that $s_i \vDash_P \varphi, \forall i \in [0..n]$.

Let us now show that at each step of the algorithm, for each state s_i of the previously described circuit, $L_c(s_i, EG[\varphi])$ is satisfied by P (Cf. Eq. 2 in Subsection 4.3.6), in other words, $P \vDash c_{s_i}^{EG}$:

- Initialisation labels all these states s_i with $(EG[\varphi], c_{s_i}^{\varphi})$, we have $P \vDash c_{s_i}^{\varphi}$.
- At each step of the algorithm, the constraint $c_{s_i}^{EG}$ is possibly updated with a disjunction of expressions of the form $c \wedge c_t \wedge c_s^{\varphi}$ (Cf. Eq. 2). One of these conjunctions concerns the transition from s_i to s_{i+1} (the successor of s_i in the circuit). Since s_i belongs to a circuit in $CKS(P)$ in which all states satisfy φ, the successor s_{i+1} of s_i is such that $P \vDash c$ (the circuit is in $CKS(P)$), $P \vDash c_{s_i}^{\varphi}$ (φ is satisfied in s_i) and $P \vDash c_{s_{i+1}}^{EG}$ (by induction). Finally we have $P \vDash c_{s_i}^{EG}$.

After convergence of the loop (item 2 of Subsection 4.3.6), c_s which is the constraint associated with s and formula $EG[\varphi]$, that is $c_s = L_c(s, EG[\varphi])$, is equal to the last computed c_s^{EG}. Then $P \vDash c_s$. Contradiction.

□

5.2. First Prototype

5.2.1. Overview

We have implemented a first prototype for parameter identification using the global approach with constrained Kripke structure[8]. This prototype takes the same inputs as our `TotemBioNet` tool. Thus the set of parameter symbols associated with the influence graph is deduced in the same way as `TotemBioNet` does. The labelling algorithm uses an associative map: each state s is associated with the list of couples $[(\varphi_i, c_s^{\varphi_i})_i]$. The CTL formula φ to model-check is handled recursively according to its sub-formulae. Each step of the labelling algorithm adds labels $(\varphi_i, c_s^{\varphi_i})$ to some states where φ_i is a sub-formula of φ. At the end, to compute the parameter settings which satisfy φ, the constraints of states labelled with φ must be examined.

[8] See `https://gitlab.com/totembionet/cks-bionet`.

5.2.2. Constrained Kripke Structure

The transitions $s \xrightarrow{c} s'$ of CKS_{IG} (see Def. 11) are built "on the fly". For each state $s \in S$, next possible states are enumerated: s itself, and states $s' \in S$ such that there exists a unique $i = 1..n$ such that $s'_i \neq s_i$ and $s'_i = s_i + 1$ (or $s'_i = s_i - 1$). If one of these states is appropriate to check the current formula φ, in other words, if it is labelled with a subformula required to ensure the validity of φ, transition $s \xrightarrow{c} s$ (or $s \xrightarrow{c} s'$) is considered. For example, for $EG[\varphi]$ appropriate states are states already labelled with φ. The constraint associated with the transition (s, s') is built according to the relation of CKS (see Subsection 4.2). This requires to first compute the applicable parameter on state s.

5.2.3. Constraint Handling

To improve efficiency, the labelling algorithm eliminates "on the fly" the transitions whose constraints are syntactically inconsistent with the constraint under construction. More precisely, the internal data structure maintains a disjunctive form: they are disjunctions of conjunctions of atoms of the form: $k < n, k > n, k = n$ where k is a parameter symbol and n is a value in its domain. Each time a new atom is added, syntactic checks and simplifications are done.

5.2.4. Stop Condition for EU/EG

To stop the loop for model checking a EU or EG formula φ, one has to check if the current constraint of a state labelled with φ, and the new disjunction built using all possible successors, represent the same solutions. To do this, we call the SMT-solver *yices*[36] on a particular formula, which is described below.

Let *cur* be the current disjunction and *new* the new constructed disjunction. If the formula $\neg((cur \vee new) \iff cur)$ is SAT, this means that at least one parameter setting satisfies *new* and does not satisfy *cur*. The loop in the EU algorithm must therefore be performed once more. For EG connective, we use formula $\neg((cur \vee new) \iff new)$, and the same reasoning applies.

Note that to improve efficiency, if *cur* and *new* formulae are syntactically equal, the SMT solver is not launched.

5.2.5. Solving the Constraint System

After the labelling step, if one of the initial states has not been labelled with the CTL formula φ to model check, there is no solution to our param-

eter identification problem. Otherwise, the parameter settings satisfying φ are the solutions of the constraint system which contains all the constraints associated with φ, for all states of the constrained Kripke structure.

To solve this constraint system, we use *Multi Valued Decision Diagrams* (MDDs), and more precisely, the *Colomoto mddlib* library designed by A. Naldi (`https://github.com/colomoto/mddlib`) in order to find stable states in biological systems [9]. We already used this library in previous work in order to compute intersection of parameter setting sets from different environments [11]. To build the MDD of the constraint system, we get for each state s_i the constraint $c_i = L_c(\varphi, s_i)$. These constraints are memorised as disjunctions of conjunctions. We naturally use *AND* combination operator on MDDs for conjunctions and *OR* combination operator on MDDs for disjunctions. Fig. 7 illustrates a small MDD for the conjunction of two atoms and Fig. 8 is an extract of the constraints calculated for *Pseudomonas æruginosa*.

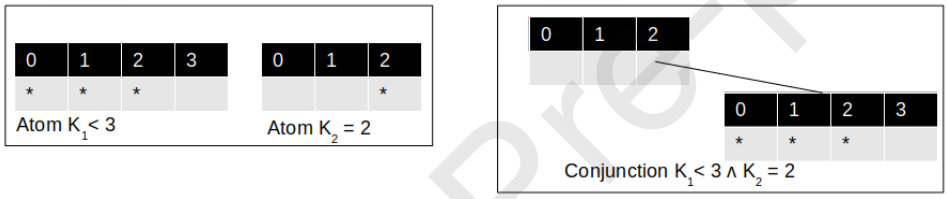


Figure 7: On the left, atoms $K_1 < 3$ (K_1 in [0..3]) and $K_2 = 2$ (K_2 in [0..2]). On the right, the conjunction $K_1 < 3 \wedge K_2 = 2$. * denotes the reference of a terminal MDD node and indicates that the corresponding value exists.

