From Model-Checking to Temporal Logic Constraint Solving

François Fages and Aurélien Rizk

EPI Contraintes, INRIA Paris-Rocquencourt,
BP105, 78153 Le Chesnay Cedex, France.
`Francois.Fages@inria.fr Aurelien.Rizk@inria.fr`
`http://contraintes.inria.fr`

Abstract. In this paper, we show how model-checking can be generalized to temporal logic constraint solving, by considering temporal logic formulae with free variables over some domain $\mathcal{D}$, and by computing a validity domain for the variables rather than a truth value for the formula. This allows us to define a continuous degree of satisfaction for a temporal logic formula in a given structure, opening up the field of model-checking to optimization. We illustrate this approach with reverse-engineering problems coming from systems biology, and provide some performance figures on parameter optimization problems with respect to temporal logic specifications.

1 Introduction

Temporal logics were introduced for program verification by Pnueli [30] as specification languages for expressing the behavior of sequential as well as concurrent programs. Temporal logics essentially extend classical logic, used for describing states, with temporal operators (**X** "at next state", **F** "finally at some future state", **G** "globally at all future states", **U** "until") and computation path quantifiers (**E** "on some path", **A** "on all paths"). Temporal logics have proven useful for formalizing and verifying the behavior of a broad variety of systems ranging from electronic circuits to software programs, physical systems and more recently biological systems, in either boolean [17, 10], discrete [6], stochastic [8, 24] or continuous [33, 19, 9, 2, 10, 4] settings.

There has been a large body of work for generalizing model-checking techniques, initially developed for discrete systems [13, 14], to quantitative transition systems [1, 7, 35, 25, 22, 28, 15, 16]. One common approach is to represent infinite sets of states with finite structures, e.g. by symbolic or numerical constraints, and develop decision or semi-decision procedures for checking the validity of closed temporal logic formulae over such structures.

In computational systems biology however, one core problem is of a reverse-engineering nature: it consists in inferring a mechanistic model of a cell process from knowledge of the elementary interactions and from the observation of the system's global behavior under various conditions (milieu, stress, mutations etc.)

[9]. Beyond verifying whether an already built model does reproduce some dynamical properties specified in temporal logic, model-checking techniques need thus be generalized to model-synthesis techniques, including parameter optimization with respect to temporal specifications. Indeed, most kinetic parameters cannot be measured and must be inferred from the global behavior of the biological system. However, the boolean valuation of temporal logic formulae is not very helpful to guide the search for kinetic parameter values since it does not quantify how far a system is from satisfying its specification.

To answer this question, we generalize the model-checking problem, i.e. deciding whether a closed temporal logic formula is true in a given structure, to a constraint solving problem, i.e. determining the validity domain of the free variables of a given temporal logic formula that make it true in a given structure (see example 2). A continuous degree of satisfaction for a closed temporal logic formula ϕ can then be defined for a given structure, as the distance between the validity domain of a pattern formula ψ obtained by replacing the constants in ϕ by variables, and the actual constant values in ϕ.

In this paper, we present a temporal logic constraint solving algorithm, in a very general first-order setting of Quantifier-Free Computation Tree Logic (QFCTL) formulae with constraints over some arbitrary computational domain $\mathcal{D}$. Then we describe our particular implementation in the Biochemical Abstract Machine BIOCHAM [1] [21] using linear constraint solving over the reals and numerical integration of (non-linear) parametric ordinary differential equations (ODE), and provide some performance figures on a benchmark of kinetic parameter optimization problems with respect to temporal specifications, coming from molecular systems biology.

Such ODE models were considered by Janssen, Van Hentenryck and Deville in [26] for enclosing their solutions and finding their stable states by constraint consistency methods. Our approach consists in using temporal logic to formalize the dynamical properties of the solutions, in a much more flexible way than by specifying their stable states, or than by curve fitting, allowing us to express concentration or time thresholds or oscillation constraints for instance. In our previous work in [9], we introduced the idea of searching parameter values by model-checking with a generate and test algorithm that was limited in practice to 2-3 parameters. In [19] we introduced a constraint solving algorithm for Linear Time Logic (LTL) queries with interval constraints over the reals, interpreted in a single finite trace. In [33] we used it for parameter search in higher dimensions over a single trace and in [34] for robustness analysis using a solver for linear constraints over the reals. Here we generalize this approach with a new constraint solving algorithm for branching time logic (QFCTL), presented in an abstract setting of constraints over some arbitrary domain D. The generalization to branching time logic is not trivial since the labeling procedure must be generalized to a fixpoint algorithm. We believe that this generality is important

[1] BIOCHAM, and the examples of this paper, are available at `http://contraintes.inria.fr/BIOCHAM`.

for relating model-checking to constraint solving independently of a particular application.

The idea of allowing free variables in temporal logic formulae to extract information from a model is however not new. It was introduced by William Chan in [11] in the propositional case with the notion of temporal logic queries, where one free variable stands as a place holder for a propositional formula that makes the query true. Following Chan's seminal paper, temporal queries have been investigated by many authors, but to our knowledge, always in a propositional setting and not in a first-order setting with constraints over some computation domain allowing to compute validity domains for variables. In [31, 15], model-checking procedures for various infinite-state structures have been presented as constraint-solving procedures, however the question of generalizing model-checking to constraint solving for temporal logic formulae containing free variables was not mentioned. Furthermore, the procedures inspired from logic programming were semi-decision procedures, whereas we shall present here decision procedures.

The rest of the paper is organized as follows. Section 2 presents the quantifier free fragment of first-order computation tree logic with constraints over some domain $\mathcal{D}$, noted QFCTL($\mathcal{D}$), and transpose the propositional CTL model-checking algorithm to this setting. Section 3 describes a QFCTL constraint solving algorithm which computes validity domains for variables by fixpoint iteration in quadratic time in the size of the structure. Section 4 studies the case where the underlying constraint domain $\mathcal{D}$ is a metric space, and defines in this case a real-valued degree of satisfaction of a QFCTL formula in a given structure. This continuous valuation in $[0, 1]$ of temporal logic formulae is defined using the validity domain of a QFCTL formula with free variables. This opens up the field of model-checking to continuous optimization with respect to temporal logic specifications over the reals. Section 5 describes our implementation in the Biochemical Abstract Machine BIOCHAM [21] of a QFCTL constraint solver over the reals, restricted to non branching traces, and evaluate its performance on a benchmark of systems biology parameter optimization problems.

