Compositional Reachability Analysis of Genetic Networks[⋆]

Gregor Gössler

Pop Art project, INRIA Rhône-Alpes, France

Abstract. Genetic regulatory networks have been modeled as discrete transition systems by many approaches, benefiting from a large number of formal verification algorithms available for the analysis of discrete transition systems. However, most of these approaches do not scale up well. In this article, we explore the use of compositionality for the analysis of genetic regulatory networks. We present a framework for modeling genetic regulatory networks in a modular yet faithful manner based on the mathematically well-founded formalism of differential inclusions. We then propose a compositional algorithm to efficiently analyze reachability properties of the model. A case study shows the potential of this approach.

1 Introduction

A genetic regulatory network usually encompasses a multitude of complex, interacting feedback loops. Being able to model and analyze its behavior is crucial for understanding the interactions between the proteins, and their functions. Genetic regulatory networks have been modeled as discrete transition systems by many approaches, benefiting from a large number of formal verification algorithms available for the analysis of discrete transition systems. However, most of these approaches face the problem of state space explosion, as even models of modest size (from a biological point of view) usually lead to large transition systems, due to a combinatorial blow-up of the number of states. Even if the modeling formalism allows for a compact representation of the state space, such as Petri nets, subsequent analysis algorithms have to cope with the full state space. In practice, non-compositional approaches for the analysis of genetic regulatory networks do not scale up well.

In order to deal with the problem of state space explosion, different techniques have been developed in the formal verification community, such as partial order reduction, abstraction, and compositional approaches. In this article, we explore the use of compositionality for the analysis of genetic regulatory networks. Compositional analysis means that the behavior of a system consisting of different components is analyzed by separately examining the behavior of the components and how they interact, rather than by monolithically analyzing the behavior of

⋆ published in proc. CMSB'06, LNBI 4210, Springer-Verlag.

the overall system. It therefore can be more efficient than non-compositional analysis, and scale better.

A precondition for compositional algorithms to be applicable, is that the model be structured. Therefore, this paper makes two contributions: first, we present a modeling framework for genetic regulatory networks in which the different components of the system (in our case, proteins or sets of proteins) and the way they constrain each other, are modeled separately and modularly. Second, we propose a compositional algorithm allowing to efficiently analyze reachability properties of the model.

Cellular functions are often distributed over groups of components that interact within large networks. The components are organized in functional modules, forming a hierarchical architecture [25, 27]. Therefore, the approach of compositional analysis agrees with the modular structure of genetic regulatory networks, and may take advantage of it by using compositionality on different levels of modularity, for instance, between individual genes, sub-networks, or individual cells.

However, compositionality is not everything. The model should also faithfully represent the actual behavior of the modeled network. The approach we present is based on the mathematically well-founded formalism of qualitative simulation [14].

Related work. By now there is a large number of approaches to model and analyze genetic networks. An overview is given in the survey of [11]. The modeling approaches adopt different mathematical frameworks, which vary in expressiveness and the availability and efficiency of verification algorithms. Most of the algorithms "flatten" the model and work on the global state space, without computationally taking advantage of the modularity of the problem. The approach of [6] compositionally models gene networks in a stochastic framework.

There has been a wide variety of modeling approaches based on differential equations since the work of [19]. However, simulation and verification of the continuous model can be expensive, and many properties are not even decidable in this framework. Therefore, several ways have been investigated to discretize the continuous model defined by differential equations while preserving properties like soundness [14] and reachability [3]. [18] and [1] use predicate abstraction to automatically compute backward reachable sets of piecewise affine hybrid automata, and find a conservative approximation of reachability for linear hybrid systems, respectively. [26] addresses the *bounded reachability* problem of hybrid automata.

In order to deal with complex networks, it may be a good choice to change precision against efficiency, and directly model genetic networks in a discrete framework, such as systems of logical equations [29, 5], Petri nets [22, 9, 28, 10], or rule-based formalisms like term rewriting systems [15, 16]. Formal verification can then be carried out enumeratively (for instance, [13, 2, 23]) or symbolically, see for example [8].

Organization of the paper. In Section 2, we introduce the modeling framework. We show how a genetic network can be modeled in a modular way in this framework, and compare the model with the qualitative model of [7]. Section 3 presents a reachability algorithm taking advantage of the modularity of the model. Section 4 illustrates our results with a case study, and Section 5 concludes.

2 Component-based Modeling of Genetic Networks

This section briefly introduces the notions of piecewise linear system, and its qualitative simulation as defined in [7]. We then define a modularized approximation of qualitative simulation, and compare both models.

2.1 Piecewise Linear Systems

The production of a protein in a cell is regulated by the current protein concentrations, which can activate or inhibit the production, for instance by binding to the gene and disabling transcription. At the same time, proteins are degraded. This behavior of a genetic network can be modeled by a system of differential equations of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) - \mathbf{g}(\mathbf{x}, \mathbf{u})\mathbf{x} \tag{1}$$

where $\mathbf{x}$ is a vector of protein concentrations representing the current state, $\mathbf{u}$ is a vector of *input* concentrations, and the vector-valued function $\mathbf{f}$ and matrix-valued function $\mathbf{g}$ model the production rates, and degradation rates, respectively.

The approach of [14, 7] considers an abstraction where the state space of each variable x_i is partitioned into a set of intervals $\mathcal{D}_i^r$ and a set of *threshold values* $\mathcal{D}_i^s$. This induces a partition of the continuous state space into a discrete set of *domains*, in each of which Equation (1) is approximated with a system of linear differential equations.