Example of constraints obtained for Pseudomonas æruginosa. The formula φ presented in Subsection 2.3, $\varphi \equiv ((Operon = 0) \Rightarrow AG[\neg(Operon = 2)]) \wedge ((Operon = 2) \Rightarrow AG[\neg(Operon = 0)])$ is translated as

$(\neg(Operon = 0) \vee \neg E[\top\ U\ Operon = 2]) \wedge$
$(\neg(Operon = 2) \vee \neg E[\top\ U\ Operon = 0])$

using logical equivalences and CTL equivalences presented in Appendix A. The constraints computed by the tool for states 10 and 21, written in concrete syntax, are given in Fig. 8. The constraint is true for state 10 because $Operon = 1$ on this state thus φ is trivially true. The parameter setting P_1 (see Fig. 3) is a solution of the constraint given line 13. On the contrary, the parameter setting P_2 does not validate any of the constraints since when $K_{MucB,\{prod\}} = 1$, $K_{Operon,\{alg\}}$ must be greater than 1 (lines 11 & 14 of Fig. 8).

```
1 [10:
2   {(((! (operon=0))|(! E [True U (operon=2)]))&
3    ((! (operon=2))|(! E [True U (operon=0)]))),
4        [or: TRUE]}
5 ]
6 [21:
7   {(((! (operon=0))|(! E [True U (operon=2)]))&
8     ((! (operon=2))|(! E [True U (operon=0)]))),
9      [or: [and:[K_operon:alg=1, K_mucB:prod=0, K_operon:alg:free=1]],
10          [and:[K_operon:alg=1, K_mucB:prod=0, K_operon:alg:free=2]],
11          [and:[K_mucB:prod=1,  K_operon:alg=1]],
12          [and:[K_operon:alg=2, K_mucB:prod=0, K_operon:alg:free=1]],
13          [and:[K_operon:alg=2, K_mucB:prod=0, K_operon:alg:free=2]],
14          [and:[K_mucB:prod=1,  K_operon:alg=2]]]}
15 ]
```

Figure 8: Constraints associated with state 10 and 21 computed for *Pseudomonas ærugi-nosa* and formula φ of Subsection 2.3 (given in the concrete syntax of the tool, character "|" stands for $\vee$). For each state, the first part is the formula, the second part (under brackets) is the constraint in disjunctive normal form.

5.3. First Experiments

5.3.1. Stable States

We first tested if our CKS model checking algorithm allows the verification of presence of two stable states on a simple network with a cyclic topology. For this topology, we have a theoritical understanding of the behavior (two stable states, see for example [37]). However, our aim is to check if this new algorithm gives the expected results, and to compare its performance with our previous approach by enumeration, when increasing the number of variables. We consider simple networks containing only a circuit of n Boolean variables: variable v_1 activates v_2 through multiplex m_1, v_2 activates v_3 through multiplex m_2, ... and v_n activates v_1 through multiplex m_n ($\forall i \in [1..n], \varphi_{m_i} \equiv v_i \geqslant 1$). In such a case, if for each $i \in [1, n]$, $K_{v_i,\{\}} = 0$ and $K_{v_i,\{m_{i-1}\}} = 1$ (with $m_0 = m_n$), the circuit is said functional and the system presents two stable states which are $(0, 0, ..., 0)$ and $(1, 1, ..., 1)$. Of course, each variable is under the control of only one another variable, that leads to 2 parameters for each variable, and then $(2^2)^n$ parameter settings. The property to be verified is then:

$$s_0 \rightarrow AG[s_0] \quad \wedge \quad s_1 \rightarrow AG[s_1]$$

where s_0 (resp. s_1) is the formula characterising the state where each variable is set to 0 (resp. 1). The formula is transcribed into:

$$\varphi_{stable} \equiv (\neg s_0 \mid (\neg E[\top U(\neg s_0)])) \quad \wedge \quad (\neg s_1 \mid (\neg E[\top U(\neg s_1)]))$$

Table 1 shows the results we obtained on this simple example which has a very low in-degrees and is very symetric.[9]

As expected, the enumeration approach is very sensitive to the growth of the parameter setting number (by its very nature). The CKS model checking should be sensitive to the state number growth, but it is not. This is due to the specific form of the formula φ_{stable} which involves the two states s_0 and s_1, and more importantly their negation inside EU. The different steps of the algorithm are:

- s_0 and s_1 are respectivelly labelled by $(s_0, \top)$ and $(s_1, \top)$, and each state different from s_0 (resp. s_1) is labelled by $(\neg s_0, \top)$ (resp. $(\neg s_1, \top)$).
- Each state different from s_0 (resp. s_1) is labelled by $(E[\top U(\neg s_0)], \top)$ (resp. $(E[\top U(\neg s_1)], \top)$). These labels will never change because of Lemma 5.1.
  s_0 and s_1 are labelled respectivelly by $(E[\top U(\neg s_0)], c_0)$ and $(E[\top U(\neg s_1)], c_1)$ where c_0 and c_1 is the disjunction of all the constraints on arcs issue from s_0 and from s_1. These two labels won't change because the label of the successors of s_0 and s_1 do not change.
- The labelling with $\neg E[\top U(\neg s_0)]$ and with $\neg E[\top U(\neg s_1)]$ is straightforward: s_0 is labelled by $(\neg E[\top U(\neg s_0)], \neg c_0)$ and similarly for s_1.
- Then each state different from s_0 (resp. s_1) is labelled by $(\neg s_0 \mid (\neg E[\top U(\neg s_0)])), \top)$ (resp. $(\neg s_1 \mid (\neg E[\top U(\neg s_1)])), \top)$.
- Finally, all states different from s_0 and s_1 are labelled by $(\varphi_{stable}, \top)$ and s_0 is labelled by $(\varphi_{stable}, \neg c_0)$ (and similarly for s_1).