2 Quantifier-Free Computation Tree Logic QFCTL

2.1 Syntax

In this paper, we consider a general setting of first-order temporal logic formulae without quantifiers, interpreted in some fixed structure. Let Σ be a signature of constant, function and predicate symbols interpreted over some fixed computation domain $\mathcal{D}$. For the sake of simplicity, we assume that the predicate symbols are closed under negation, i.e. each predicate p comes with its dual $\overline{p}$ such that for all $e_1, \ldots, e_n \in \mathcal{D}$, $\models_{\mathcal{D}} \neg p(e_1, \ldots, e_n)$ if and only if $\models_{\mathcal{D}} \overline{p}(e_1, \ldots, e_n)$.

Let $\mathcal{V}$ be an infinite set of variables, among which a finite set $V \subseteq \mathcal{V}$ of *state variables* (also called rigid variables) is distinguished. An *atomic constraint* is an atomic formula formed over Σ and $\mathcal{V}$ and a *constraint*, noted $c, \ldots$, is a conjunction of atomic constraints.

Quantifier-free first-order computation tree logic (QFCTL*) formulae are formed using the atomic constraints, the logical connectives $\neg$, $\vee$, $\wedge$, $\Rightarrow$, the path quantifiers: **E** (exists a path), **A** (forall paths) and the temporal operators: **X** (next), **F** (finally, at some time point), **G** (globally, at all time points), **U** (until), **W** (weak until). QFCTL denotes the fragment of QFCTL* in which the temporal operators are immediately preceded by a path quantifier.

The set of variables occurring in a formula ϕ is denoted by $V(\phi)$. We say that a formula ϕ is *closed* if $V(\phi) \subseteq V$, i.e. if it contains only state variables.

Example 1. Let us consider inequality constraints over the reals and the QFCTL* formula $\mathbf{EF}\ (x = v_1 \wedge \mathbf{X}\ (x = v_2 \wedge v_1 < v_2 \wedge \mathbf{X}\ (x = v_3 \wedge v_3 < v_2)))$, where x is a state variable and v_1, v_2, v_3 are free variables. This formula expresses that on some path (**E**), at some time point (**F**), the value of x is v_1, and at next time point (**X**), x gets a greater value v_2, and at next time point (**X**), x gets a lesser value, i.e. v_2 is local maximum for x.

2.2 Semantics

QFCTL* formulae are interpreted in branching structures, called Kripke structures, over some computation domain $\mathcal{D}$. We assume that the constraint satisfiability problem in $\mathcal{D}$, i.e. whether $\models_{\mathcal{D}} \exists X\ c$ for a constraint c where $X = V(c)$, is decidable. In the following, the notation $\exists(c)$ stands for the existencial closure of constraint c, i.e; $\exists X\ c$ where $X = V(c) \setminus V$.

A *state*, $s : V \to \mathcal{D}$, is a snapshot of values of state variables at a given time. A state is thus represented by a $\mathcal{D}$-valuation of the variables in V. We will write $s(v)$ for the value of state variable v in state s, and by extension, $s(e)$ for the value of a closed expression e (made up of state variables, function and constraint symbols) in state s. The set of states over V and $\mathcal{D}$ is denoted by $\mathcal{S}(V, \mathcal{D})$, or simply $\mathcal{S}$ when V and $\mathcal{D}$ are implicit.

A *Kripke structure over a set of variables V and a domain $\mathcal{D}$* is a quadruple $K = (S, R, V, \mathcal{D})$ where

– $S \subset \mathcal{S}(V, \mathcal{D})$ is a set of states over V and $\mathcal{D}$,
– $R \subset S \times S$ is a left-total relation over S (i.e. $\forall s \in S\ \exists t \in S\ (s, t) \in R$) called the state transition relation.

We say that a Kripke structure $K = (S, R, V, \mathcal{D})$ is *finite* if S is finite, and *finite state* if $\mathcal{S}(V, \mathcal{D})$ is finite (which implies that K and $\mathcal{D}$ are finite),

A path in K is an infinite sequence of states, noted $\pi = (s_0, s_1, ...)$, such that $(s_i, s_{i+1}) \in R$ for all $i \geq 0$. For such a path π and an integer k, π^k denotes the k^{th} suffix path $(s_k, s_{k+1}, ...)$.

Definition 1 *A closed QFCTL*($\Sigma, V, \mathcal{D}$) formula ϕ is* true in a state s in a Kripke structure $K(S, R, V, \mathcal{D})$*, if the relation $K, s \models_{\mathcal{D}} \phi$ holds following the inductive definition given in Table 1.*