Definition 1 (Domain). *Consider a Cartesian product $\theta = \theta_1 \times ... \times \theta_n$ with $\theta_i = \{\theta_i^1, ..., \theta_i^{p_i}\}$ an ordered set of* thresholds*, such that $0 < \theta_i^1 < ... < \theta_i^{p_i} < max_i$. Let*

$$\mathcal{D}_i^r(\theta) = \{[0, \theta_i^1)\} \cup \{(\theta_i^j, \theta_i^{j+1}) \mid 1 \leqslant j < p_i\} \cup \{(\theta_i^{p_i}, max_i]\}$$

and $\mathcal{D}_i^s(\theta) = \big\{\{\theta_i^j\} \mid 1 \leqslant j \leqslant p_i\big\}$. We omit the argument θ when it is clear from the context. Let $\mathcal{D}_i = \mathcal{D}_i^r \cup \mathcal{D}_i^s$, and $\mathcal{D} = \mathcal{D}_1 \times \mathcal{D}_2 \times ... \times \mathcal{D}_n$ be the set of domains*. The domains in $\mathcal{D}^r = \mathcal{D}_1^r \times \mathcal{D}_2^r \times ... \times \mathcal{D}_n^r$ are called* regulatory *domains, the domains $\mathcal{D}^s = \mathcal{D} \setminus \mathcal{D}^r$ are called* switching domains*.*

The state space $[0, max_1] \times \cdots \times [0, max_n]$ is thus partitioned into the set of domains $\mathcal{D}$.

Definition 2 (Piecewise linear system). *A* piecewise linear system *is a tuple $M = (X, \theta, \boldsymbol{\mu}, \boldsymbol{\nu})$ where*

– $X = \{x_1, ..., x_n\}$ *a set of real-valued* state variables*;*
– $\theta = \theta_1 \times ... \times \theta_n$*, with* $\theta_i = \{\theta_i^1, ..., \theta_i^{p_i}\}$ *such that* $0 < \theta_i^1 < ... < \theta_i^{p_i} < max_i$*, associates with each dimension an ordered set of* thresholds*;*
– $\boldsymbol{\mu} : \mathcal{D}^r(\theta) \to \mathbb{R}^n_{\geqslant 0}$ *associates with each regulatory domain a vector of* production rates*;*
– $\boldsymbol{\nu} : \mathcal{D}^r(\theta) \to diag(\mathbb{R}^n_{>0})$ *associates with each regulatory domain a diagonal matrix of* degradation rates*.*

Within a regulatory domain $D \in \mathcal{D}^r$, the protein concentrations $\mathbf{x}$ evolve according to the ratio of production rate and degradation rate:

$$\dot{\mathbf{x}} = \boldsymbol{\mu}(D) - \boldsymbol{\nu}(D)\mathbf{x} \tag{2}$$

and thus converge monotonically towards the *target equilibrium* $\boldsymbol{\phi}$, solution of $\mathbf{0} = \boldsymbol{\mu}(D) - \boldsymbol{\nu}(D)\mathbf{x}$.

Definition 3 ($\boldsymbol{\phi}$). *For any* $D \in \mathcal{D}^r$*, let* $\boldsymbol{\phi}(D)$ *denote the* target equilibrium *of* D *such that*

$$\phi_i(D) = \mu_i(D)/\nu_i(D)$$

for any variable $x_i \in X$*.*

Hypothesis: Throughout this paper we make the assumption that for any regulatory domain D, $\exists D' \in \mathcal{D}^r \,.\, \boldsymbol{\phi}(D) \in D'$, that is, all target equilibria lie within regulatory domains, as in [14].

If $\boldsymbol{\phi}(D) \in D$ then the systems stays in D, otherwise it eventually leaves D and enters an adjacent switching domain. In switching domains, where $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are not defined, the behavior of M is defined using differential inclusions as proposed by [17, 21].

Notations. Let *reg* be the predicate characterizing the set of regulatory domains. For any $i \in \{1, ..., n\}$, let reg_i and $switch_i$ be predicates on $\mathcal{D}$ characterizing the regulatory intervals and thresholds of $\mathcal{D}_i$, respectively: $reg_i(D) \iff D_i \in \mathcal{D}_i^r$, and $switch_i(D) \iff D_i \in \mathcal{D}_i^s$ for any $D \in \mathcal{D}$. The *order* of a domain D is the number of variables taking a threshold value in D. Let $succ_i$ and $prec_i$ be the successor and predecessor function on the ordered set of intervals $\mathcal{D}_i$ (in the sense that for any $D_1, D_2 \in \mathcal{D}_i$, $D_1 < D_2$ if $\forall x_1 \in D_1\ \forall x_2 \in D_2 \,.\, x_1 < x_2$). We define $succ_i\big((\theta_i^{p_i}, max_i]\big) = prec_i\big([0, \theta_i^1)\big) = \bot$.

Definition 4 ($R(D)$). *For any domain* $D = (D_1, ..., D_n) \in \mathcal{D}$*, let* $R(D)$ *be the set of regulatory domains that have* D *in their boundary, such that* $R(D) = \{D\}$ *for* $D \in \mathcal{D}^r$*:*

$$R(D) = \Big\{(D'_1, ..., D'_n) \mid reg_i(D_i) \wedge D'_i = D_i \vee \\ switch_i(D_i) \wedge \big(D'_i = prec(D_i) \vee D'_i = succ(D_i)\big)\Big\}$$

Gouzé and Sari [21] define the possible behaviors by the differential inclusion $\dot{\mathbf{x}} \in H(\mathbf{x})$ with

$$H(\mathbf{x}) = \bar{co}\big(\{\boldsymbol{\mu}(D') - \boldsymbol{\nu}(D')\mathbf{x} \mid D' \in R(D)\}\big)$$

where $\bar{co}(E)$ is the smallest closed convex set containing the set E. For any regulatory domain $D \in \mathcal{D}^r$ and $\mathbf{x} \in D$, $H(\mathbf{x}) = \{\boldsymbol{\mu}(D) - \boldsymbol{\nu}(D)\mathbf{x}\}$, that is, the behavior is consistent with Equation (2).