At the end of the model checking, all states except s_0 and s_1 are labeled with $\top$ and the solution set of $\neg c_0 \wedge \neg c_1$ is easilly computed with the MDD library.

5.3.2. Repressilators

A repressilator is a genetic regulatory network consisting of at least one feedback loop with at least three genes, each expressing a protein that represses the next gene in the loop. Such construction has been used for understanding natural biological rythms and also in synthetic biology [38, 39].

[9] Experiments were done on a six cores CPU at 3.00GHz, with 32 Gio of memory.

Table 1: Execution time for constrained model checking of a positive loop allowing multi-stationarity. The time with the symbol * has been estimated considering that the number of parameter settings treated in a second is constant.

	n=5	n=7	n=10	n=20
# states	32	128	1024	1,048,576
# parameter settings	1024	16,384	1,048,576	1,099,511,627,776
Enumeration approach	6s 061 ms	1min 48s	2h 42min	581 years*
CKS Model checking	48 ms	72 ms	248 ms	4min 4s

Table 2: Experimental results for n-dimensional repressilator ($n = 3$ to 8).

	n=3	n=4	n=5	n=6	n=7	n=8
# states	12	24	48	96	192	384
# parameter settings	576	2,304	9,216	36,864	147,456	589,824
Enumeration approach	3s 904ms	16s 139ms	1m 6s	4m 47s	20m 20s	Time out > 1h
Average time for 1 setting	69ms	70ms	72ms	78ms	83ms	86ms
CKS Model checking	62ms	481ms	5s 134ms	59s 295ms	56min 13s	Time out > 1h
Time for Yices	48ms	464ms	2s 567ms	16s 538ms	34min 10s	Time out > 1h

We consider here a n-dimensional repressilator. Recently, the conditions for periodic oscillations in 4-dimentional repressilators have been studied in [40]. In this work, the authors consider all 4-dimensional repressilators under the following conditions: each variable has at least one successor, the influence between variables are inhibitions and only the variables having 2 sucessors can have a domain with 3 levels (0,1 and 2), the other variables are boolean.

We have evaluated our CKS model checking on the repressilator proposed in [40] that has a periodic attractor that passes through the state 1000 (see Fig. 3-Left and 4 in [40]). Using this repressilator, we have built a family of repressilators of increasing size by adding variables to the cycle that contains only the threshold "1", see Fig. 9 for the initial 4-dimensional repressilator and its extension to n variables. When considering the repressilator with n variables, the oscilation passes through the state $s_{osc} = 1000...0$ such that variable G_0 is expressed at level 1 and all other variables are not expressed (level 0). This oscillation is expressed by the CTL formula $s_{osc} \rightarrow EX(EF(s_{osc}))$ where s_{osc} is also, by abuse of notation, the characteristic formula of state s_{osc}. This formula is automatically transcribed as

$$\varphi_{osc} \equiv \neg s_{osc} \mid (EX(E[\top U s_{osc}])) \ .$$

Table 2 shows that the CKS model checking behaves well until $n = 6$. Between $n = 6$ and $n = 7$, there is a large gap in the execution times. This is mainly due to the number of steps of the EU algorithm. In fact, the formula to check if the constraints of the states labeled with $E[\top U s_{osc}]$ have

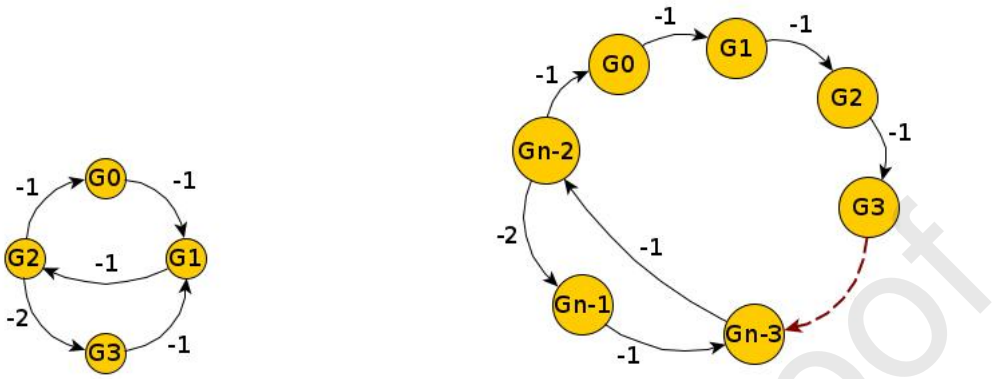


Figure 9: (Left) Influence graph of the 4-dimentional repressilator from [40]. (Right) influence graph of the n-dimensional repressilator. Red dashed arrow represents the part of the loop which grows.

changed grows, and therefore the execution times for *Yices* also grow. Note also that φ_{osc} and φ_{stable} (see example 5.3.1) both refer to specific states. But in φ_{osc} the EU operator is applied to s_{osc}, while in φ_{stable} it is applied to $\neg s_0$. Thus in the first step of the EU algorithm, the only state labeled with $(E[\top U s_{osc})], \top)$ is s_{osc}. Then, the `while` loop of the EU procedure labels all the states from near to far. Thus many steps are required to model-check the formula, while for φ_{stable}, only two steps in the `while` loop are necessary.

Table 2 also shows that the execution time of `TotemBioNet` is essentially sensitive to the number of parameter settings (the average time for 1 setting, line 4 in Table 2, remains in the same range).

Let us now consider two more realistic biological examples.