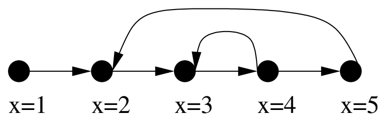
$K, s \models_{\mathcal{D}} c$	if c is an atomic constraint and $\models_{\mathcal{D}} s(c)$,
$K, s \models_{\mathcal{D}} \mathbf{E}\ \phi$	if for some path π starting from s, $K, \pi \models_{\mathcal{D}} \phi$,
$K, s \models_{\mathcal{D}} \mathbf{A}\ \phi$	if for all paths π starting from s, $K, \pi \models_{\mathcal{D}} \phi$,
$K, \pi \models_{\mathcal{D}} \phi$	if $K, s \models_{\mathcal{D}} \phi$ where s is the first state of π,
$K, \pi \models_{\mathcal{D}} \mathbf{X}\ \phi$	if $K, \pi^1 \models_{\mathcal{D}} \phi$,
$K, \pi \models_{\mathcal{D}} \mathbf{F}\ \phi$	if there exists $k \geq 0$ s.t. $K, \pi^k \models_{\mathcal{D}} \phi$,
$K, \pi \models_{\mathcal{D}} \mathbf{G}\ \phi$	if for all $k \geq 0$, $K, \pi^k \models_{\mathcal{D}} \phi$,
$K, \pi \models_{\mathcal{D}} \phi \mathbf{U}\ \phi'$	if there exists $k \geq 0$ s.t. $K, \pi^k \models_{\mathcal{D}} \phi'$ and $K, \pi^j \models_{\mathcal{D}} \phi$ for all $0 \leq j < k$.
$K, \pi \models_{\mathcal{D}} \phi \mathbf{W}\ \phi'$	if either for all $k \geq 0$, $K, \pi^k \models_{\mathcal{D}} \phi$ or there exists $k \geq 0$ s.t. $K, \pi^k \models_{\mathcal{D}} \phi \wedge \phi'$ and for all $0 \leq j < k$, $K, \pi^j \models_{\mathcal{D}} \phi$.
$K, \pi \models_{\mathcal{D}} \neg\phi$	if $K, \pi \not\models_{\mathcal{D}} \phi$,
$K, \pi \models_{\mathcal{D}} \phi\ \wedge\ \phi'$	if $K, \pi \models_{\mathcal{D}} \phi$ and $K, \pi \models_{\mathcal{D}} \phi'$,
$K, \pi \models_{\mathcal{D}} \phi\ \vee\ \phi'$	if $K, \pi \models_{\mathcal{D}} \phi$ or $K, \pi \models_{\mathcal{D}} \phi'$,
$K, \pi \models_{\mathcal{D}} \phi \Rightarrow \phi'$	if $K, \pi \models_{\mathcal{D}} \phi'$ or $K, \pi \not\models_{\mathcal{D}} \phi$,

Table 1. Inductive definition of the truth values of closed QFCTL* formulae in a Kripke structure.

It is worth noting that the only difference between the inductive definition of Table 1 and the usual definition for propositional CTL* formulae [14] is in the base case of an atomic constraint c. Such an atomic constraint is closed and is evaluated in $\mathcal{D}$ by checking its validity: $\models_{\mathcal{D}} s(c)$.

Definition 2 *A QFCTL*$(\Sigma, \mathcal{V}, \mathcal{D})$ formula ϕ is* satisfiable in a Kripke structure *$K = (S, R, V, \mathcal{D})$, noted (by a slight abuse of notation) $K \models_{\mathcal{D}} \exists Y\ \phi$, where $Y = V(\phi) \setminus V$, if there exists a state $s \in S$ and a valuation $\rho : Y \to \mathcal{D}$ such that $K, s \models_{\mathcal{D}} \rho(\phi)$.*

Example 2. Let us consider the following finite Kripke structure over the reals, composed of five states, where state variable x takes values 1 to 5 respectively:



The QFCTL formula $\mathbf{EG}\ (x \leq V)$ has one free variable V. This formula is satisfiable in all states. It is true for all valuations of $V \geq 5$ in the last state, and for all valuations of $V \geq 4$ in the other states. The QFCTL constraint solving problem consists in computing these validity domains for the free variables of the formula in each state.

QFCTL* formulae enjoy the classical duality properties: $\neg\mathbf{E}\ \phi = \mathbf{A}\ \neg\phi$, $\neg\mathbf{X}\ \phi = \mathbf{X}\ \neg\phi$, $\neg\mathbf{F}\ \phi = \mathbf{G}\ \neg\phi$, $\neg(\phi_1 \mathbf{U}\ \phi_2) = (\neg\phi_2 \mathbf{W}\ \neg\phi_1)$ Moreover, $\mathbf{F}$ and $\mathbf{G}$ can be defined as abbreviations: $\mathbf{F}\ \phi = (true \mathbf{U}\ \phi)$, $\mathbf{G}\ \phi = (\phi \mathbf{W}\ false)$. Similarly, $\mathbf{W}$ can be defined from $\mathbf{G}$ and $\mathbf{U}$ by $\phi_1 \mathbf{W}\ \phi_2 = \mathbf{G}\ \phi_1 \vee (\phi_1 \mathbf{U}\ \phi_1 \wedge \phi_2)$. By assuming that the constraint language is closed under negation, negations (and implications) can be eliminated from QFCTL formulae by pushing them down to the constraints. Without loss of generality, we will thus sometimes restrict the QFCTL* formulae to *negation-free normal forms*, formed using $\vee$, $\wedge$, **E** , **A** , **X** , **U** , and **G** only.

2.3 QFCTL Model-Checking Algorithm

The (global) model-checking problem is the problem of determining the set of states in which a given temporal logic formula is true in a given finite Kripke structure. In particular, it solves the (local) decision problem of determining whether a given formula is true in a given initial state. The *QFCTL($\Sigma, V, \mathcal{D}$) model-checking problem* is the following:
Input: a finite Kripke structure $K = (S, R, V, \mathcal{D})$, a closed QFCTL($\Sigma$,V,$\mathcal{D}$) formula ϕ,
Output: the set of states $s \in S$ such that $K, s \models_{\mathcal{D}} \phi$.
By assuming that closed constraints can be evaluated in $\mathcal{D}$, the usual propositional CTL model-checking algorithm for finite Kripke structures [14] can be generalized to QFCTL as follows:

Algorithm 1
```
1. Construct the graph $(S, R)$ of $K$
2. for each subformula $\psi$ of $\phi$, taken in increasing order, add $\psi$ to the labels of
   (a) the states $s$ s.t. $\models_{\mathcal{D}} s(\psi)$ if $\psi$ is an atomic constraint,
   (b) the states not labelled by $\psi_1$ if $\psi = \neg\psi_1$,
   (c) the states labelled by $\psi_1$ and $\psi_2$ if $\psi = \psi_1 \wedge \psi_2$ (similarly for $\vee, \Rightarrow$),
   (d) the immediate predecessors of states labeled by $\psi_1$ if $\psi = \mathbf{EX}\ \psi_1$
   (e) i. the states labelled by $\psi_2$
       ii. and their predecessors labelled by $\psi_1$
       if $\psi = \mathbf{E}\ (\psi_1 \mathbf{U}\ \psi_2)$
   (f) i. the states of non trivial strongly connected components labelled by $\psi_1$,
       ii. and their predecessors labelled by $\psi_1$
       if $\psi = \mathbf{EG}\ \psi_1$,
3. return the states labeled by $\phi$.
```