Definition 5 (Trajectory). *A* trajectory *of M is a solution of $\dot{\mathbf{x}} \in H(\mathbf{x})$.*

Qualitative model. The continuous behavior according to Definition 5 can be approximated by a discrete transition graph on the set of domains $\mathcal{D}$ [14, 7] (where the qualitative model of [7] is more precise than [14]). This graph simulates the behavior of the underlying genetic network.

Example 1. Consider the example of two proteins a and b inhibiting each other's production [14], as shown in Figure 1. The respective production rates of proteins a and b are defined by

$$\mu_a = \begin{cases} 20 & \text{if } 0 \leqslant x_a < \theta_a^2 \wedge 0 \leqslant x_b < \theta_b^1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_b = \begin{cases} 20 & \text{if } 0 \leqslant x_a < \theta_a^1 \wedge 0 \leqslant x_b < \theta_b^2 \\ 0 & \text{otherwise} \end{cases}$$

with $\theta_a^1 = \theta_b^1 = 4$ and $\theta_a^2 = \theta_b^2 = 8$. The degradation rate ν of both proteins is always 2. The example is thus modeled by the piecewise linear system $M = \big(\{x_a, x_b\}, \{\theta_a^1, \theta_a^2\} \times \{\theta_b^1, \theta_b^2\}, (\mu_a, \mu_b)^t, diag(\nu, \nu)\big)$.

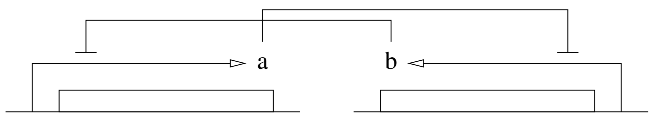

Fig. 1. Two proteins inhibiting each other.

2.2 Transition systems and Constraints

In the following, we present a simplified version of the component model adopted in [20]. For a set of variables X, let $V(X)$ denote the set of valuations of X, and let $\mathcal{P}(X) = 2^{V(X)}$ be the set of predicates on $V(X)$.

Definition 6 (Transition system). *A* transition system *B is a tuple (X, A, G, F) where*

- *X is a finite set of variables;*

– *A is a finite set of actions;*
– *$G : A \to \mathcal{P}(X)$ associates with every action its* guard *specifying when the action can occur;*
– *$F : A \to \big(V(X) \to V(X)\big)$ associates with every action its* transition function*.*

For convenience, we write G^a and F^a for $G(a)$ and $F(a)$, respectively.

Definition 7 (Semantics of a transition system). *A transition system $B = (X, A, G, F)$ defines a* transition relation *$\to$: $V(X) \times A \times V(X)$ such that: $\forall \mathbf{x}, \mathbf{x}' \in V(X)\ \forall a \in A\ .\ \mathbf{x} \xrightarrow{a} \mathbf{x}' \iff G^a(\mathbf{x}) \wedge \mathbf{x}' = F^a(\mathbf{x})$.*

We write $\mathbf{x} \to \mathbf{x}'$ for $\exists a \in A\ .\ \mathbf{x} \xrightarrow{a} \mathbf{x}'$, and $\to^*$ for the transitive and reflexive closure of $\to$. Given states $\mathbf{x}$ and $\mathbf{x}'$, $\mathbf{x}'$ is *reachable* from $\mathbf{x}$ if $\mathbf{x} \to^* \mathbf{x}'$.

Definition 8 (Predecessors). *Given a transition system $B = (X, A, G, F)$ and a predicate $P \in \mathcal{P}(X)$, let the predicate $pre_a(P)$ characterize the* predecessors *of P by action a: $pre_a(P)(\mathbf{x}) \iff G^a(\mathbf{x}) \wedge P\big(F^a(\mathbf{x})\big)$. Let $pre(P) = \bigvee_{a\in A} pre_a(P)$, $pre^0(P) = P$, and $pre^{i+1}(P) = pre(pre^i(P))$, $i \geqslant 0$.*

The predicate $pre_a(P)$ (resp. $pre(P)$) characterizes the states from which execution of a (resp. execution of some action) leads to a state satisfying P.

We define two operations on transition systems: composition and restriction. The composition of transition systems is a transition system again, and so is the restriction of a transition system.

Definition 9 (Composition). *Let $B_i = (X_1, A_i, G_i, F_i)$, $i = 1, 2$, with $X_1 \cap X_2 = \emptyset$ and $A_1 \cap A_2 = \emptyset$. $B_1 \| B_2$ is defined as the transition system $(X_1 \cup X_2, A_1 \cup A_2, G_1 \cup G_2, F_1 \cup F_2)$.*

This is the standard asynchronous product. Restrictions allow to constrain the behavior of a transition system.

Definition 10 (Action constraint). *Given a transition system $B = (X, A, G, F)$, an* action constraint *is a tuple of predicates $U = (U^a)_{a\in A}$ with $U^a \in \mathcal{P}(X)$.*

Definition 11 (Restriction). *The* restriction *of $B = (X, A, G, F)$ with an action constraint $U = (U^a)_{a\in A}$ is the transition system $B/U = (X, A, G', F)$ where for any $a \in A$, $G'(a) = G(a) \wedge U^a$ is the (restricted) guard of a in B/U.*

Example 2. Consider two transition systems $B_i = (\{x_i\}, \{inc_i, dec_i\}, G_i, F_i)$ where x_i are variables on $\{low, high\}$, $G_i(inc_i) = (x_i = low)$, $G_i(dec_i) = (x_i = high)$, $F_i(inc_i) = (x_i := high)$, and $F_i(dec_i) = (x_i := low)$, $i = 1, 2$. The composition is $B_1 \| B_2 = (\{x_1, x_2\}, \{inc_1, inc_2, dec_1, dec_2\}, G_1 \cup G_2, F_1 \cup F_2)$.