5.3.3. Pseudomonas æruginosa *in the presence of Calcium*

The small biological system *Pseudomonas æruginosa* used as an example throughout this article, becomes more significant when considering the presence or absence of calcium. In fact, calcium plays an important role in the condition of the lungs of patients with cystic fibrosis. In [11], we introduced the concept of *environment variables* to find all settings that are compatible with the dynamic properties in different environments.

Fig. 10 illustrates the influence graph of *Pseudomonas æruginosa* in the presence of calcium, with the new parameters associated with variable *Operon* which has now 3 predecessors.

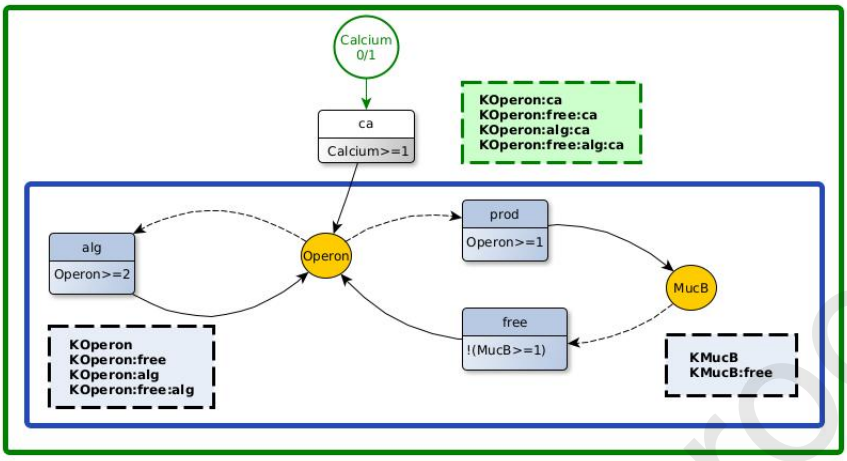


Figure 10: Influence graph of the *Pseudomonas æriginosa* in the presence of Calcium. Calcium is an environment variable (in green). *Operon* depends on 4 more parameters (in green).

The temporal properties that the model must satisfy can be translated into CTL as:

$$p_1 \equiv (Calcium = 0) \rightarrow \left(\begin{array}{l} (Operon = 0 \rightarrow AG[\neg(Operon = 2)]) \wedge \\ (Operon = 2 \rightarrow AG[\neg(Operon = 0)]) \end{array} \right)$$
$$p_2 \equiv (Calcium = 1) \rightarrow AF[AG[operon = 2]];$$

Formula p_1 represents the stable states property presented in Subsection 2.3. Formula p_2 expresses that in the presence of a very high concentration of calcium-ion in the cell environment, non-mucoid bacteria become mucoid and mucoid bacteria remain mucoid.

The constrained Kripke structure associated with this system has 3 variables (*Operon*, *Calcium* and *MucB*) and 12 states. There are 8 parameters whose domain is [0..2] and 2 parameters whose domain is [0..1]. Then we have to consider $3^8 \times 2^2 = 26,244$ parameter settings. The enumerative approach leads to a completion of the computation in 13min 4s (728 parameter settings selected) whereas the constrained model checking is completed in only 22s 36ms.

We then evaluated our prototype on a more complex biological system: the cell cycle of mammals.

5.3.4. Check Points in Cell Cycle

The cell cycle denotes the biological system controlling a series of events leading to both correct DNA duplication of a cell (synthesis or S phase) and its division into two genetically identical daugther cells (mitosis or M phase). Gap phases $G1$ and $G2$ lie respectively before S and M. Progression through the cell cycle is driven by Cyclins/Cyclin-dependent kinases complexes (Cyc/Cdks) and their inhibitors known as ennemies. A 5-variables cell cycle model has been designed in [1]:

- *sk* is the abstraction of both complexes CycE/Cdk2 and CycH/Cdk7, known as starting kinases.
- *a* and *b* respectively represent CycA/Cdk1 and CycB/Cdk1.
- *en* is the abstraction of the main Cyc/Cdks ennemies: the anaphase-promoting complex APC/Cdh1, cyclin-kinase inhibitors p21 and p27, and Wee1 protein.
- The variable *ep* is the anaphase-promoting complex APC/Cdc20, which is a Cyc/Cdks ennemy involved in mitosis exit and so-called exit protein.

Fig. 11 shows the influence graph of this 5-variables model. The detail of regulations between variables are described in [1].

The proper functioning of a cell cycle depends on a number of *checkpoints* [41]. A checkpoint is a stage at which the cell examines internal and external cues and "decides" whether or not to move forward with division. The main checkpoints are the G1 checkpoint at the G1/S transition and G2 checkpoint, at the G2/M transition. It has been characterized in [41] that a checkpoint between two phases P_1 and P_2, should ensure that none of the possible first transitions of P_2 can be performed before one of the transitions of P_1 (so, P_2 can only start when all the transitions of P_1 have occurred). Thus in order to formalize some necessary conditions of G1/S transition and G2/M transition, we need both *(i)* a description of S and M phases in terms of possible paths (CTL formulae) and *(ii)* characteristic states of phases G1 and G2 on which the checkpoint condition can be evaluated.