Example 3. On the Kripke structure of example 2, the model-checking algorithm evaluates the formula **EG** $(x \leq 4)$ by labeling the states with the subformulas as follows:

state	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$
step *(a)*	$x \leq 4$	$x \leq 4$	$x \leq 4$	$x \leq 4$	
step *(f)i.*			**EG** $(x \leq 4)$	**EG** $(x \leq 4)$	
step *(f)ii.*	**EG** $(x \leq 4)$	**EG** $(x \leq 4)$	**EG** $(x \leq 4)$	**EG** $(x \leq 4)$	

By using Tarjan's linear time algorithm for computing strongly connected components as usual [14], we get

Proposition 3 *Algorithm 1 solves the model-checking problem for QFCTL formulae over a finite Kripke structure K and a computation domain $\mathcal{D}$ in time $O(|K| * |\phi| * f(|\phi|))$, where ϕ is the formula to verify, $|\phi|$ its size (number of subformulae) and $f(n)$ is the time complexity for checking the $\mathcal{D}$-validity of a closed constraint of size n.*

3 QFCTL Constraint Solving Algorithm

Definition 4 *The QFCTL($\Sigma, \mathcal{V}, \mathcal{D}$)* satisfiability problem *is the following:*
Input: *a finite Kripke structure $K = (S, R, V, \mathcal{D})$, a QFCTL($\Sigma$,$\mathcal{D}$) formula ϕ where $Y = V(\phi) \setminus V$,*
Output: *set of states s such that $K, s \models_{\mathcal{D}} \exists Y\ \phi$.*

The constraint solving problem *is distinguished from the satisfiability problem by asking moreover to compute the validity domains of the variables:*
Output: *set of pairs (s, ρ) where s is a state and $\rho : Y \rightarrow \mathcal{D}$ is a valuation s.t. $K, s \models_{\mathcal{D}} \rho(\phi)$, assuming a finite representation of an infinite set of valuations, e.g. by a finite set of constraints.*

3.1 Fixpoint Computation of Validity Domains

It is well known that a propositional CTL formula ϕ can be identified with the set of states which satisfy it, i.e. $\{s \in \mathcal{S} \mid K, s \models_{\mathcal{D}} \phi\}$, and that for finite Kripke structures, the basic CTL operators can be characterized as the least or greatest fixpoints of certain monotonic operators in $2^{\mathcal{S}} \rightarrow 2^{\mathcal{S}}$, called predicate transformers [14, 18].

Interestingly, this fixpoint characterization extends to the solutions of QFCTL formulae containing free variables, by associating, to a QFCTL formula ϕ, a set of states given with constraints for the free variables of ϕ describing the solutions of the satisfiability problem for ϕ.

$[c] = \{(s \vert c) \ :\ s \in S$ and $\models_{\mathcal{D}} \exists(s(c))\}$ for a constraint c,	
$[\mathbf{EX}\ \phi] = ex([\phi])$	$[\mathbf{AX}\ \phi] = ax([\phi])$
$[\mathbf{EF}\ \phi] = \mu Z.[\phi] \cup ex(Z)$	$[\mathbf{AF}\ \phi] = \mu Z.[\phi] \cup ax(Z)$
$[\mathbf{EG}\ \phi] = \nu Z.[\phi] \sqcap ex(Z)$	$[\mathbf{AG}\ \phi] = \nu Z.[\phi] \sqcap ax(Z)$
$[\mathbf{E}\ (\phi_1 \mathbf{U}\ \phi_2)] = \mu Z.[\phi_2] \cup ([\phi_1] \sqcap ex(Z))$	$[\mathbf{A}\ (\phi_1 \mathbf{U}\ \phi_2)] = \mu Z.[\phi_2] \cup ([\phi_1] \sqcap ax(Z))$
$[\mathbf{E}\ (\phi_1 \mathbf{W}\ \phi_2)] = \nu Z.[\phi_1] \cup ([\phi_2] \sqcap ex(Z))$	$[\mathbf{A}\ (\phi_1 \mathbf{W}\ \phi_2)] = \nu Z.[\phi_1] \cup ([\phi_2] \sqcap ax(Z))$

Table 2. Fixpoint characterization of the constrained states satisfying a QFCTL formula.

Let $\mathcal{S}_{\mathcal{C}}$ be the set of state-constraint pairs, noted $s|c$, where s is a state and c is a $\mathcal{D}$-satisfiable constraint. Let us consider the set lattice $(2^{\mathcal{S}_{\mathcal{C}}}, \emptyset, \mathcal{S}_{\mathcal{C}}, \cup, \sqcap)$ with the operations of set union $\cup$ and constrained intersection $\sqcap$, i.e. intersection of states with a satisfiable constraint conjunction:

$$A \sqcap B = \{(s|c \wedge c') \ :\ s|c \in A,\ s|c' \in B,\ \models_{\mathcal{D}} \exists(c \wedge c')\}.$$

We shall not describe subsumption checks in this presentation, however it is worth noticing that the lattice top element $\mathcal{S}_{\mathcal{C}}$ of all constrained states is logically equivalent to the element of all states given with the constraint true $\{(s|true) \ :\ s \in \mathcal{S}\}$. This element will be used as top element in computations.

Let $ex, ax : 2^{\mathcal{S}_{\mathcal{C}}} \rightarrow 2^{\mathcal{S}_{\mathcal{C}}}$ be the two constrained predicate transformers (associated to CTL operators **EX** and **AX**) defined by:

$$ex(Z) = \{(s|c) \in \mathcal{S}_\mathcal{C} \ : \ \exists (s,t) \in R \ s.t. \ t|c \in Z\},$$

$$\begin{aligned} ax(Z) = \{(s|c_1 \wedge \ldots \wedge c_n) \in \mathcal{S}_\mathcal{C}) \ &: \{t \ : \ (s,t) \in R\} = \{t_1, \ldots, t_n\}, \\ &\forall i, \ 1 \leq i \leq n, \ \ t_i|c_i \in Z, \ \models_\mathcal{D} \exists (c_1 \wedge \ldots \wedge c_n)\}. \end{aligned}$$

Let us consider the fixpoint equations between QFCTL formulae and sets of state-constraint pairs given in Table 2. These equations can be used to compute the validity domains of the free variables of a QFCTL formula in each state, by finite fixpoint iteration.