Further suppose that we want to prevent B_1 from entering state $x_1 = high$ if $x_2 = high$, and vice versa. This can be done by restricting $B_1 \| B_2$ with action constraint $U = (U^{inc_1}, U^{inc_2}, U^{dec_1}, U^{dec_2})$ where $U^{inc_1} = (x_2 = low)$, $U^{inc_2} = (x_1 = low)$, and $U^{dec_1} = U^{dec_2} = true$. The restricted system is $(B_1 \| B_2)/U = (\{x_1, x_2\}, \{inc_1, inc_2, dec_1, dec_2\}, G', F_1 \cup F_2)$ with $G'(inc_1) = G_1(inc_1) \wedge (x_2 = low)$, $G'(inc_2) = G_2(inc_2) \wedge (x_1 = low)$, $G'(dec_1) = G_1(dec_1)$, and $G'(dec_2) = G_1(dec_2)$.

Definition 12 **(*incr*, *decr*).** *Given a predicate P on $\mathcal{D}$ and $i \in \{1, ..., n\}$, we define the predicates $incr_i(P)$ and $decr_i(P)$ such that for any domain $D = (D_1, ..., D_i, ..., D_n) \in \mathcal{D}$, $incr_i(P)(D) = P\big(D_1, ..., succ_i(D_i), ..., D_n\big)$ if $succ_i(D_i) \neq \bot$, and $incr_i(P)(D) =$ false otherwise. Similarly, let $decr_i(P)(D) = P\big(D_1, ..., prec_i(D_i), ..., D_n\big)$ if $prec_i(D_i) \neq \bot$, and $decr_i(P)(D) =$ false otherwise.*

Intuitively, $incr_i(P)$ and $decr_i(P)$ denote the predicate P "shifted" by one domain along the i-th dimension, towards lower and higher values, respectively. For instance, consider predicate $P = (x_a = \theta_a^2)$ on the state space of Example 1. Then, $incr_a(P) = (\theta_a^1 < x_a < \theta_a^2)$ and $decr_b(P) = (x_a = \theta_a^2 \wedge \theta_b^1 \leqslant x_b \leqslant max_b)$.

2.3 Component Model of Genetic Networks

We now propose the construction of a component-based model from a piecewise linear system.

Definition 13 **(*eq*).** *Given $\theta = \theta_1 \times ... \times \theta_n$, we define predicates $eq_i^{\#}$ on $\mathcal{D}$, $i \in \{1, ..., n\}$, $\# \in \{<, \leqslant, \geqslant, >\}$ such that for any domain $D = (D_1, ..., D_n) \in \mathcal{D}$,*

$$eq_i^{\#}(D) \iff \exists D' = (D'_1, ..., D'_n) \in R(D)\ \forall x \in D'_i \ .\ \phi_i(D') \# x \text{ for } \# \in \{<, >\}$$
$$eq_i^{=}(D) \iff \exists D' = (D'_1, ..., D'_n) \in R(D)\ \exists x \in D'_i \ .\ \phi_i(D') = x$$

and $eq_i^{\leqslant} = eq_i^{<} \vee eq_i^{=}$, $eq_i^{\geqslant} = eq_i^{=} \vee eq_i^{>}$.

The predicates $eq_i^{\#}$ reflect the relative position of target equilibria of the adjacent regulatory domains. The predicates $eq_i^{<}(D)$ and $eq_i^{>}(D)$ specify when some adjacent regulatory domain has its target equilibrium "left" of D_i and "right" of D_i, respectively.

Definition 14 **($\check{C}(M)$).** *Given a piecewise linear system $M = (X, \theta, \boldsymbol{\mu}, \boldsymbol{\nu})$ with $|X| = n$, we define the transition system $\check{C}(M) = (B_1 \| B_2 \| ... \| B_n)/U$ as follows.*

- *$\forall i = 1, ..., n$. $B_i =$ `counter`$(\mathcal{D}_i)$, where `counter`$(\mathcal{D}_i)$ is a bounded counter defined on $\mathcal{D}_i(\theta)$ by the transition system `counter`$(\mathcal{D}_i) = \big(\{level_i\}, \{inc_i, dec_i\}, \{G^{inc_i} = level_i \leqslant \theta_i^{p_i}, G^{dec_i} = level_i \geqslant \theta_i^1\}, \{F^{inc_i} = \big(level_i := succ_i(level_i)\big), F^{dec_i} = \big(level_i := prec_i(level_i)\big)\}\big)$.*
- *U is an action constraint such that $U(inc_i) = V_i^{>}$ and $U(dec_i) = V_i^{<}$ with*

$$V_i^{<} = reg \wedge eq_i^{<} \ \vee \ decr_i(reg \wedge eq_i^{\leqslant}) \tag{3}$$
$$V_i^{>} = reg \wedge eq_i^{>} \ \vee \ incr_i(reg \wedge eq_i^{\geqslant}) \tag{4}$$

Actions inc_i (dec_i) correspond to an increase (decrease) by one of the discretized concentration $level_i$ of protein i. The predicates $V_i^{<}$ and $V_i^{>}$ specify when a transition decrementing $level_i$ and incrementing $level_i$, respectively, is enabled. More precisely, the first term in the disjunctions of lines (3) and (4) specifies that there is a transition from a regulatory domain to a first-order switching domain in the direction of the target equilibrium of the source domain. The second term gives the conditions for transitions decreasing the order: they must be compatible with the relative position of the target equilibrium of the destination domain. Definition 14 limits the behavior of the model to transitions between regulatory and first-order switching domains. The generalization to the set of domains $\mathcal{D}$ is not presented here due to space limitation.