For a phase to run smoothly, the different variables must evolve sequentially. Nevertheless the right order of events corresponding to the evolutions of the different variables is sometime fuzzy for biologists. For example, during M phase, one observed a decrease of variable a and an increase of *ep* and

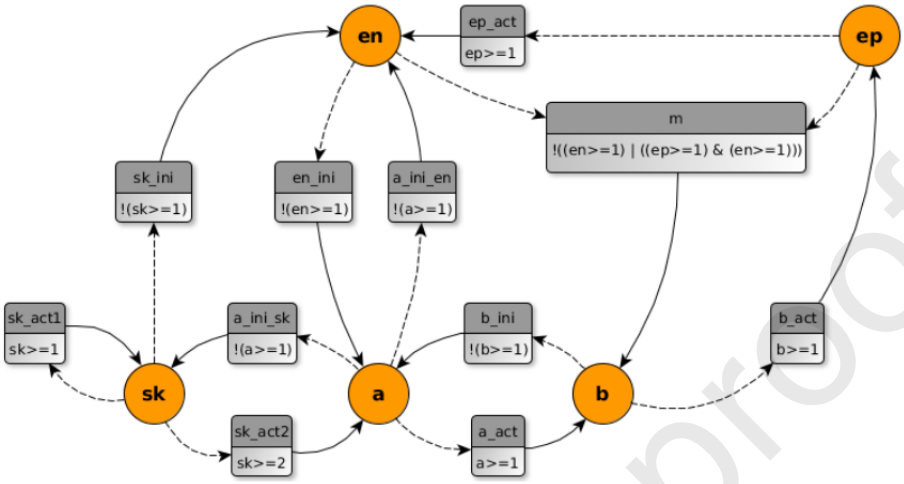


Figure 11: A 5-variables influence graph of the mammalian cell cycle, from [1].

en. Considering all possible orders of these incrementations/decrementations which are biologically plausible, the M phase can be characterised by a CTL formula using nested EX:

$$\Phi_M \equiv \left(\begin{array}{lllllllllll} & (EX[& \mathbf{(en{=}1)} & \wedge & EX[& (a=0) & \wedge & EX[& (ep=1) &]]]) \\ \vee & (EX[& \mathbf{(en{=}1)} & \wedge & EX[& (ep=1) & \wedge & EX[& (a=0) &]]]) \\ \vee & (EX[& \mathbf{(a{=}0)} & \wedge & EX[& (en=1) & \wedge & EX[& (ep=1) &]]]) \\ \vee & (EX[& \mathbf{(ep{=}1)} & \wedge & EX[& (en=1) & \wedge & EX[& (a=0) &]]]) \end{array} \right)$$

Note that there is some knowledge on the order of evolution of the variables. For example, if one first observes a decreasing of a ($a = 0$), then one has first an increasing of en ($en = 1$) before an increasing of ep ($ep = 1$).
Following the same idea, the phase S can be translated into the CTL next formula:

$$\Phi_S \equiv \left(\begin{array}{lllllllllllll} & (EX[& (a=1) & \wedge & EX[& (sk=1) & \wedge & EX[& (en=0) & \wedge & EX[(sk=2)]]]]) \\ \vee & (EX[& (sk=1) & \wedge & EX[& (a=1) & \wedge & EX[& (en=0) & \wedge & EX[(sk=2)]]]]) \\ \vee & (EX[& (sk=1) & \wedge & EX[& (en=0) & \wedge & EX[& (a=1) & \wedge & EX[(sk=2)]]]]) \\ \vee & (EX[& (a=1) & \wedge & EX[& (en=0) & \wedge & EX[& (sk=1) & \wedge & EX[(sk=2)]]]]) \\ \vee & (EX[& (sk=1) & \wedge & EX[& (a=1) & \wedge & EX[& (sk=2) & \wedge & EX[(en=0)]]]]) \\ \vee & (EX[& (sk=1) & \wedge & EX[& (sk=2) & \wedge & EX[& (a=1) & \wedge & EX[(en=0)]]]]) \\ \vee & (EX[& (a=1) & \wedge & EX[& (sk=1) & \wedge & EX[& (sk=2) & \wedge & EX[(en=0)]]]]) \\ \vee & (EX[& (en=0) & \wedge & EX[& (a=1) & \wedge & EX[& (sk=1) & \wedge & EX[(sk=2)]]]]) \\ \vee & (EX[& (en=0) & \wedge & EX[& (sk=1) & \wedge & EX[& (a=1) & \wedge & EX[(sk=2)]]]]) \end{array} \right)$$

Moreover, it has been shown in [41] that there are two states, χ_{G1} at the begining of G1 and χ_{G2} at the begining of G2, through which the system must pass. These states are considered as characteristic of the phases and

they can be expressed by logical formulae:

$$\chi_{G1} \equiv (sk = 0 \wedge ep = 0 \wedge a = 0 \wedge b = 0 \wedge en = 1)$$
$$\chi_{G2} \equiv (sk = 0 \wedge ep = 0 \wedge a = 1 \wedge b = 1 \wedge en = 0)$$

Thus, a necessary condition for ensuring the checkpoint property for the transition G2/M is that it must not be possible to perform the M phase from the characteristic state χ_{G2} of G2 (by short-circuiting the events in phase G2):

$$G2/M \equiv (\chi_{G2} \rightarrow \neg\Phi_M)$$

In a similar manner, a necessary condition for ensuring the checkpoint property for the transition G1/S is that it must not be possible to perform the S phase from the characteristic state χ_{G1} of G1 (by short-circuiting the events in phase G1):

$$G1/S \equiv (\chi_{G1} \rightarrow \neg\Phi_S)$$

These two checkpoints properties are very important for the functioning of cell cycle but the cell cycle can also be defined by its cyclic property. When considering this cycle property, we only take into account the capability of the system to return to the first state after having gone through all the phases of the cycle. This cyclic property is translated into CTL using EF and EX connectives:

$$\Phi_{cyclic} \equiv \begin{pmatrix} sk = 0 \\ \wedge \quad ep = 0 \\ \wedge \quad a = 0 \\ \wedge \quad b = 0 \\ \wedge \quad en = 1 \end{pmatrix} \rightarrow EX \left[EF \left[\begin{array}{rl} & sk = 0 \\ \wedge & ep = 0 \\ \wedge & a = 0 \\ \wedge & b = 0 \\ \wedge & en = 1 \end{array} \right] \right]$$

The execution time of the enumerative approach depends mainly on the number of parameter settings. Although some parameter values of this model have been defined using biological knowledge, there are still 2 unknown parameters whose domain is [0..2], and 16 unknown parameters whose domain is [0..1] to consider. Thus there is $3^2 \times 2^{16} = 589,824$ parameter settings to consider.