Example 4. For the formula **EG** $(x \leq V)$ and the Kripke structure of Example 2, the fixpoint iteration provides the following result:

state	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$
$x \leq V$	$1 \leq V$	$2 \leq V$	$3 \leq V$	$4 \leq V$	$5 \leq V$
$\tau^0_{\mathbf{EG}\ (x \leq V)}$	true	true	true	true	true
$\tau^1_{\mathbf{EG}\ (x \leq V)}$	$1 \leq V$	$2 \leq V$	$3 \leq V$	$4 \leq V$	$5 \leq V$
$\tau^2_{\mathbf{EG}\ (x \leq V)}$	$2 \leq V$	$3 \leq V$	$4 \leq V$	$4 \leq V$	$5 \leq V$
$\tau^3_{\mathbf{EG}\ (x \leq V)}$	$3 \leq V$	$4 \leq V$	$4 \leq V$	$4 \leq V$	$5 \leq V$
$\tau^4_{\mathbf{EG}\ (x \leq V)}$	$4 \leq V$	$4 \leq V$	$4 \leq V$	$4 \leq V$	$5 \leq V$
$\tau^5_{\mathbf{EG}\ (x \leq V)} = [\mathbf{EG}\ (x \leq V)]$	$4 \leq V$	$4 \leq V$	$4 \leq V$	$4 \leq V$	$5 \leq V$

While for the formula **AG** $(x \leq V)$ involving greatest fixpoint computation, starting from all states labelled by the constraint true, we get the fixpoint

state	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$
$[\mathbf{AG}\ (x \leq V)]$	$5 \leq V$	$5 \leq V$	$5 \leq V$	$5 \leq V$	$5 \leq V$

Let us say that an operator τ over a set S is *bounded* if $\forall s \in S \ \exists i \in N \ \tau^{i+1}(s) = \tau^i(s)$. It is clear from the proof of Knaster-Tarski-Kleene theorem that:

Proposition 1. *A bounded monotonic operator τ over a lattice $(L, \bot, \top, \sqcup, \sqcap)$ admits a least fixpoint equal to $\tau^i(\bot)$ for some $i \geq 0$, and a greatest fixpoint equal to $\tau^j(\top)$ for some $j \geq 0$.*

Lemma 1. *The constrained predicate transformers ex and ax, and the constrained predicate transformers associated to QFCTL operators, are monotonic and bounded in finite Kripke structures.*

Proof. As for monotonicity, it is clear that if $Z_1 \subset Z_2$ then $ex(Z_1) \subseteq ex(Z_2)$, $ax(Z_1) \subseteq ax(Z_2)$. The same goes for the predicate transformers of QFCTL operators since they proceed by intersection or union of monotonic operators.

Predicate transformers ex and ax are also bounded since otherwise one could exhibit an infinite chain of states such that $(s_i, s_{i+1}) \in R$ with $\forall i, j, \ 1 \leq i < j, \ s_i \neq s_j$, a contradiction in a finite Kripke structure. The same goes for the other predicate transformers since they are built by union or intersection of bounded operators.

Proposition 2 (soundness). *If $s|c \in [\phi]$ then $K, s \models_\mathcal{D} \rho(\phi)$ for every valuation ρ such that $\models_\mathcal{D} \rho(c)$.*

Proof. By structural induction on ϕ.

Proposition 3 (completeness). *If $K, s \models_{\mathcal{D}} \rho(\phi)$ then there exists $s|c \in [\phi]_K$ such that $\models_{\mathcal{D}} \rho(c)$.*

Proof. By structural induction on ϕ.

We thus get:

Theorem 1. *The satisfiability problem of QFCTL formulae in a finite Kripke structure over a domain $\mathcal{D}$ with a decidable language of constraints, is decidable.*

Proposition 4. *The number of fixpoint iteration steps in the QFCTL constraint solving algorithm 2 is in $O(n * k^2)$ where n is the size of the formula and k is the number of states.*

Proof. The algorithm proceeds by iteratively computing constrained states for the subformulae of the formula, hence in at most n steps. For each subformula, the algorithm computes a fixpoint of constrained states by iteration on constrained states, hence in at most k^2 steps. Each elementary step involves constraint satisfiability checking operations whose time complexity is not counted here.

This quadratic complexity in the number of states must be contrasted with the linear complexity of the QFCTL model-checking algorithm (Prop. 3). The linear complexity of model-checking relies on Tarjan's algorithm for computing strongly connected components of the structure for **EG** formulae. For computing validity domains however, one would need to consider the different circuits of the structure in order to label the states with appropriate domains for **EG** formulae (see example 2). The fixpoint computation shows that this labeling can be done in quadratic time, whereas a naive algorithm considering all the circuits of the finite Kripke structure would require an exponential time.

4 QFCTL Formulae over a Metric Space $\mathcal{D}$

In this section, we consider the case where the computation domain $\mathcal{D}$ is a metric space, i.e. a domain given with a distance function $d : \mathcal{D}^2 \to \mathbb{R}$.

4.1 Continuous Valuation of QFCTL Formulae

In this general setting of metric spaces as computation domain, the QFCTL constraint solving algorithm provides a mean to evaluate closed QFCTL formula continuously in the interval $[0, 1]$, instead of by a Boolean value.

For this, given a QFCTL formula ϕ, a QFCTL *pattern formula* $\psi(x_1, ..., x_k)$ is introduced by replacing some constants in ϕ by new variables $\{x_1, ..., x_k\}$: we have $\phi = \psi(v_1, ..., v_k)$ for some instantiation of the variables by domain values $v_1, ..., v_k$. The satisfaction degree of ϕ is then defined using the distance between the validity domain of the variables $x_1, ..., x_k$ in ψ and the objective values $(v_1, ..., v_k)$ in $\mathbb{R}^k$.