Remark 1. Since $\|$ is associative, Definition 14 leaves open how the system is actually partitioned into components (in the sense of sets of transition systems). The two extreme cases are that each B_i is considered as one component, or that $B_1 \| B_2 \| ... \| B_n$ is considered as one single component. This choice will usually depend on the degree of interaction between the modeled proteins. Putting all proteins in one component amounts to a non-modular model leading to non-compositional analysis. Representing each protein with a separate component may lead to a too heavy abstraction of the behavior. A good choice may gather closely interacting proteins, for instance proteins in the same cell, in one component, while modeling neighboring cells as separate components.

Notice that the above modeling framework enforces *separation of concerns* by making a clear distinction between the behaviors of the individual components, and constraints between the components.

Example 3. Figure 2 shows the transition relations of $counter(\mathcal{D}_a)$, $counter(\mathcal{D}_b)$, and $\check{C}(M)$ for the piecewise linear system M of Example 1.

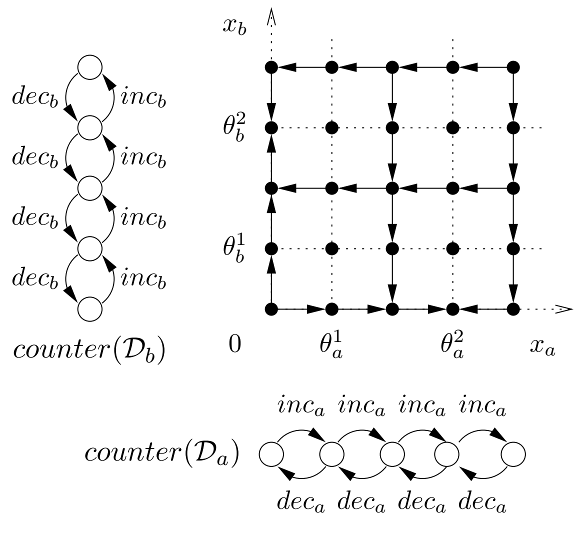

Fig. 2. The transition relations of $counter(\mathcal{D}_a)$, $counter(\mathcal{D}_b)$, and $\check{C}(M)$.

Theorem 1 (Correctness). *Consider a piecewise linear system $M = (X, \theta, \boldsymbol{\mu}, \boldsymbol{\nu})$. The behavior of $\check{C}(M)$ under-approximates qualitative simulation as defined in [14, 7].*

3 Compositional Reachability Analysis

Based on the transition system $\check{C}(M)$, the compositional algorithm shown below can be used to check for reachability of a goal domain, or set of domains, from an initial domain. The algorithm exhibits a path, if one is found, that solves the reachability problem.

In the sequel we consider a system $B = (X, A, G, F) = (B_1 \| \dots \| B_N)/U$ with $B_i = (X_i, A_i, G_i, F_i)$, $i \in K = \{1, \dots, N\}$, and U an action constraint. That is, we suppose the n proteins to be modeled with N $(1 \leqslant N \leqslant n)$ components, according to Remark 1. Given a conjunction $c = c_1 \wedge ... \wedge c_N$ of predicates $c_i \in \mathcal{P}(X_i)$, $i = 1, ..., N$, let $c[i] = c_i$ denote the *projection* of c on X_i.

Let $path_k : V(X_k) \times \mathcal{P}(X_k) \rightarrow 2^{A_k}$ be a function on component k telling which action to take in order to get from some current component state towards a state satisfying some predicate. This function can be computed locally: for any predicate $P \in \mathcal{P}(X_k)$ and domain D, let

$$\begin{aligned} path_k(D[k], P) = \{a \in A_k \mid \exists i \geqslant 0 \,.\, pre_a\big(pre_k^i(P)\big)(D[k]) \wedge \\ \forall j \in \{0, ..., i\} \,.\, \neg pre_k^j(P)(D[k])\} \end{aligned}$$

That is, $path_k(D[k], P)$ contains an action a if and only if executing a from $D[k]$ will bring component k closer to P.

For a set of actions A, let $enabling(A)$ be a list of predicates enabling some action in A: $\forall c \in enabling(A) \,.\, c \implies \bigvee_{a \in A} G(a)$. We suppose each of these predicates to be a conjunction of predicates on the components. Let $\oplus$ denote list concatenation. Given a non-empty list l, we write $l = e.l'$ where e is the first element, and l' the rest of the list. Given a list A of actions and a domain D, let $first\_enabled(A, D)$ be the first action a of A such that $G(a)(D)$, and $first\_enabled(A, D) = \perp$ if all actions are disabled.

Algorithm 1 is constructive, that is, it establishes reachability from some initial domain D_{init} to a set of domains P by constructing a path from D_{init} to P. Function **move** works as follows. It takes as arguments the current domain D, a predicate to be reached in the form of a list d of conjunctions, and a list *good* of all actions requested to be executed, and returns a new domain, the part of the path constructed so far, and a boolean indicating whether a path was found. The five cases are (1) if the current domain satisfies the predicate to be reached, then we are done. (2) Otherwise, execute the first action in *good* that is enabled. If there is none, compute the set *goal* of actions not considered so far that bring the system closer to the first element c of d. (3) If *goal* is non-empty, recursively call **move** so as to reach some domain D' enabling some action in *goal*, then call **move** once more to continue moving towards c. (4) If reaching c fails, try the next conjunction of d. (5) If all above fails, then this call of **move**

Algorithm 1 Initial call to construct a path σ from domain D_{init} to predicate P: $(D', \sigma, success) = \mathbf{move}(D_{init}, P, \langle\rangle)$.

```
$\mathbf{move}\ (D, c.l, good) =$
    $(D, \langle\rangle, true)$                         if $c(D)$                                                                      (1)
    $(F(a)(D), \langle a\rangle, true)$             if $\neg c(D) \wedge a \neq \bot$                                                      (2)
    $(D'', \sigma \oplus \sigma', true)$                if $\neg c(D) \wedge a = \bot \wedge goal \neq \emptyset \wedge ok \wedge ok'$                       (3)
    $\mathbf{move}(D, l, good)$                      if $\neg c(D) \wedge a = \bot \wedge (goal = \emptyset \vee \neg(ok \wedge ok')) \wedge l \neq \langle\rangle$   (4)
    $(D, \langle\rangle, false)$                        $otherwise$                                                                    (5)
where
```

$$\begin{aligned} a &= first\_enabled(good, D) \\ goal &= \bigcup_k path_k(D[k], c[k]) \setminus good \\ (D', \sigma, ok) &= \mathbf{move}\big(D, enabling(goal), good \oplus goal\big) \\ (D'', \sigma', ok') &= \mathbf{move}(D', c, good) \end{aligned}$$

failed. It can be shown that Algorithm 1 is guaranteed to terminate. It is not guaranteed to find a path even if one exists, though. If a path is found on $\check{C}(M)$, then Theorem 1 ensures that the same path exists in the qualitative model of [7].