Of course, the number of parameter settings is the same for both properties (checkpoints and cyclic). As a consequence, the computation time is approximatively the same when using the enumerative approach: It takes 1h

and 40 minutes. Note that the parameter settings selected are not the same for both properties: 206,592 have been selected for the checkpoints property whereas 373,751 have been selected for the cyclic one. When working with the constrained model checking, the exectution time mainly depends on the temporal connectives present in the CTL formula to check. Thus the execution time for both properties (checkpoints and cyclic) are not expected to be the same: The parameter identification problem for the checkpoints property takes 250ms and the one for the cyclic property takes 2 minutes and 20 seconds. Indeed, the translation of EF connective involves EU connectives and the labelling algorithm for EU is more complex than the one for EX.

This case study involves EU connective as for the repressilator example (see Section 5.3.2). Here the constrained model checking is more effective than the enumeration approach. In fact, the ratio between the number of parameter settings and the number of states is large: there are $589,824$ parameter settings and only 48 states. Thus, the execution time of CKS model checking is not dramatically affected because there are a *small* number of states, while the execution time of the enumeration approach is strongly affected because the number of parameter settings is *large*.

6. Conclusion

In this article, we have designed a model checking algorithm that considers a target formula for which we want to know exhaustively all models that satisfy this formula. To achieve this, we have developed a labelling algorithm whose answer is not simply yes or no: for each state s, it constructs the conditions on parameters under which the target formula φ is satisfied. Such conditions characterise the members of the family of parameter settings that make φ true in s.

For this purpose, the notion of a constrained Kripke structure was introduced and we have shown that it helps to identify the parameters of a discrete model of biological systems. When the set of parameter settings is too enormous, the enumeration approach which builds successivelly each model and launches a model checking procedure takes to much time, even if a *symbolic* model checker as NuSMV is used. The intuitive reason is that the combinatorics due to parameter settings is much more numerous than the state space which is symbolically represented in symbolic model checkers. For parameter identification, we have a choice between a symbolic representation of states or a symbolic representation of parameterisations. We think

that, without other knowledge, the best choice is to symbolically represent the set whose combinatorics is the greatest. Of course under other assumptions (symetry for example), the situation would be to evaluate. Thus, when the brute force method cannot be used because of the enormous number of possible parameter settings, the constrained Kripke structure representation combined with the model checking presented in this article can help to find the parameter settings that lead to dynamics that are compatible with the dynamical behaviors expressed in temporal logic. This has been made possible by two important properties of our modelling framework: all models in our family share the same states, and their transitions can be derived by constraints on the values of the system parameters. In this way, the atomic formulae are the same regardless of the value of the parameters.

This labelling algorithm has been implemented in our `TotemBioNet` platform, which is designed to help modelers of regulatory biological networks to find all parameter settings that are compatible with known behavior. We compared its performances with those of `TotemBioNet` that works by enumerating all the Kripke structures associated with all possible parameter settings. In some cases, model checking the constrained Kripke structure is more efficient than enumerating and running a classical model checker for each Kripke structure, because enumeration takes too much time.

Unfortunately, the efficiency of the algorithm depends drastically on the target temporal formula, and in some cases the proposed algorithm takes more time to finish than the enumerative approach, or even takes so much time that the authors stopped the run before obtaining the results, see Appendix B and Appendix C. To be more precise, the processing of the EU connective converges in a reasonable time, whereas the convergence of the EG connective processing is slower. Both algorithms (for EU and for EG) seem similar, but differ in the initialisation.

In the EU case, during initialisation, only states which satisfy ψ are labelled with $E[\varphi U \psi]$. In the EG case, the algorithm first supposes that all states satisfying φ also satisfy $EG[\varphi]$ and then makes these conditions more and more restrictive (see Theorem 3). Thus, when EG is true at least for one parameter setting, all states are labelled with $EG[\varphi]$ during the initialisation step, and they all need to be considered.

Improvement of the prototype is envisioned: launching an external tool for the stop criterion of the loop in the processing of EU and EG connectives is time-consuming. It could be replaced by the semantic test with representation of set of parameter settings by MDD. This opens also the perspective

of replacement of all the constraints by MDD representing sets of parameter settings. Nevertheless, the tool can already be very useful as an initial filter for enumerating parameterisations. If we have a CTL formula on which our new model checking algorithm does not converge quickly enough, it is possible to write other formulae that are logical consequences of the initial formula. If these formulae are sufficiently simple, we can assume that the model checking algorithm will converge on the constrained Kripke structure, and this will already allow us to eliminate a large proportion of the parameter settings to be rejected.

Appendix A. Equivalence between CTL operators

Among the 8 temporal connectives, we choose EX, EU and EG. The other ones are translated in the following way (see [35] for proof of equivalence):

- $AX[\phi] \equiv \neg EX[\neg\phi]$
- $AF[\phi] \equiv \neg EG[\neg\phi]$
- $AG[\phi] \equiv \neg EF[\neg\phi] \equiv \neg E[\top\ U\ \neg\phi]$
- $EF[\phi] \equiv E[\top\ U\ \phi]$
- $A[\phi\ U\ \psi] \equiv \neg(E[\neg\psi\ U\ (\neg\phi \wedge \neg\psi)] \vee EG[\neg\psi])$

Appendix B. Negative loops

We have tried to test cyclic behaviors on artificial simple negative loops: as for positive loops, we consider simple networks containing only a circuit of n Boolean variables: variable v_1 activates v_2 through multiplex m_1, v_2 activates v_3 through multiplex m_2, ... but v_n *inhibits* v_1 through multiplex m_n ($\forall i \in [1..n-1], \varphi_{m_i} \equiv v_i \geqslant 1$ and $\varphi_{m_n} \equiv \neg(v_n \geqslant 1)$). In such a case, if for each $i \in 1..n$, $K_{v_i} = 0$ and $K_{v_i,m_{i-1}} = 1$ (with $m_0 = m_n$), the circuit is said functional, and the system presents an oscillatory behavior. As previously, each variable is under the control of only one another variable, that leads to 2 parameters for each variables, and then $(2^2)^n$ parameter settings. The property to be verified is then:

$$s_0 \rightarrow AX[AF[s_0]] \quad \equiv \quad \neg s_0 \vee \neg EX[EG[\neg s_0]]$$

where $s_0 = (0, 0, 0, \ldots, 0)$.