Definition 5 *The* violation degree $vd(\phi, \psi)$ *of a QFCTL formula* ϕ *in a Kripke structure* K *with respect to a pattern formula* $\psi(\boldsymbol{y})$ *such that* $\phi = \psi(\boldsymbol{v})$ *for some real values* $\boldsymbol{v}$*, is the euclidean distance* d *between* $\boldsymbol{v}$ *and the initial state validity domain* $D^{\boldsymbol{y}}_{\psi}$ *of the free variables of* ψ*, or* $+\infty$ *if* ψ *is not satisfiable:*

$$vd(\phi, \psi) = min_{\boldsymbol{v'} \in D^{\boldsymbol{y}}_{\psi}} d(\boldsymbol{v'}, \boldsymbol{v})$$

The satisfaction degree*,* $sd(\phi, \psi) \in [0, 1]$*, of* ϕ *with respect to* ψ *is obtained by normalization:*

$$sd(\phi, \psi) = \frac{1}{1 + vd(\phi, \psi)}$$

Example 5. For instance, in example 3, the formulae **EG** $(x \leq 2)$ and **AG** $(x \leq 2)$ can now be given a continuous degree of satisfaction with respect to the pattern formulae **EG** $(x \leq V)$ and **AG** $(x \leq V)$ respectively. This is obtained simply by computing the distance between the validity domain of V and the objective value 2. This gives the following satisfaction degrees:

state	$x = 1$	$x = 2$	$x = 3$	$x = 4$	$x = 5$
$sd($**EG** $(x \leq 2)$	1/3	1/3	1/3	1/3	1/4
$sd($**AG** $(x \leq 2))$	1/4	1/4	1/4	1/4	1/4

Interestingly, this notion of satisfaction degree gives also rise naturally to a pseudometric between Kripke structures with respect to a temporal specification ϕ, by considering the difference in the degrees of satisfaction of ϕ between both structures.

4.2 Parameter Optimization with respect to a QFCTL Formula

Because it quantifies how far a Kripke structure K is from a given specification, the satisfaction degree can be used to guide the search when one tries to modify a Kripke structure to make it satisfy a specification. When $\mathcal{D} = \mathbb{R}$, this enables the use of (non-linear) continuous optimization methods, where state valuations are the variables being optimized and where the satisfaction degree provides a "black box" fitness function [33].

An optimization problem is thus defined for the satisfaction degree of a QFCTL formula ϕ^* w.r;t. a valuation ρ_ϕ of some of its variables. This has an intuitive interpretation : ϕ^* is the kind of property considered while ρ_ϕ defines the objective valuation of the actual property.

In the implementation described below, we use the covariance matrix adaptive evolution strategy of Hansen and Ostermeier [23], as non-linear optimization method for maximizing the satisfaction degree of QFCTL specifications.

4.3 Robustness Estimation with respect to QFCTL($\mathbb{R}_{\mathbf{lin}}$) Specifications

The notion of satisfaction degree can also be used to define a degree of robustness of a behavior described in temporal logic with respect to a set of perturbations, and estimate it computationally [34]. This robustness degree is defined as the mean value of the satisfaction degree of the property of interest over all admissible perturbations, possibly weighted by probabilities. This definition is an adaptation of the general definition given by Kitano [27] to the temporal logic setting:

Definition 6 *[34] Let P be a set of perturbations, $prob(p)$ be the probability of perturbation p, s be the initial state of the numerical trace of the system under perturbation $p \in P$. The* robustness degree *$R_{\phi,P}$ of a property ϕ with respect to P is the real value $R_{\phi,P} = \int_{p \in P} sd(\phi, s) prob(p) dp$*

In the case of an infinite perturbation set, this robustness degree can be estimated by sampling. The robustness degree can be used to compare models and can even be integrated as a criterion in the parameter optimization procedure [33].

4.4 Implementation

Our current implementation of QFCTL constraint solving is restricted to linear constraints over the reals and to linear Kripke structures, i.e. numerical traces without branching. The constraint solving problem of a QFCTL formula on a numerical trace, the computation of the satisfaction degree of a formula and its use as a fitness function for parameter optimization are implemented in version 2.8 of the freely available tool BIOCHAM, a modeling environment for the analysis of biological systems [21]. The computation of validity domains is handled by a simplified version of the QFCTL constraint solving algorithm dealing only with a single numerical trace, i.e. a finite linear structure [34]. The atomic constraints are linear constraints over the reals. Their satisfiability is checked using the Parma Polyhedra Library [3].

5 Applications in Systems Biology

5.1 Context of Molecular Systems Biology

The use of temporal logics in systems biology relies on a logical paradigm which consists in making the following identifications:

biological model = (quantitative) transition system
biological properties = temporal logic formulae
automatic validation = model-checking

In this domain, temporal logics have been used in recent years in many applications, either as query languages of large interaction maps [10] or gene regulatory networks [4], or as specification languages of biological properties known or inferred [19] from experiments, and used for validating models, discriminating between models and proposing new biological experiments [6], finding parameter values [9], or estimating robustness [33, 5].

The difficulty inherent in using quantitative, but still incomplete, uncertain and imprecise biological knowledge makes the modeling problem a challenging task. Temporal logics help cope with this difficulty by providing a powerful specification language of the behavior of the system. The advantage of temporal logics is particularly explicit in comparison with the essentially qualitative properties considered in dynamical systems theory (e.g. multistability, existence of oscillations) or with the exact quantitative properties considered in optimization theory (e.g. curve fitting) as it allows us to express both qualitative (e.g. some protein is eventually produced) and quantitative (e.g. a concentration exceeds 10) properties.

Numerical traces representing evolution over a given time span of biological species can be obtained either from measurements in biological experiments, or from simulations by numerical integration of (non-linear) ODE models. QFCTL formulae are interpreted on these finite traces by adding a loop on the last state which only changes the meaning of **X** [33]. Non-linear continuous optimization methods can then be used to optimize a biological model with respect to a QFCTL specification. Note that in this case, the Kripke structure is optimized indirectly, by optimizing the biological model producing it.

5.2 Parameter Optimization with respect to QFCTL($\mathbb{R}_{\mathbf{lin}}$) Specifications

We examine here the problem of finding parameter values of biological models such that the numerical trace obtained by simulation satisfies a given specification. We consider several examples of parameter optimization problems [33] in three ordinary differential equation models. We provide the CPU time (in seconds) required for optimization and for a single validity domain computation.