Algorithm 1 is compositional in the sense that it independently computes local paths through the state spaces of the components (line $goal = \bigcup_k path_k(D[k], c[k]) \setminus good$). A global path is then constructed from the local paths and the constraints between the components: when an action a to be executed is blocked by a constraint involving other components, the algorithm is called recursively to move the blocking components into a domain where a is enabled.

Example 4 (Example 3 continued.). The functioning of Algorithm 1 is illustrated by the path construction from domain $D_{init} = (\theta_a^1 < x_a < \theta_a^2 \wedge \theta_b^1 < x_b < \theta_b^2)$ to domain $D_{goal} = (x_a = \theta_a^2 \wedge 0 \leqslant x_b < \theta_b^1)$ representing a stable equilibrium. The subsequent calls of **move** are

```
$\mathbf{move}\ (D_{init}, \langle D_{goal}\rangle, \langle\rangle)$
    $a = \bot$, $goal = \{inc_a, dec_b\}$
    $\mathbf{move}\ (D_{init}, \langle\theta_a^1 < x_a < \theta_a^2, \ldots\rangle, \langle inc_a, dec_b\rangle)$
        $= (D_1 = (\theta_a^1 < x_a < \theta_a^2 \wedge x_b = \theta_b^1), \langle dec_b\rangle, true)$                 (2)
    $\mathbf{move}\ (D_1, \langle D_{goal}\rangle, \langle\rangle)$
        $a = \bot$, $goal = \{inc_a, dec_b\}$
        $\mathbf{move}\ (D_1, \langle\theta_a^1 < x_a < \theta_a^2, \ldots\rangle, \langle inc_a, dec_b\rangle)$
            $= (D_2 = (\theta_a^1 < x_a < \theta_a^2 \wedge 0 \leqslant x_b < \theta_b^1), \langle dec_b\rangle, true)$     (2)
        $\mathbf{move}\ (D_2, \langle D_{goal}\rangle, \langle\rangle)$
            $a = \bot$, $goal = \{inc_a\}$
            $\mathbf{move}\ (D_2, \langle x_a < \theta_a^2 \wedge 0 \leqslant x_b < \theta_b^1\rangle, \langle inc_a\rangle)$
                $= (D_{goal}, \langle inc_a\rangle, true)$                                           (2)
            $\mathbf{move}\ (D_{goal}, \langle D_{goal}\rangle, \langle\rangle) = (D_{goal}, \langle\rangle, true)$
```

$$= (D_{goal}, \langle inc_a \rangle, true) \tag{3}$$
$$= (D_{goal}, \langle dec_b, inc_a \rangle, true) \tag{3}$$
$$= (D_{goal}, \langle dec_b, dec_b, inc_a \rangle, true) \tag{3}$$

Thus, D_{goal} is reached from D_{init} by decrementing $level_b$ twice and then incrementing $level_a$.

4 Case Study: Delta-Notch Cell Differentiation

Cell differentiation by delta-notch lateral inhibition is a well-studied genetic network [24, 18]. Cell differentiation is an important step in embryonic development, as it causes initially uniform cells to assume different functions.

For each cell we consider the concentrations of two trans-membrane proteins, *Delta* and *Notch*. Following the model provided in [24], high concentrations of *Delta* and *Notch* inhibit each other's production within the same cell. High *Delta* levels activate further *Delta* production in the same cell and *Notch* production in the neighboring cells. Figure 3 illustrates these interactions.

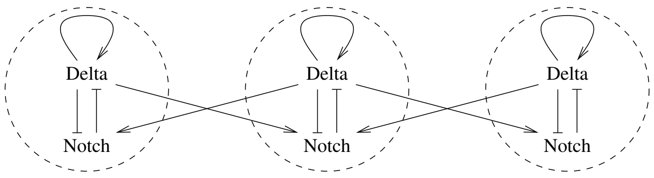

Fig. 3. Interactions within and between neighbor cells.

For our case study, we consider a network consisting of 19 cells with the layout shown in Figure 4, a network of 37 cells with a similar layout, and the network of 49 cells shown in Figure 4.

For each protein we partition the continuous state space into two intervals and one threshold value: $\mathcal{D}_\Delta = \{[0, \theta_\Delta), \{\theta_\Delta\}, (\theta_\Delta, max_\Delta]\}$ and $\mathcal{D}_N = \{[0, \theta_N), \{\theta_N\}, (\theta_N, max_N]\}$. Cells with low *Delta* and high *Notch* levels ($0 \leqslant \Delta < \theta_\Delta$, $\theta_N < Notch \leqslant max_N$) are undifferentiated, whereas cells with high *Delta* and low *Notch* concentrations ($\theta_\Delta < \Delta \leqslant max_\Delta$, $0 \leqslant Notch < \theta_N$) are differentiated. We are not interested in the actual production and degradation rates of the proteins but require the target equilibria ϕ_{Δ_i} and ϕ_{Notch_i} to satisfy

$$\begin{aligned} 0 \leqslant \phi_{\Delta_i} < \theta_\Delta &\text{ if } Notch_i > \theta_N \\ \theta_\Delta < \phi_{\Delta_i} \leqslant max_\Delta &\text{ if } Notch_i < \theta_N \\ 0 \leqslant \phi_{Notch_i} < \theta_N &\text{ if } \max\{\Delta_j \mid j \in neighbors(i)\} < \theta_\Delta \\ \theta_N < \phi_{Notch_i} \leqslant max_N &\text{ if } \max\{\Delta_j \mid j \in neighbors(i)\} > \theta_\Delta \end{aligned}$$