For a negative loop of size $n = 5$, the model is controlled by 10 parameters, leading to 1024 possible parameter settings. The constrained Kripke structure has 32 states. The enumeration policy coupled with usual model checking completes the job in 6s 242ms whereas the constrained model checking needs 7s 338ms. Here, our approach does not improve the search of the exhaustive set of solutions. Moreover when looking at the times spent to treat each of the connectives, we observed that the processing of the EG connective is the most time consuming.

Appendix C. Metabolism in aerobic Environment

A highly abstract formal model of eukaryote metabolism regulation has been described in [30, 26]. The model showed that the main high-level interactions between metabolic pathways are sufficient to produce the Warburg/Crabtree metabolic shift and the respiration/fermentation balance. Certain molecular elements are crucial (0^2, $NADHPH/NAD^+$, $NADH/NAD^+$, ...) but an abstract description of the major pathways (glycolysis, Krebs, fermentation, ...) is, in fact, the proper level of description, simply because we study interactions between pathways rather than internal functioning of the pathways themselves. The qualitative model of metabolic regulation includes both exchange metabolites and abstract representations of the major metabolic pathways. This case study focuses on an environment where the availability of nutrients for cells is sufficient to produce energy *via* the oxidative respiration, and does not use fermentation process ($FERM$ variable). In other terms, whatever the initial value of the variable $FERM$, there exists at least one path leading to $FERM = 0$ and from there, $FERM$ stays forever equal to 0. This knowledge is translated as follows in CTL:

$$NoFerm \quad \equiv \quad EF\ [\ EG\ [\ FERM = 0\ ]\ ]$$

The model contains 10 variables and almost all parameters are hand-identified thus it remains only 360 parameter settings to test. As the number of parameter settings is small, the enumerative approach takes a short time to compute the 297 valid parameter settings. Conversely, the constraint based model checking does not compute in a reasonable amount of time. This is due to the large number of states but firstly to the EF and EG connectives. They involve EU connectives and the labelling algorithm for EU implies a `while` loop until the semantics of constraints associated with the EU formula does not change any more (see Subsection 4.3.5).

References

[1] D. Boyenval, G. Bernot, H. Collavizza, J.-P. Comet, What is a cell cycle checkpoint? the TotemBioNet answer, in: Proceedings of the 18th International Conference on Computational Methods in Systems Biology (CMSB), Vol. 12314 of LNCS, online, 2020, pp. 362–372.

[2] J. J. Tyson, B. Novak, Temporal organization of the cell cycle, Current Biology 18 (2008) R759–R768. doi:10.1016/j.cub.2008.07.001.

[3] E. de Alteriis, F. Cartenì, P. Parascandola, J. Serpa, S. Mazzoleni, Revisiting the Crabtree/Warburg effect in a dynamic perspective: a fitness advantage against sugar-induced cell death, Cell Cycle 17 (6) (2018) 688–701.

[4] R. Thomas, Boolean formalization of genetic control circuits, J. Theor. Biol. 42 (3) (1973) 563–585.

[5] S. A. Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets, J. Theor. Biol. 22 (3) (1969) 437–67.

[6] L. Glass, S. Kauffman, The logical analysis of continuous non linear biochemical control networks, J. Theor. Biol. 39 (1) (1973) 103–129.

[7] 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, J. Theor. Biol. 229 (3) (2004) 339 – 347.

[8] N. Chabrier, F. Fages, Symbolic model checking of biochemical networks, in: C. Priami (Ed.), CMSB'2003, LNCS 2602, Springer-Verlag, 2003, pp. 149–162.

[9] A. Naldi, D. Thieffry, C. Chaouiya, Decision diagrams for the representation and analysis of logical models of genetic networks, in: M. Calder, S. Gilmore (Eds.), CMSB'07, Springer Berlin Heidelberg, 2007, pp. 233–247.

[10] G. Bernot, J.-P. Comet, Z. Khalis, A. Richard, O. F. Roux, A genetically modified Hoare logic, Theoretical Computer Science 765 (2019) 145–157.

[11] L. Gibart, H. Collavizza, J.-P. Comet, Greening R. Thomas' framework with environment variables: a divide and conquer approach, in: Proceedings of the 19th International Conference on Computational Methods in Systems Biology (CMSB), Vol. 12881 of LNBI, 2021, pp. 36–56.

[12] G. Batt, M. Page, I. Cantone, G. Goessler, P. Monteiro, H. de Jong, Efficient parameter search for qualitative models of regulatory networks using symbolic model checking, Bioinformatics 26 (18) (2010) i603–i610.

[13] J.-P. Comet, H. Klaudel, S. Liauzu, Modeling multi-valued genetic regulatory networks using high-level Petri nets, in: In ICATPN 2005, Vol. 3536 of LNCS, 2005, pp. 208–227.

[14] N. Beneš, L. Brim, J. Kadlecaj, S. Pastva, D. Šafránek, Aeon: Attractor bifurcation analysis of parametrised boolean networks, in: S. K. Lahiri, C. Wang (Eds.), Computer Aided Verification, Springer International Publishing, Cham, 2020, pp. 569–581.

[15] J. Barnat, L. Brim, A. Krejci, A. Streck, D. Šafránek, M. Vejnar, T. Vejpustek, On parameter synthesis by parallel model checking, IEEE/ACM Transactions on Computational Biology and Bioinformatics 9 (3) (2012) 693–705.

[16] L. Brim, M. Češka, M. Demko, S. Pastva, D. Šafránek, Parameter synthesis by parallel coloured CTL model checking, in: O. Roux, J. Bourdon (Eds.), Computational Methods in Systems Biology, Springer International Publishing, Cham, 2015, pp. 251–263.