The first model is the budding yeast cell cycle ODE model of Chen et al. [12] which displays how some proteins interact to form an heterodimer known as maturation promoting factor (*MPF*) playing a key role in the control of mitotic cycles. For given sets of kinetic parameter values, *MPF* exhibits periodic activity peaks. We use formulae ϕ_1^*, ϕ_2^* and ϕ_3^* to see if it is possible to respectively find values in order to have higher *MPF* peaks, greater *MPF* amplitude, or shorter oscillation periods.

The second model is a model of the MAPK signal transduction cascade in the cell [29]. In [32], oscillations have been found in this model of the cascade of enzymatic reactions is directional and does not contain any negative feedback reaction. In [20] we analyzed this phenomenon in terms of the negative circuits of the influence graph associated to a reaction graph. Here, we use ϕ_5 to find parameter values and initial conditions that exhibit sustained oscillations with some amplitude constraint on the protein complex *MAPKp1p2*. Formula ϕ_4 illustrates a curve fitting problem on two time points and the protein complex *MEKRAFp1*.

The last example is a design problem. Given a specification of a synthetic gene transcriptional cascade system, whose input is *EYFP* we search for transcription rate parameter values with well-timed and fast-switching constraints (formula ϕ_6).

The following QFCTL formulae are considered with the objective valuations given in column ρ_ϕ of Table 3:

$$\phi_1^* = \; EF([MPF] > max)$$

$$\begin{aligned}\phi_2^* = \; & EF([MPF] > x1 \wedge EF([MPF] < x2 \\ & \wedge EF([MPF] > x1 \wedge EF([MPF] < x2)))) \\ & \wedge x1 - x2 > a\end{aligned}$$

$$\begin{aligned}\phi_3^* = \; & EF(d([Cdc2])/dt < 0 \wedge EX(d([Cdc2])/dt > 0 \wedge Time = t1 \\ & \wedge EX(F(d([Cdc2])/dt > 0 \wedge EX(d([Cdc2])/dt < 0 \wedge Time = t2)) \\ & \wedge t2 - t1 > p\end{aligned}$$

$$\begin{aligned}\phi_4^* = \; & EG(Time = 30 \rightarrow [MEKRAFp1] = u \\ & \wedge Time = 60 \rightarrow [MEKRAFp1] = v) \wedge \cdots\end{aligned}$$

$$\begin{aligned}\phi_5^* = \; & EF([MAPKp1p2] > x1 \wedge EF([MAPKp1p2] < x2)) \\ & \wedge x1 - x2 > a\end{aligned}$$

$$\begin{aligned}\phi_6^* = \; & EG(Time < t_1 \rightarrow [EYFP] < 10^3) \\ & \wedge EG(Time > t_2 \rightarrow [EYFP] > 10^5) \\ & \wedge t_1 > b1 \wedge t_2 < b_2 \wedge t_2 - t_1 < b_3\end{aligned}$$

Model	#parameters	$\|V\|$	last $\|S\|$	ϕ^*	ρ_ϕ	$\|Y\|$	CPU time (s)	#iterations	CPU time (s) for val. domain in one iteration
Cell cycle	2	6	186	ϕ_1^*	max=0.3	1	27	132	0.02
Cell cycle	2	6	204	ϕ_2^*	a=0.3	3	131	138	1.2
Cell cycle	8	6	267	ϕ_3^*	p=20	3	23	40	0.39
MAPK	22	22	35	ϕ_4^*	$u, v.. = 0.03, 0.04..$	6	259	611	0.001
MAPK	37	22	234	ϕ_5^*	a=0.5	3	453	868	0.17
Cascade	15	3	346	ϕ_6^*	b_1, b_2, b_3	5	70	96	0.48
Cascade					$= 150, 150, 450$				

Table 3. Parameter optimization benchmark where #parameters is the number of parameters of the biological model being optimized, $|V|$ the number of species in that model (i.e the number of variables in the produced numerical trace), last $|S|$ the number of states in the last produced numerical trace, $|Y|$ the number of formula variables, CPU time the time in seconds required to complete the optimization on a Core 2 Duo 2GHz, #iterations the number of calls to the fitness function (i.e validity domain computations and numerical simulations of the model) and CPU time for val. domain in one iteration the time to compute the validity domain on the last simulated trace.

Table 3 summarizes our performance evaluation on these benchmarks. The dimension of the search space (i.e. number of parameters) does not determine alone the complexity of an optimization problem : problems involving ϕ_4^* and ϕ_5^* have high dimensions and are the longest to solve but ϕ_3^* and ϕ_6^* are faster than ϕ_2^* despite their much higher dimensions. But even with search space dimensions as high as 37, by guiding the search with the continuous valuation of QFCTL formulae, it is possible to find solutions with only less than a thousand calls to the fitness function. Notice that the time required to compute the validity domains is in general only a small fraction of the total CPU time. This can be explained by the optimization method overhead and more importantly by the time required to generate traces by numerical integration.

6 Conclusion

We have shown that the QFCTL constraint satisfiability problem is decidable in finite Kripke structures over an arbitrary computation domain with a decidable language of constraints, i.e. that any constraint solver can be lifted to a temporal logic constraint solver over finite Kripke structures. We have presented a generic QFCTL constraint solver which computes validity domains for the free variables of a formula, in quadratic time in the number of states, and linear time in the size of the formula, apart from the basic constraint satisfiability checks.

We have shown that the computation of validity domains for QFCTL constraints over metric spaces makes it possible to define a continuous measure of satisfaction of QFCTL formulae, opening up the field of model-checking to optimization. This has been illustrated with central computational issues in systems biology, from which our present work originates, for inferring kinetic parameter values in structural models from the observed behaviors of the system formalized in temporal logic with numerical constraints. It should be clear however that these methods for parameter optimization with respect to temporal logic specifications and robustness analyses, should be of a wider application spectrum in dynamical systems, reverse-engineering and synthesis of hybrid systems.

Acknowledgements. We gracefully acknowledge discussions with Sylvain Soliman and Grégory Batt on this topic, partial support from the European FP6 Strep project TEMPO, and the reviewer's comments for improving the presentation.

References

1. Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *Logic in Computer Science*, pages 313–321, 1996.
2. Marco Antoniotti, Alberto Policriti, Nadia Ugel, and Bud Mishra. Model building and model checking for biochemical processes. *Cell Biochemistry and Biophysics*, 38:271–286, 2003.
3. R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.
4. G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page, and D. Schneider. Validation of qualitative models of genetic regulatory networks by model checking : Analysis of the nutritional stress response in *Escherichia coli. Bioinformatics*, 21(Suppl.1):i19–i28, 2005.

5. G. Batt, B. Yordanov, R. Weiss, and C. Belta. Robustness analysis and tuning of synthetic gene networks. *Bioinformatics*, 23(18):2415–2422, 2007.
6. Gilles Bernot, Jean-Paul Comet, Adrien Richard, and J. Guespin. A fruitful application of formal methods to biological regulatory networks: Extending thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347, 2004.
7. Tevfik Bultan, Richard Gerber, and William Pugh. Symbolic model checking of infinite state systems using presburger arithmetic. In *CAV'97: Proceedings of the 9th International Conference on Computer Aided Verification*, pages 400–411. Springer-Verlag, 1997.
8. Muffy Calder, Vladislav Vyshemirsky, David Gilbert, and Richard Orton. Analysis of signalling pathways using the continuous time markow chains. In Gordon Plotkin, editor, *Transactions on Computational Systems Biology VI*, volume 4220 of *Lecture Notes in BioInformatics*, pages 44–67. Springer-Verlag, November 2006. CMSB'05 Special Issue.
9. Laurence Calzone, Nathalie Chabrier-Rivier, François Fages, and Sylvain Soliman. Machine learning biochemical networks from temporal logic properties. In Gordon Plotkin, editor, *Transactions on Computational Systems Biology VI*, volume 4220 of *Lecture Notes in BioInformatics*, pages 68–94. Springer-Verlag, November 2006. CMSB'05 Special Issue.
10. Nathalie Chabrier and François Fages. Symbolic model checking of biochemical networks. In Corrado Priami, editor, *CMSB'03: Proceedings of the first workshop on Computational Methods in Systems Biology*, volume 2602 of *Lecture Notes in Computer Science*, pages 149–162, Rovereto, Italy, March 2003. Springer-Verlag.
11. William Chan. Temporal-logic queries. In *CAV'00: Proceedings of the 12th International Conference on Computer Aided Verification*, number 1855 in Lecture Notes in Computer Science, pages 450–463. Springer-Verlag, 2000.
12. Katherine C. Chen, Attila Csikász-Nagy, Bela Györffy, John Val, Bela Novàk, and John J. Tyson. Kinetic analysis of a molecular model of the budding yeast cell cycle. *Molecular Biology of the Cell*, 11:396–391, 2000.
13. Edmund M. Clarke. and Allen E. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs 1981*, pages 52–71, London, UK, 1982. Springer-Verlag.
14. Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking.* MIT Press, 1999.
15. Giorgio Delzanno and Andreas Podelski. Constraint-based deductive model checking. *STTT*, 3(3):250–270, 2001.
16. Magnus Egerstedt and Bud Mishra, 2008.
17. Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln, José Meseguer, and M. Kemal Sönmez. Pathway logic: Symbolic analysis of biological signaling. In *Proceedings of the seventh Pacific Symposium on Biocomputing*, pages 400–412, January 2002.
18. Allen E. Emerson and Edmund M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, pages 169–181, London, UK, 1980. Springer-Verlag.
19. François Fages and Aurélien Rizk. On temporal logic constraint solving for the analysis of numerical data time series. *Theoretical Computer Science*, 408(1):55–65, November 2008.

20. François Fages and Sylvain Soliman. From reaction models to influence graphs and back: a theorem. In *Proceedings of Formal Methods in Systems Biology FMSB'08*, number 5054 in Lecture Notes in Computer Science. Springer-Verlag, February 2008.
21. François Fages, Sylvain Soliman, and Aurélien Rizk. *BIOCHAM v2.8 user's manual*. INRIA, 2009. `http://contraintes.inria.fr/BIOCHAM`.
22. Nicolas Halbwachs, Yann-Erick Proy, and Patrick Roumanoff. Verification of real-time systems using linear relation analysis. In *Formal Methods in System Design*, pages 157–185, 1997.
23. Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
24. J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. In *Proc. Computational Methods in Systems Biology (CMSB'06)*, volume 4210 of *Lecture Notes in Computer Science*, pages 32–47. Springer-Verlag, 2006.
25. Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. In *CAV'97: Proceedings of the 9th International Conference on Computer Aided Verification*, pages 460–463. Springer-Verlag, 1997.
26. Micha Janssen, Pascal Van Hentenryck, and Yves Deville. A constraint satisfaction approach for enclosing solutions to parametric ordinary differential equations. *SIAM Journal on Numerical Analysis*, 40(5), 2002.
27. H. Kitano. Towards a theory of biological robustness. *Molecular Systems Biology*, 3:137, 2007.
28. Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1:134–152, 1997.
29. Andre Levchenko, Jehoshua Bruck, and Paul W. Sternberg. Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. *PNAS*, 97(11):5818–5823, May 2000.
30. Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.
31. Andreas Podelski. Model checking as constraint solving. In Jens Palsberg, editor, *Proceedings of SAS: Static Analysis Symposium*, volume 1824 of *LNCS*, pages 22–37. Springer-Verlag, 2000.
32. Liang Qiao, Robert B. Nachbar, Ioannis G. Kevrekidis, and Stanislav Y. Shvartsman. Bistability and oscillations in the huang-ferrell model of mapk signaling. *PLoS Computational Biology*, 3(9):1819–1826, September 2007.
33. Aurélien Rizk, Grégory Batt, François Fages, and Sylvain Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In Monika Heiner and Adeline Uhrmacher, editors, *CMSB'08: Proceedings of the fourth international conference on Computational Methods in Systems Biology*, volume 5307 of *Lecture Notes in Computer Science*, pages 251–268. Springer-Verlag, October 2008.
34. Aurélien Rizk, Grégory Batt, François Fages, and Sylvain Soliman. A general computational method for robustness analysis with applications to synthetic gene networks. *BioInformatics*, 25(12):169–178, June 2009.
35. Pierre Wolper and Bernard Boigelot. Verifying systems with infinite but regular state space. In *CAV'98: Proceedings of the 10th International Conference on Computer Aided Verification*, pages 88–97, 1998.