Considering only regulatory and first-order switching domains for a system modeling n cells, the $2n$-dimensional global state space encompasses 4^n regula-

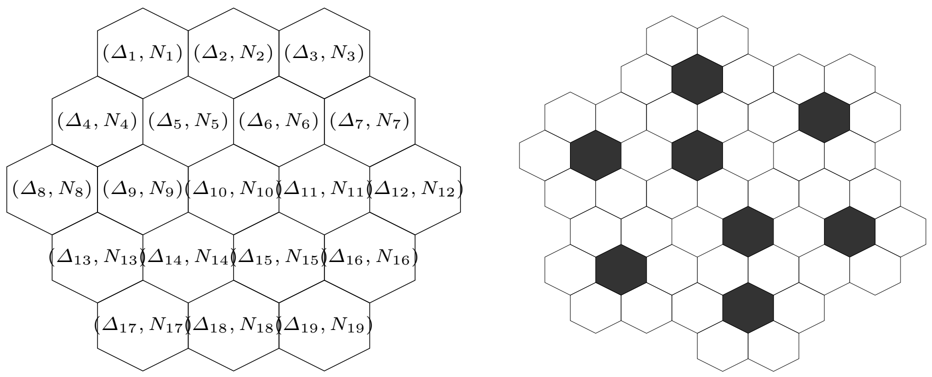

Fig. 4. Model of 19 communicating cells (left); a stable equilibrium state involving 49 cells where dark cells are differentiated (right).

tory domains and $2n \times 2^{2n-1}$ first-order switching domains, that is, 5.5×10^{12} states for 19 cells, 7.2×10^{23} states for 37 cells, and 1.6×10^{31} states for 49 cells.

We have implemented Algorithm 1 in the compositional verification tool PROMETHEUS. To start, we choose to represent each cell by one component, and check reachability of a given stable equilibrium from the initial state where all cells are non differentiated. The results reported by PROMETHEUS are consistent with the actual, experimentally observed behavior [24]. For the case of 49 cells and the state shown in Figure 4, PROMETHEUS finds a path of length 32 reaching the state.

Table 1 shows the execution times for the models of cell differentiation with 19, 37, and 49 cells, and for models of the nutritional stress response of E. coli [4] and sporulation initiation of B. subtilis taken from [12]. The subsequent columns show the number of domains of the model, and the times for constructing the component model and a path to the final state using Algorithm 1. All measurements have been made on the same machine, a Pentium4 at 3 GHz with 512 MB of memory.

	state space	model	reachability
E. coli	7.8×10^3	< 10 ms	0.02 s
B. subtilis	2.7×10^4	< 10 ms	0.28 s
Delta-Notch 19	5.5×10^{12}	0.01 s	1.06 s
Delta-Notch 37	7.2×10^{23}	0.05 s	10.8 s
Delta-Notch 49	1.6×10^{31}	0.13 s	7.5 s

Table 1. Performance on different models.

In order to evaluate the performance increase due to compositionality, we compare the compositional approach with a non-compositional reachability analysis, using the same framework. More precisely, we use Algorithm 1 to find a path from the initial, undifferentiated state to the state of Figure 4, on different instances of the Delta-Notch model with 49 cells. The only parameter that varies is the size of the components, where extreme cases are given by the model of 98 components each modeling one protein, and the model consisting of one single component. The measured performance is shown in Table 2. For this example, the optimal degree of modularity lies around one component per cell. It should be noted that the optimal partitioning of proteins into components depend on the system, and cannot be easily generalized. For a higher degree of modularity (1 component per protein), the algorithm performs somewhat slower, probably due to an overhead in coordination between closely interacting components. As the component size increases, complexity of the (non compositional) path construction within the components exponentially blows up. Although the algorithm used for path construction within a component is not designed to be optimal for large state spaces, it allows to compare the complexity for different degrees of granularity.

cells per component	0.5	1	3/4	7	9/10	49
reachability	10.7 s	7.5 s	8.4 s	35.5 s	(*)	(*)

Table 2. Benchmarks for different levels of modularity of Delta-Notch 49. (*): computation interrupted after 12 hours.

5 Discussion

We have presented a novel approach for component-based modeling and reachability analysis of genetic regulatory networks. The model discretizes the network dynamics defined by a system of piecewise linear differential equations. On this model, a compositional algorithm constructively analyzes reachability properties, allowing to deal with complex, high-dimensional systems. A case study and several benchmarks show the potential of this approach. In spite of the conservative approximation, our approach has yielded the expected results in the case studies carried out so far, and confirmed its efficiency.

We intend to apply the technique to genetic networks involving a hierarchy of communicating functional modules, and to models of not yet fully understood networks. We are currently investigating compositional analysis of further properties like equilibria and cyclic behavior, based on the same component model. In order to further improve precision, we intend to study the integration of the qualitative model of [3] using piecewise *affine* differential equations in our framework.

Acknowledgment. The author thanks Hidde de Jong for many fruitful discussions and comments on earlier versions of this work.

References

1. R. Alur, T. Dang, and F. Ivancić. Reachability analysis of hybrid systems via predicate abstraction. *Trans. on Embedded Computing Systems*, 2004.
2. G. Batt, D. Bergamini, H. de Jong, H. Garavel, and R. Mateescu. Model checking genetic regulatory networks using GNA and CADP. In *proc. SPIN'04*, volume 2989 of *LNCS*, pages 158–163. Springer-Verlag, 2004.
3. G. Batt, H. de Jong, J. Geiselmann, M. Page, D. Ropers, and D. Schneider. Symbolic reachability analysis of genetic regulatory networks using qualitative abstraction. Research Report 5362, INRIA, France, 2004.
4. G. Batt, D. Ropers, H. de Jong, J. Geiselmann, M. Page, and D. Schneider. Qualitative analysis and verification of hybrid models of genetic regulatory networks: Nutritional stress response in escherichia coli. In *proc. HSCC'05*, volume 3414 of *LNCS*, pages 134–150. Springer-Verlag, 2005.
5. G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. Application of formal methods to biological regulatory networks: Extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–348, 2004.
6. R. Blossey, L. Cardelli, and A. Phillips. A compositional approach to the stochastic dynamics of gene networks. *Trans. on Comput. Syst. Biol.*, 4:99–122, 2006.
7. R. Casey, H. de Jong, and J.-L. Gouzé. Piecewise-linear models of genetic regulatory networks: Equilibria and their stability. *Mathematical Biology*, 52(1):27–56, 2006.
8. N. Chabrier and F. Fages. Symbolic model checking of biochemical networks. In *proc. CMSB'03*, 2003.
9. C. Chaouiya, E. Remy, P. Ruet, and D. Thieffry. Qualitative modelling of genetic networks: From logical regulatory graphs to standard petri nets. In J. Cortadella and W. Reisig, editors, *proc. ICATPN'04*, volume 3099 of *LNCS*, pages 137–156. Springer-Verlag, 2004.
10. J.-P. Comet, H. Klaudel, and S. Liauzu. Modeling multi-valued genetic regulatory networks using high-level petri nets. In G. Ciardo and P. Darondeau, editors, *proc. ICATPN'05*, volume 3536 of *LNCS*, pages 208–227. Springer-Verlag, 2005.
11. H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):69–105, 2002.
12. H. de Jong, J. Geiselmann, G. Batt, C. Hernandez, and M. Page. Qualitative simulation of the initiation of sporulation in Bacillus subtilis. *Bulletin of Mathematical Biology*, 66(2):261–300, 2004.
13. H. de Jong, J. Geiselmann, C. Hernandez, and M. Page. Genetic Network Analyzer: Qualitative simulation of genetic regulatory networks. *Bioinformatics*, 19(3):336–344, 2003.
14. H. de Jong, J.-L. Gouzé, C. Hernandez, M. Page, T. Sari, and J. Geiselmann. Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bulletin of Mathematical Biology*, 66:301–340, 2004.
15. S. Eker, M. Knapp, K. Laderoute, P. Lincoln, and C. Talcott. Pathway logic: Executable models of biological networks. *ENTCS*, 71, 2002.
16. F. Fages, S. Soliman, and N. Chabrier-Rivier. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *J. of Biological Physics and Chemistry*, 4(2):64–72, 2004.
17. A.F. Filippov. Differential equations with discontinuous righthand side. *Mathematics and its Applications*, 18, 1988.

18. R. Ghosh, A. Tiwari, and C. Tomlin. Automated symbolic reachability analysis; with application to delta-notch signaling automata. In O. Maler and A. Pnueli, editors, *proc. HSCC'03*, volume 2623 of *LNCS*, pages 233–248. Springer-Verlag, 2003.
19. L. Glass and S.A. Kauffman. The logical analysis of continuous, non-linear biochemical control networks. *Journal of Theoretical Biology*, 39(1):103–129, 1973.
20. G. Gössler and J. Sifakis. Component-based construction of deadlock-free systems (extended abstract). In *proc. FSTTCS'03*, volume 2914 of *LNCS*. Springer-Verlag, 2003.
21. J.-L. Gouzé and T. Sari. A class of piecewise linear differential equations arising in biological models. *Dynamical Systems*, 17(4):299–316, 2003.
22. M. Heiner and I. Koch. Petri net based model validation in systems biology. In J. Cortadella and W. Reisig, editors, *proc. ICATPN'04*, volume 3099 of *LNCS*, pages 216–237. Springer-Verlag, 2004.
23. A. Larrinaga, A. Naldi, L. Sánchez, D. Thieffry, and C. Chaouiya. GINsim: a software suite for the qualitative modelling, simulation and analysis of regulatory networks. *Biosystems*, 2005.
24. G. Marnellos, G.A. Deblandre, E. Mjolsness, and G. Kintner. Delta-Notch lateral inhibitory patterning in the emergence of ciliated cells in *Xenopus*: Experimental observations and a gene network model. In *proc. PSB'00*, volume 5, pages 326–337. World Scientific Publishing, 2000.
25. Z.N. Oltvai and A.L. Barabási. Life's complexity pyramid. *Science*, 298:763–764, 2002.
26. C. Piazza, M. Antoniotti, V. Mysore, A. Policriti, F. Winkler, and B. Mishra. Algorithmic algebraic model checking I: Challenges form systems biology. In *proc. CAV'05*, volume 3576 of *LNCS*, pages 5–19. Springer-Verlag, 2005.
27. O. Resendis-Antonio, J.A. Freyre-González, R. Menchaca-Méndez, R.M. Gutiérrez-Ríos, A. Martínez-Antonio, C. Avila-Sánchez, and J. Collado-Vides. Modular analysis of the transcriptional regulatory network of e. coli. *Trends in Genetics*, 21(1):16–20, 2005.
28. E. Simão, E. Remy, D. Thieffry, and C. Chaouiya. Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in e.coli. *Bioinformatics*, 21(Suppl. 2):ii190–ii196, 2005.
29. R. Thomas. Boolean formalisation of genetic control circuits. *J. Theor. Biol.*, 42:565–583, 1973.