[17] N. Beneš, L. Brim, M. Demko, S. Pastva, D. Šafránek, Parallel SMT-based parameter synthesis with application to piecewise multi-affine systems, in: C. Artho, A. Legay, D. Peled (Eds.), Automated Technology for Verification and Analysis, Springer International Publishing, Cham, 2016, pp. 192–208.

[18] A. Classen, P. Heymans, P.-Y. Schobbens, A. Legay, Symbolic model checking of software product lines, in: 2011 33rd International Conference on Software Engineering (ICSE), 2011, pp. 321–330.

[19] M. Cordy, X. Devroey, A. Legay, G. Perrouin, A. Classen, P. Heymans, P.-Y. Schobbens, J.-F. Raskin, From Software Engineering to Formal

Methods and Tools, and Back: Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday, Springer International Publishing, Cham, 2019, Ch. A Decade of Featured Transition Systems, pp. 285–312.

[20] A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, J.-F. Raskin, Featured transition systems: Foundations for verifying variability-intensive systems and their application to LTL model checking, IEEE Transactions on Software Engineering 39 (8) (2013) 1069–1089.

[21] A. S. Dimovski, A. S. Al-Sibahi, C. Brabrand, A. Wasowski, Family-based model checking without a family-based model checker, in: B. Fischer, J. Geldenhuys (Eds.), Model Checking Software, Springer International Publishing, Cham, 2015, pp. 282–299.

[22] M. H. ter Beek, E. P. de Vink, T. A. C. Willemse, Family-based model checking with mCRL2, in: M. Huisman, J. Rubin (Eds.), Fundamental Approaches to Software Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, 2017, pp. 387–405.

[23] G. Arellano, J. Argil, E. Azpeitia, M. Benítez, M. Carrillo, P. Góngora, D. A. Rosenblueth, E. R. Alvarez-Buylla, "Antelope": a hybrid-logic model checker for branching-time boolean grn analysis, BMC Bioinformatics (2011).

[24] N. Beneš, L. Brim, O. Huvar, S. Pastva, D. Šafránek, Boolean network sketches: a unifying framework for logical model inference, Bioinformatics 39 (4) (04 2023).

[25] L. Gibart, G. Bernot, H. Collavizza, J.-P. Comet, TotemBioNet enrichment methodology: Application to the qualitative regulatory network of the cell metabolism, in: Proceedings of the 14th International Joint Conference on Biomedical Engineering Systems and Technologies, Volume 3: BIOINFORMATICS, Vol. 3, 2021, pp. 85–92.

[26] L. Gibart, Cycle de vie des modèles discrets et application à la régulation du métabolisme des cellules cancéreuses du pancréas, Ph.D. thesis, Université Côte d'Azur, France (2022).
URL `http://www.theses.fr/2022COAZ4074`

[27] 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.

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

[29] S. Malhotra, D. Hayes, D. J. Wozniak, Cystic Fibrosis and Pseudomonas aeruginosa: the Host-Microbe Interface, Clinical Microbiology Reviews 32 (3) (Jun. 2019).

[30] L. Gibart, R. Khoodeeram, G. Bernot, J.-P. Comet, J.-Y. Trosset, Regulation of eukaryote metabolism: An abstract model explaining the Warburg/Crabtree effect, Processes 9 (2021) 1496.

[31] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, A. Tacchella, NuSMV 2: An opensource tool for symbolic model checking, in: Proceedings of the 14th International Conference on Computer Aided Verification, CAV '02, Springer-Verlag, London, UK, UK, 2002, pp. 359–364.

[32] A. Naldi, C. Hernandez, N. Levy, G. Stoll, P. Monteiro, C. Chaouiya, T. Helikar, A. Zinovyev, L. Calzone, S. Cohen-Boulakia, D. Thieffry, L. Paulevé, The CoLoMoTo interactive notebook: Accessible and reproducible computational analyses for qualitative biological networks, Front. Physiol. 9 (2018) 680. doi:10.3389/fphys.2018.00680.

[33] C. Müssel, M. Hopfensitz, H. A. Kestler, BoolNet–an R package for generation, reconstruction and analysis of Boolean networks, Bioinformatics 26 (10) (2010) 1378–1380.

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

[35] M. Huth, M. Ryan, Logic in Computer Science: Modelling and reasoning about systems, Cambridge University Press, 2000.

[36] B. Dutertre, Yices 2.2, in: A. Biere, R. Bloem (Eds.), Computer-Aided Verification (CAV'2014), Vol. 8559 of Lecture Notes in Computer Science, Springer, 2014, pp. 737–744.

[37] J. Demongeot, T. Melliti, M. Noual, D. Regnault, S. Sené, Automata and Complexity: Essays Presented to Eric Goles on the Occasion of His 70th Birthday, Springer International Publishing, 2022, Ch. On Boolean Automata Isolated Cycles and Tangential Double-Cycles Dynamics, pp. 145–178.

[38] M. Elowitz, S. Leibler, A synthetic oscillatory network of transcriptional regulators, Nature (2000).

[39] C. Sequeiros, C. Vázquez, J. R. Banga, I. Otero-Muras, Automated design of synthetic gene circuits in the presence of molecular noise, ACS Synthetic Biology 12 (10) (2023) 2865–2876. doi:10.1021/acssynbio.3c00033.

[40] H. Sun, M. Folschette, M. Magnin, Condition for periodic attractor in 4-dimensional repressilators, in: J. Pang, J. Niehren (Eds.), Computational Methods in Systems Biology, Springer Nature Switzerland, Cham, 2023, pp. 184–201.

[41] D. Boyenval, Modélisation formelle de comportements cycliques biologiques avec points de contrôle : la régulation du cycle cellulaire, Ph.D. thesis, Université Côte d'Azur, France (2022).
URL `http://www.theses.fr/2022COAZ4067`

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: