Composition and abstraction of logical regulatory modules: application to multicellular systems

Nuno D. Mendes[1,*], Frédéric Lang[2], Yves-Stan Le Cornec[3], Radu Mateescu[2], Grégory Batt[4], Claudine Chaouiya[1,5,*]

[1] IGC, Instituto Gulbenkian de Ciência, Rua da Quinta Grande 6, P-2780-156 Oeiras, PORTUGAL
[2] Inria Grenoble – Rhône-Alpes, 655 av de l'Europe, Montbonnot, 38 334 St Ismier Cedex, FRANCE
[3] IBISC, Université d'Evry Val d'Essonne, 40, Rue du Pelvoux, 91 020 Evry Cedex, FRANCE
[4] Inria Paris – Rocquencourt, Domaine de Voluceau, 78 153 Le Chesnay, FRANCE
[5] TAGC, INSERM U1090 AMU, 163 avenue de Luminy, 13 288 Marseille Cedex 9, FRANCE

July 15, 2014

Abstract

Motivation: Logical (Boolean or multi-valued) modelling is widely employed to study regulatory or signalling networks. Even though these discrete models constitute a coarse, yet useful, abstraction of reality, the analysis of large networks faces a classical combinatorial problem. Here, we propose to take advantage of the intrinsic modularity of inter-cellular networks to set up a compositional procedure that enables a significant reduction of the dynamics, yet preserving the reachability of stable states. To that end, we rely on process algebras, a well-established computational technique for the specification and verification of interacting systems.

Results: We develop a novel compositional approach to support the logical modelling of interconnected cellular networks. First, we formalise the concept of logical regulatory modules and their composition. Then, we make this framework operational by transposing the composition of logical modules into a process algebra framework. Importantly, the combination of incremental composition, abstraction and minimisation using an appropriate equivalence relation (here the safety equivalence) yields huge reductions of the dynamics. We illustrate the potential of this approach with two case-studies: the Segment-Polarity and the Delta-Notch modules.

Availability and Implementation: GINsim[1] and CADP[2] are freely available for academic users. Files needed to reproduce our results are provided at `http://compbio.igc.gulbenkian.pt/nmd/node/45`.

Contact: chaouiya@igc.gulbenkian.pt

1 Introduction

The growing number of published models shows the suitability of qualitative logical modelling to study regulatory and signalling networks (*e.g.*, [3, 12, 24, 27]). However, when dealing with large networks, a classical combinatorial explosion arises, hampering efficient analyses of the dynamical properties of these systems, in particular, reachability properties. Although efficient algorithms have been proposed to identify all stable states via static analysis (*e.g.*, [9, 10, 23]), reachability analysis is hard to perform because it requires exploring the dynamics. To address this issue, we propose to rely on the concepts of modularity

*to whom correspondence should be addressed
[1] `http://ginsim.org`
[2] `http://cadp.inria.fr`

and compositionality, focusing on the multi-valued logical formalism, initially defined by R. Thomas and co-workers [33].

Modularity has emerged as a key feature of molecular networks (*e.g.* [36]), but a precise definition of functional regulatory modules is still lacking, let alone a method to decompose large, intricate networks into such functional modules. Nevertheless, when cellular patterns are governed by both inter-cellular signals and intra-cellular regulatory networks, it is natural to consider each cellular network as a module. Moreover, based on previous knowledge, sub-networks are often attributed specific functions within complex cellular processes. This is the case of the cell cycle control for which specific modules are associated to check-points, entry or exit control of specific phases, etc. In [12], a logical model was defined by combining three modules involved in the control of the budding yeast cell cycle. Logical modelling was also applied to multi-cellular networks controlling early embryonic developmental processes in *Drosophila* [6, 15, 29].

The composed models cited above were manually defined from smaller modules. Indeed, little work on model composition has been carried out, although it has been identified as a major goal in systems biology [32]. Generally, existing composition procedures are not automatic and aim at properly defining a composed model, while little attention is paid to the analysis of the (potentially very large) associated dynamics [26, 30, 31]. In any case, these studies mainly focus on biochemical reaction networks. Concerning the composition of logical regulatory modules, a framework based on high-level Petri nets has been developed, providing a formal and systematic procedure [5]. However, it still does not solve the scalability issue of the analysis since the resulting composed models have still to be analysed in a monolithic manner.

[26] consider models in the form of sets of nonlinear differential equations and propose three different ways of combining sub-models: composition, fusion and aggregation. Model composition and fusion, respectively, keep or eliminate references to the original sub-models, while model aggregation requires that individual sub-models come with their input and output ports (similar to our definition of logical regulatory modules). These notions are important to distinguish different types of model combination, but, again, they do not address the issue of analysing the resulting models. In this paper, we simply refer to the combination of models as model composition.

Process algebras aim at representing and analysing complex interacting (discrete) systems. These approaches have led to the development of compositional approaches that mitigate the combinatorial explosion problem through the minimisation of the individual dynamics while preserving properties of interest, and then by incrementally composing and minimising intermediate dynamics, until a description of the global behaviour is obtained.

Process algebras have already been considered for biological processes (see, *e.g.*, [7] and references therein), adopting different modelling approaches.

Recently, a compositional algorithm was proposed for gene regulatory networks modelled as piecewise-linear ODE systems to check the reachability of a specific steady state from an initial condition [16]. In contrast, we determine *all* stable states reachable from a given initial condition (see Section 4 for a comparison of the two approaches).

In this paper, we propose a new computational approach to cope with the combinatorial explosion that hampers proper analyses of models defined as compositions of logical regulatory modules. For simplicity, we restrict ourselves to the composition of identical modules, which is often the case of multi-cellular systems. In any case, this constitutes an important class of applications that notably includes most of the patterning problems in developmental biology. Furthermore, we are concerned with the reachability of stable states that is a property of real interest for differentiation regulatory networks.

We thus define a framework to compose logical modules by means of logical integration functions describing how modules interact with their neighbours. We establish a constructive method to determine the dynamics of the composition from the dynamics of the individual modules and their interactions. Then, we rely on process algebra techniques to generate, abstract and minimise the behaviours of the modules, yet preserving reachability properties. These dynamics are iteratively combined and minimised, to obtain a final, reduced description of the dynamics of the composed model. Minimisation relies on an equivalence relation, which is chosen depending on the property to be preserved. Here, we rely on the safety equivalence that preserves the reachability of the stable states, while ensuring significant reductions of the dynamics.

For the implementation, we use the CADP toolbox (Construction and Analysis of Distributed Processes [14]) to specify the dynamics of the logical modules and to implement operations of abstraction, minimisation

and incremental composition.

The paper is organised as follows. First, in Section 2, we introduce the modelling framework with formal definitions of Logical Regulatory Module (LRM), logical integration function, composition rules, as well as the corresponding dynamics. Section 3 presents the principles of abstraction and minimisation, relying on classical process algebra operations; other implementation aspects are also briefly discussed. Section 4 includes the application of our procedure to the Segment-Polarity and the Delta-Notch modules. These case-studies illustrate the potential of our approach to analyse crucial properties of composed LRMs dynamics that are far too large to be comprehensively tackled with currently available tools.

2 Methods

This section introduces Logical Regulatory Modules (LRMs) and their composition, which results in a unique LRM. Furthermore, it presents the dynamics of LRMs represented as State Transition Graphs (STGs). Importantly, we prove that composing the individual dynamics is equivalent to constructing the dynamics of the composed model (formal definitions and proof of the theorem are provided in the Supplementary Materials).

2.1 Logical regulatory modules and their dynamics

A schematic representation of a four-components LRM is displayed in Fig. 1-A. In this view, a LRM is an open system with two kinds of components. *Proper components* and their regulatory interactions define the internal dynamics of the system, while *input components* represent external stimuli. Proper components are subject to regulatory effects of other components, whereas input components are unconstrained (their values may be set by the environment or by other modules during the composition).

Definition 1. *A* Logical Regulatory Module *(LRM) is defined by a triplet $N = (G, U, K)$, where:*

- $G = \{g_i\}_{i \in L_G}$ *is the indexed set of the* proper components *(L_G is the corresponding set of indices);* $U = \{u_i\}_{i \in L_U}$ *is the indexed set of the* input components *(L_U is the corresponding set of indices);* $C = G \cup U = \{c_i\}_{i \in L_C}$ *is the set of all the components.*
- *Each component $c_i \in C$ is associated with a domain $D_i = \{0, \dots, M_i\} \subsetneq \mathbb{N}$ and the variable $v_i \in D_i$ denotes its level. The state space S is given by $\prod_{i \in L_C} D_i$ and $v \in S$ denotes a state.*
- $K = (K_i)_{i \in L_G}$ *are the logical regulatory functions of the proper components; $\forall g_i \in G$, $K_i : S \to D_i$, and $K_i(v)$ is the target value of g_i in state v,* i.e. *the value towards which it evolves.*

Briefly, the variable associated to a component of a LRM (gene, protein, etc.) represents its functional level (*e.g.* activity or concentration). Generally, this variable is Boolean, but some situations require additional values (see [34]). The logical regulatory function defines the evolution of the corresponding proper components depending on the levels of their regulators. Hence, given a state (current levels of all the components), some components may remain stable, while others may be called to change their values, giving rise to state transitions.

The asynchronous dynamics of a LRM is represented by a STG (see Def. S1 in the Supplementary Materials and Fig. 1–B for an illustration). The successors of a state v are defined by all transitions going out of v: *input transitions* (towards states that differ from v only by the value of an input component) and *proper transitions* over components $g_i \in G$ such that $K_i(v) - v_i \neq 0$ (*i.e.* called to update in state v). In this discrete framework, stable states are those with no outgoing transitions. Because input components freely vary, there are no such states for a LRM with inputs: each state is connected to all states that differ only by the value of an input variable. This leads to the definition of *strong* and *weak* stable states with respect to the proper components (see Fig. 1–B and [22]): in strong stable states, proper components remain stable whatever the variation of the inputs, while in weak stable states, proper components are stable only for specific values of the inputs.

In practice, since we are interested in reachability properties from an initial condition s_0, instead of the full STG, we consider only the sub-graph that is reachable from s_0 (Def. S2 in the Supplementary Materials).

2.2 Module composition

In order to compose LRMs, one needs to specify how they influence each other. This is done by first specifying a *neighbourhood relation* for each input component, which is then mapped to proper components of its neighbouring modules. This *mapping*, along with the specification of a *logical integration function*, determines how signals are combined. Thus, the evolution of mapped inputs depends on the evolution of the arguments of their integration function, whereas unmapped (free) inputs remain unconstrained. Finally, regulatory effects of mapped inputs are replaced by the (integrated) regulatory effects from the components they are mapped to. This amounts to adequately redefining the logical functions of proper components regulated by mapped inputs, which are removed (reduced) following the reduction method introduced in [25].

Definition 2 below formalises the composition of r LRMs $\left(N^{(k)}\right)_{k=1,\dots,r}$. For simplicity, we assume that for all distinct k, k' (in $\{1,\dots,r\}$) $L_{C^{(k)}} \cap L_{C^{(k')}} = \emptyset$ (in other words, all indices are distinct). As a consequence, indices uniquely correspond to components. Hence all objects associated to a component g_i will be indexed by i without any confusion. We also introduce an additional notation: $\forall X \subsetneq C, S_{|X} \triangleq \prod_{i\in X} D_i$ and $v_{|X}$ is such that $(v_{|X})_i \triangleq v_i$, $\forall i \in X$.

Definition 2. *Consider r LRMs $N^{(k)} = \left(G^{(k)}, U^{(k)}, K^{(k)}\right)$, $k = 1,\dots,r$ and the composition rule $\mathcal{M}$ defined over the set of all the input components such that, for any input $g_j \in U^{(k)}$, $\mathcal{M}(g_j) = (Z_j, h_j)$ with $Z_j \subseteq \bigcup_{k'\neq k} G^{(k')}$, the set of proper components mapped to g_j; $h_j : \prod_{g_i\in Z_j} D_i \to D_j$, the* logical integration function *defining the behaviour of g_j (if $Z_j = \emptyset$, g_j remains free and h_j is the empty function).*

Then the composition of the r LRMs, denoted $\bigotimes_{\mathcal{M}} \left\{N^{(k)}\right\}_{k=1,\dots,r}$, is a LRM $N = (G, U, K)$ with:

- *$G = \bigcup_{k=1,\dots,r} G^{(k)}$, the set of proper components, and $U = \bigcup_{k=1,\dots,r} \left(U^{(k)} \setminus \widetilde{U}^{(k)}\right)$, the set of input components (where $\widetilde{U}^{(k)} = \{g_j \in U^{(k)} \mid Z_j \neq \emptyset\}$ is the set of mapped input components), $C = G \cup U$ and L_C the corresponding set of indices;*
- *$S = \Pi_{i\in L_C} D_i$ the state space;*
- *The logical regulatory functions $(K_i)_{i\in L_G}$ are defined adequately for any proper component $g_i \in G^{(k)}$: $\forall v \in S, \quad K_i(v) = K_i^{(k)}(w)$, $w \in S^{(k)}$, s.t. $\forall g_j \in C^{(k)} \setminus \widetilde{U}^{(k)}$, $w_j = v_j$ and $\forall g_j \in \widetilde{U}^{(k)}, w_j = h_j\left(v_{|Z_j}\right)$.*

Note that nothing in our definition of LRM composition requires the individual modules to be identical; they can be different LRMs provided that they do not overlap (*i.e.* do not share identical proper components). Therefore, even if our applications involve only identical networks, the proposed framework is generic.

2.3 Dynamics of composed modules

We now proceed with the characterisation of the dynamics of composed models since our aim is to check reachability properties over these dynamics.

The composition of STGs can be formally defined and the dynamics of the LRM, composed from r modules, equals the composition of the individual dynamics of the modules (see Theorem 1 below). In other words, to study the dynamics of a composed model, we can either compose the modules and then construct and analyse the (large) STG or analyze the modules dynamics in a compositional manner. Definitions and proof of the theorem are provided in the Supplementary Materials. Here, we give an intuitive introduction to STG composition and include our main result ensuring the compositionality of our approach.

Composing STGs leads to a new STG, where the states are any combinations of states of the original STGs in which the values of the mapped inputs are compatible with the values of the proper components they are mapped to (compatible states; see Def. S3 in the Supplementary Materials). Transitions going out of a composed state involve only proper components and free inputs, if any free input remains after the composition. Hence, considering a particular state s in the STG of one module, the transitions leaving s have their counterparts in the composed dynamics in all states resulting from the composition of s with compatible states of the others STGs. Moreover, any transition going out of s that involves a mapped input is accounted for by (or synchronised with) transitions over proper components involved in the integration

function of this input. Panels D-F in Fig. 1 illustrate the composition of two STGs reachable from two compatible initial states.

The following theorem asserts that the dynamics of the composition of LRMs matches the composition of their individual dynamics. Importantly, this also applies to the composition of STGs reachable from (compatible) initial conditions.

Theorem 1. *Consider* $N = \bigotimes_{\mathcal{M}} \left\{N^{(k)}\right\}_{k=1,\dots r}$ *the LRM defined as the composition of* r *LRMs. The STG* E_N *of* N *is equal to the composition of the* r *STGs:* $E_N = \bigotimes_{\mathcal{M}} \left\{E^{(k)}\right\}_{k=1,\dots,r}$.

So far, we have defined the framework for LRM composition and shown that one can equivalently generate the dynamics of the composed model or generate the individual dynamics and compose them in any order. The next section is devoted to the implementation of this framework. Because compositional analysis has been well studied in the framework of process algebras, and efficient tools have been developed, we recast our original problem in the realm of process algebras.

3 Implementation

To alleviate the combinatorial explosion of the dynamics associated with LRMs, we rely on classical abstraction and minimisation techniques. Here we describe the key features of the implementation. More details can be found in the Supplementary Materials.

3.1 LTS abstraction and minimisation

Process algebra techniques apply to Labelled Transition Systems (LTSs) representing the dynamics to be analysed. Basically, in contrast to STGs where all the information is stored in the states, in a LTS the information is put onto transition labels. These refer to actions performed by the transitions (here, component updates). Converting a STG into a LTS is thus quite direct. The addition of a specific self-loop transition on (weak and strong) stable states (in which proper components are stable) is a technicality that will ensure the preservation of all paths leading to these states. This transition translates into a specific action denoted $\perp$ in the LTS representation.

Abstraction is obtained by defining a set of components as *non-visible*: their modification is not observable. All transitions involving such components, termed non-visible transitions, can simply be labelled by a special action denoted τ. Then, minimisation consists in building a new LTS equivalent, in some sense, to the original LTS. Equivalence is defined with respect to one of the various equivalence relations described in the literature, each of them preserving certain properties of the original LTS [2, 21, 35]. Some of these relations are implemented in publicly available tools (in particular, CADP).

The choice of an equivalence relation depends on the property to be checked. Here, we aim at identifying all the stable states reachable from an initial condition. We thus opt for the *safety equivalence* that elicits the elimination of all τ transitions and redundant paths, while preserving reachability properties.

We define key components as those allowing the distinction between the potential stable states. They are defined as the visible components for the abstraction operation (the remaining components are thus non-visible). After the minimisation step, the states in the reduced LTS that correspond to stable states in the original dynamics are recovered thanks to the $\perp$ action.

From the onset, the dynamics of each module can be abstracted and minimised in terms of the key components *before* the composition. Only visible components and components that are involved in the composition need to be preserved. The main advantage of this approach is that, by successively performing composition followed by abstraction and minimisation, the full dynamics of the composed model are never generated. However, this incremental composition requires some adaptation described in Section 3.2.

The aforementioned abstraction, minimisation and composition operations are performed using the CADP toolbox [14], which provides several tools to produce, transform and analyse LTSs from process specifications.

LRMs behaviours are expressed using the `LOTOS NT` specification language [4], in terms of a *process* with as many associated *gates* as components (input and proper components), and with *state variables* representing

the values of the components. Each gate can issue actions with labels denoting updates of component values. The process evolution is driven by the logical rules that elicit updates of the state variables. When the conditions associated to a component update are met, an action associated to the gate of the component is issued, the corresponding state variable is updated accordingly, and the main process loops back. When, for the current values of the state variables, the logical rules do not permit any further evolution of the proper components, a $\perp$ action is issued.

Then, the reachability of each potential stable state is verified on the minimised LTS, checking whether there is a path from the initial state towards a $\perp$ action and leading to a final value of the key components corresponding to that stable state. In our implemented workflow, this reachability analysis is performed by using the model checker of CADP.

3.2 Incremental composition of LTSs

In this section we discuss the main implementation aspects of the incremental composition of LTSs (see Supplementary Materials and the code documentation for further details). The composition is specified by providing:

1) The LRMs to be composed and a neighbourhood relation;

2) The integration functions of the mapped inputs;

3) The list of key components – having the potential stable states of the composed model (obtained via static analysis), we can specify the minimal set of components that should remain visible;

4) The (global) initial state – which indicates the initial value of the components in each individual LRM.

The integration of the regulatory signals originating from neighbouring modules is specified by the integration function associated with each mapped input. The values of these functions are partially constrained in the course of an incremental composition, as the values of each argument become bound to an actual proper component – the corresponding actions are to be synchronised. This is why we propose to model integration functions as independent processes with their own dynamics. More precisely, a LTS is generated for each integration function h_i (associated to an input component g_i). Actions in this LTS reflect the updates of both the function arguments and the function value. The values of the function arguments are allowed to vary freely. More precisely, from a given state, if the update of a proper component g_k influencing the value of g_i (*i*.e. $g_k \in Z_i$) has no impact on the value of the integration function h_i, the LTS contains an action corresponding solely to the update of g_k. If, on the other hand, this update does change the value of h_i, the action refers to both g_i and g_k updates. In subsequent composition operations, all these actions must be synchronised with the appropriate actions in the LTSs accounting for the evolution of the relevant components (the arguments of h_i), and with the actions over g_i.

Most importantly, to ensure a correct synchronisation, actions over mapped input components and arguments of integration functions must be kept visible during the incremental composition until they are no longer required.

Reduced LTSs of individual modules are incrementally composed following the specified rules. At each step, components that are no longer needed are made non-visible (new abstraction step) and the LTS of the intermediate composition is minimised.

Upon the last composition step, a final abstraction and minimisation step is undertaken, where only key components are kept visible thus obtaining a minimal description of the dynamics of the whole composition.

Given the `LOTOS NT` specifications of the module and of the integration functions, the synchronisation (composition) of the whole is specified by way of synchronisation vectors [18]. A high-level language provided by CADP called `SVL` [13] is used to specify the generation of the LTSs for each individual LRM, their preliminary minimisation step, subsequent synchronisation steps, as well as the final abstraction and minimisation operations. The intermediate composition steps are automatically produced by CADP. The `SVL` script as well as the `EXP` [18] file, which specifies the synchronisation vectors, are automatically generated from a symbolic representation of the LRM exported from GINsim and processed using Perl scripts.

Note that, for simplicity, our current implementation performs composition for multi-cellular systems such as the ones presented in Section 4: modules are identical (r instances of a unique LRM), the neighbourhood relation is defined as a $r \times r$ adjacency matrix, and the (same) inputs in distinct modules are uniformly mapped to proper components from neighbouring modules. Hence, it is enough to specify the integration

functions of the mapped inputs, with the proper components that are to be taken as arguments.

3.3 Note on composition order

The order followed to compose the modules affects the performance of our method. Our implementation relies on the *smart reduction* [8], which is an operator of `SVL` that determines the order of the composition, including the synchronisation, abstraction and minimisation operations. The underlying heuristics aims at controlling the size of the intermediate LTSs. It generally performs poorly in the case of LRM composition, because it preferentially synchronises all LTSs related to the integration functions (which are usually smaller), then composes the LTSs associated to the modules and finally combines all these intermediate LTSs. As a consequence, all the restrictions on the dynamics of a module (imposed by its neighbours via the integration functions) are put in place later rather than sooner, giving rise to large intermediate LTSs. We further discuss and illustrate this issue of the composition order in Section 4.1.

4 Applications

To evaluate our method, we consider the Segment-Polarity and the Delta-Notch modules (various compositions of four instances of the toy LRM introduced in Fig. 1 are presented in the Supplementary Materials).

4.1 The Segment-Polarity module

The Segment-Polarity module (SP) is involved in the fruit fly embryo segmentation, which has been extensively studied by geneticists as a model system for development. Early embryo organisation into a series of segments along the antero-posterior axis is initiated by maternal morphogens, which control a few dozens of genes. These genes have been split into several classes. The first classes, *gap*, *pair-rule* and *segment-polarity* modules, constitute a temporal hierarchical genetic system. Segment-polarity genes are under the control of the pair-rule genes. Their patterns of expression define the anterior and posterior parts of the embryonic segments and they are responsible for the consolidation of these borders [28]. The segment-polarity (SP) module has been modelled using continuous [17] and logical approaches [6, 29]. Here, we rely on the model defined in [29], with an intracellular network of a dozen of components, submitted to two external inputs (the Wingless (Wg) and Hedgehog (Hh) signals).

We compose two modules, accounting for the cells flanking the segmental border. Figure 2 illustrates this model and the results contrasting the STG size with the minimised LTS size for a full version of the model and a reduced one. The initial condition accounts for the outcome of the activity of the pair-rule system [28, 29]: significant amounts of Wg and Slp in the anterior cell, a significant amount of En in the posterior cell. The three stable states reachable from this initial condition combine three cellular patterns: a Wg expressing state (denoted W), an En expressing state (E), and a *trivial* state (T) with neither Wg, nor En. They correspond to the expected wild type pattern WE in addition to the TT and EW patterns [29]. We also consider a reduced (intra-cellular) regulatory graph with 9 components and 31 regulatory interactions. It has been obtained by applying the reduction method available in GINsim and described in [24]. Note that with GINsim, it was impossible to construct the STG for the full model, and for the reduced model the resulting LTS structure is much smaller and thus more amenable to further analysis (see Fig. 2-C-D). In [29], the construction of the STG was interrupted as soon as the WT and TT stable states were reached.

To investigate the impact of the composition order, we now consider six instances of a further reduced SP module (with 3 proper components: Wg, En, Hh). These modules are organised along a line, each having two neighbours, except for the two extreme ones. For each instance i ($i = 1, \ldots, 6$), our method generates three LTSs: 1) M_i, LTS of the i-th module; 2) $H_i(\text{Hh})$, LTS of the integration function of the input Hh in the i-th module; 3) $H_i(\text{Wg})$, LTS of the integration function of the input Wg in the i-th module.

Table 1 illustrates the impact of the composition order. It first includes the composition steps as performed by the smart reduction implemented in CADP. The intermediate LTSs grow very fast. Indeed, we can observe that composition operations are performed over LTSs that have no synchronisation restrictions (because they are not related, such as, e.g. $H_2(\text{Wg})$ and $H_5(\text{Wg})$). In this specific case, the smart reduction heuristics performs poorly. Even the monolithic composition of all the LTSs (in a single step) leads to a much

better performance. A better composition order, drawn from the knowledge of how each LTS restricts its neighbours, consists in first composing the LTSs of each regulatory module with the LTSs of the integration functions of their input components, and then to iteratively compose these LTSs along the line of the six modules (see Table 1).

4.2 The Delta-Notch module

The Delta-Notch module is involved in cell differentiation in crucial steps of embryonic development of several species [16,19]. In each cell, when active, the Notch protein inhibits the production of Delta. The production of Notch is stimulated by the presence of Delta in neighbouring cells. These regulatory interactions, for a single cell, can be represented by the logical module shown in Fig. 3, where the Delta_ext component accounts for Delta in the adjacent cells. This simplified model is the Boolean counterpart of the model used as a case study for the compositional verification approach described in [16]. Similarly to [16], cells are hexagonal (*i.e.* they can have up to 6 neighbours), and the integration function of the input component Delta_ext is a logical `OR` between all the Delta components in neighbouring modules. We illustrate our approach by considering

Composition	**Resulting LTS size**	**Minimised LTS size**
Incremental composition, smart reduction		
$\mathcal{L}_1 = H_3(\mathrm{Wg}) \otimes H_5(\mathrm{Wg})$	27	27
$\mathcal{L}_2 = H_2(\mathrm{Wg}) \otimes H_4(\mathrm{Wg})$	27	27
$\mathcal{L}_3 = H_3(\mathrm{Hh}) \otimes H_5(\mathrm{Hh})$	27	27
$\mathcal{L}_4 = H_2(\mathrm{Hh}) \otimes H_4(\mathrm{Hh})$	27	27
$\mathcal{L}_5 = M_1 \otimes M_2$	729	729
$\mathcal{L}_6 = M_5 \otimes M_6$	747	747
$\mathcal{L}_7 = \mathcal{L}_2 \otimes H_6(\mathrm{Wg})$	27	27
$\mathcal{L}_8 = \mathcal{L}_1 \otimes H_1(\mathrm{Wg})$	27	27
$\mathcal{L}_9 = \mathcal{L}_4 \otimes H_6(\mathrm{Hh})$	27	27
$\mathcal{L}_{10} = \mathcal{L}_3 \otimes H_1(\mathrm{Hh})$	27	27
$\mathcal{L}_{11} = M_3 \otimes M_4$	8 019	8 019
$\mathcal{L}_{12} = \mathcal{L}_{11} \otimes \mathcal{L}_9 \otimes \mathcal{L}_5$	1 764 450	1 130 157
$\mathcal{L}_{13} = \mathcal{L}_{12} \otimes \mathcal{L}_{10} \otimes \mathcal{L}_6$	25 999 469	*out of memory*
Monolithic composition, in one step		
$\bigotimes_{i=1,\ldots,6} (M_i \otimes H_i(\mathrm{Hh}) \otimes H_i(\mathrm{Wg}))$	548 208	864
Incremental composition, specific order		
$C_1 = M_1 \otimes H_1(\mathrm{Hh}) \otimes H_1(\mathrm{Wg})$	81	81
$C_2 = M_2 \otimes H_2(\mathrm{Hh}) \otimes H_2(\mathrm{Wg})$	729	729
$C_3 = M_3 \otimes H_3(\mathrm{Hh}) \otimes H_3(\mathrm{Wg})$	729	729
$C_4 = M_4 \otimes H_4(\mathrm{Hh}) \otimes H_4(\mathrm{Wg})$	891	891
$C_5 = M_5 \otimes H_5(\mathrm{Hh}) \otimes H_5(\mathrm{Wg})$	891	891
$C_6 = M_6 \otimes H_6(\mathrm{Hh}) \otimes H_6(\mathrm{Wg})$	81	81
$\mathcal{L}_1 = C_1 \otimes C_2$	729	585
$\mathcal{L}_2 = \mathcal{L}_1 \otimes C_3$	5 265	2 691
$\mathcal{L}_3 = \mathcal{L}_2 \otimes C_4$	27 027	7 101
$\mathcal{L}_4 = \mathcal{L}_3 \otimes C_5$	75 852	32 409
$\mathcal{L}_5 = \mathcal{L}_4 \otimes C_6$	19 829	864

Table 1: Impact of the composition order illustrated with 6 instances of a reduced version of the SP module (3 proper components), interconnected along a line; the 1st and the 6th modules have one neighbour (the 2nd and the 5th), all other modules have two neighbours. The left column indicates the LTSs that are synchronised, each step defining a new LTS; the middle column contains the size of the resulting LTS, while the right column gives the size of the minimised LTS. First, the results relate to the order performed by the smart reduction available in CADP. Then, the composition is performed in one step and, finally, incremental composition is performed following an order based on the structure of the model (see text).

three compositions: DN7, DN10 and DN12 depicted in Fig. 3.

In Gössler's paper, the reachability of a *specific* stable state from an initial condition was checked in models encompassing up to 343 modules. In contrast, we could only deal with models encompassing up to a dozen of modules. However, we answer a different problem, since our procedure accounts for all the trajectories from a given initial condition to *any* reachable stable state, which is certainly different from verifying the existence of one trajectory leading to one specific stable state (note that the number of stable states grows rapidly with the number of interconnected modules and depends on the neighbouring relations).

An isolated, open module has two stable states: when Delta_ext is active, Notch eventually becomes active and Delta inactive, otherwise Notch remains inactive and Delta becomes active. Either stable state can be univocally identified by the value of *e.g.* Delta, hence there is a single key component in each module.

In the case of DN7 as depicted in Fig. 3, if all modules start with both Delta and Notch inactive, then the 6 stable expression patterns are reachable. The pattern of Delta expression that ultimately emerges depends on which cells express Delta sooner. The STG generated with GINsim shows that from this initial condition, almost the entire state space is explored (there are 16 024 reachable states for a total state space of size 16 384). The minimised LTS (retaining only Delta as a visible component) contains 2 290 states, which is a significant reduction (it is then possible to check on this reduced LTS that the 6 stable expression patterns are reachable). The reduction is even more significant in the case of DN10. For DN12, only the compositional framework we propose here can solve the problem.

5 Conclusions

Pattern formation notably relies on inter-cellular communication, while involving intra-cellular regulatory processes. Several logical models dealing with such developmental processes have been published [1,6,15,29]. However, current monolithic approaches are not appropriate to answer questions that require searching the (generally huge) state space. In particular, the study of inter-cellular networks involved in differentiation processes focuses on the reachability of stable expression patterns from given initial conditions. Here, we have introduced a framework to address the combinatorial explosion of logical models that can be specified as module compositions. Our approach is made operational by recasting Logical Regulatory Module composition in terms of process algebra operations. We rely on GINsim for the definition of the LRMs and on CADP for the composition, abstraction and minimisation operations. The procedure has been applied to the Segment-Polarity and the Delta-Notch modules, showing that huge reductions can be obtained.

With the study of the SP module, we could discuss the problem of the composition order. At present, the public release of CADP does not provide instructions to make smart reduction adopt an order different from what the built-in heuristics determines, although the appropriate specification of the several intermediate synchronisation steps can be manually done by a proficient CADP user. In general, it is not trivial to determine the optimal composition order. However, considering the application we are dealing with, we propose an alternative heuristics that should lead to reasonable performances. It consists in composing the LTSs of individual LRMs with the integration functions of their inputs and progressively proceed according to the specified neighbourhood relation.

Our results indicate that, while our compositional framework leads to a significant decrease of the size of the behaviour, there is still room for improvement. In the case of the reduced SP model, the number of states in the dynamics is reduced by two orders of magnitude. Nevertheless, the asynchronous dynamics are often huge, since they include all possible behaviours from a given initial condition, resulting in dynamics that explore a large portion of the state space. Most of the trajectories represented in these dynamics are generally not pertinent from the biological point of view. Several methods aim to avoid non-realistic trajectories in asynchronous dynamics, among them the consideration of priority classes [11]. Future work will extend LRM composition rules, accounting for such modified dynamics.

In this paper, we have used minimisation modulo safety equivalence, which provides a good compromise between algorithmic complexity and compression of the state space. Most importantly, the safety equivalence preserves the reachability of stable states, a crucial property when studying differentiation processes. This property could be enriched to verify additional features along the trajectories leading to the states of interest, such as whether a given component is always required to change. Other verification techniques could prove efficient in our context, including the consideration of on-the-fly verification directed by a property to be checked [18,20].

The compositional framework presented here constitutes a novel and systematic method to compose logical regulatory modules and to efficiently perform comprehensive analyses of their behaviours. This will greatly facilitate the definition and analysis of network models involved in various multicellular patterning systems.

Acknowledgements

We are most grateful to H. Klaudel and F. Pommereau for insightful discussions. This work was supported by the research grants [ANR-08-SYSC-003] and [ANR-10-COSINUS-007] from the Agence Nationale de la Recherche and by [PTDC/EIA-CCO/099229/2008] from the Fundação para a Ciência e a Tecnologia.

References

[1] Eugenio Azpeitia, Mariana Benitez, Iliusi Vega, Carlos Villarreal, and Elena Alvarez-Buylla. Single-cell and coupled GRN models of cell patterning in the Arabidopsis thaliana root stem cell niche. *BMC Systems Biology*, 4(1):134, 2010.

[2] Ahmed Bouajjani, Jean-Claude Fernandez, Susanne Graf, Carlos Rodriguez, and Joseph Sifakis. Safety for branching time semantics. In *ICALP*, volume 510 of *LNCS*, pages 76–92. Springer, 1991.

[3] Laurence Calzone, Laurent Tournier, Simon Fourquet, Denis Thieffry, Boris Zhivotovsky, Emmanuel Barillot, and Andrei Zinovyev. Mathematical modelling of cell-fate decision in response to death receptor engagement. *PLoS Comput Biol*, 6(3):e1000702, 2010.

[4] David Champelovier, Xavier Clerc, Hubert Garavel, Yves Guerte, Frédéric Lang, Christine McKinty, Vincent Powazny, Wendelin Serwe, and Gideon Smeding. Reference manual of the LOTOS NT to LOTOS translator (version 5.5). Technical report, INRIA, 2011.

[5] C. Chaouiya, H. Klaudel, and F. Pommereau. A modular, qualitative modeling of regulatory networks using Petri nets. In *Modeling in Systems Biology, The Petri Net Approach*, volume 16, chapter 12, pages 253–279. Springer, 2011.

[6] M. Chaves, R. Albert, and E. D. Sontag. Robustness and fragility of Boolean models for genetic regulatory networks. *J. Theor. Biol.*, 235(3):431–49, 2005.

[7] Federica Ciocchetta and Jane Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theor. Comput. Sci.*, 410(33-34):3065–3084, August 2009.

[8] Pepijn Crouzen and Frédéric Lang. Smart reduction. In *Proceedings of FASE'2011 (Saarbrücken, Germany)*, volume 6603 of *LNCS*, pages 111–126, Mar 2011.

[9] Hidde de Jong and Michel Page. Search for steady states of piecewise-linear differential equation models of genetic regulatory networks. *IEEE/ACM Trans Comput Biol Bioinform*, 5(2):208–22, 2008.

[10] Elena Dubrova and Maxim Teslenko. A sat-based algorithm for finding attractors in synchronous boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(5):1393–1399, 2011.

[11] A. Fauré, A. Naldi, C. Chaouiya, and D. Thieffry. Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):124–131, 2006.

[12] A. Fauré, A. Naldi, F. Lopez, C. Chaouiya, A. Ciliberto, and D. Thieffry. Modular logical modelling of the budding yeast cell cycle. *Mol Biosyst*, 5(12):1787–1796, 2009.

[13] Hubert Garavel and Frédéric Lang. SVL: a scripting language for compositional verification. In *Proc. of IFIP WG 6.1, FORTE'2001*, pages 377–92. Kluwer Academic Publishers, 2001.

[14] Hubert Garavel, Frédéric Lang, Radu Mateescu, and Wendelin Serwe. CADP 2010: A toolbox for the construction and analysis of distributed processes. In *Proc. of TACAS'2011*, volume 6605 of *LNCS*, pages 372–87, Mar 2011.

[15] A. González, C. Chaouiya, and D. Thieffry. Logical modelling of the role of the Hh pathway in the patterning of the Drosophila wing disc. *Bioinformatics*, 24:i234–40, 2008.

[16] Gregor Gössler. Component-based modeling and reachability analysis of genetic networks. *IEEE/ACM Trans Comput Biol Bioinform*, 8(3):672–82, 2011.

[17] N. T. Ingolia. Topology and robustness in the Drosophila segment polarity network. *PLoS Biol.*, 2(6):e123, 2004.

[18] Frédéric Lang. EXP.OPEN 2.0: A flexible tool integrating partial order, compositional, and on-the-fly verification methods. In *Proc. of IFM'2005*, volume 3771 of *LNCS*, pages 70–88, 2005.

[19] G Marnellos, G A Deblandre, E Mjolsness, and C Kintner. Delta-Notch lateral inhibitory patterning in the emergence of ciliated cells in Xenopus: experimental observations and a gene network model. *Pac Symp Biocomput*, pages 329–340, 2000.

[20] Radu Mateescu and Damien Thivolle. A model checking language for concurrent value-passing systems. In Jorge Cuellar, Tom Maibaum, and Kaisa Sere, editors, *Proc. of the 15th Int Symp on Formal Methods FM'08*, number 5014 in LNCS, pages 148–64, 2008.

[21] Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[22] A Naldi, PT. Monteiro, and C. Chaouiya. Efficient handling of large signalling-regulatory networks by focusing on their core control. In *CMSB'12*, volume 7605 of *LNCS*, pages 288–306, 2012.

[23] A. Naldi, D. Thieffry, and C. Chaouiya. Decision diagrams for the representation of logical models of regulatory networks. In *CMSB'07*, volume 4695 of *Lecture Notes in Bioinformatics (LNBI)*, pages 233–247, 2007.

[24] Aurélien Naldi, Jorge Carneiro, Claudine Chaouiya, and Denis Thieffry. Diversity and plasticity of Th cell types predicted from regulatory network modelling. *PLoS Comput Biol*, 6(9):e1000912, 2010.

[25] Aurélien Naldi, Elisabeth Remy, Denis Thieffry, and Claudine Chaouiya. Dynamically consistent reduction of logical regulatory graphs. *Theor. Comput. Sci.*, 412:2207–18, 2011.

[26] R. Randhawa, C. A Shaffer, and JJ. Tyson. Model composition for macromolecular regulatory networks. *IEEE/ACM Trans Comput Biol Bioinform*, 7(2):278–87, 2010.

[27] J Saez-Rodriguez, L G Alexopoulos, J Epperlein, R Samaga, D A Lauffenburger, S Klamt, and P K Sorger. Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Mol Syst Biol*, 5:331, 2009.

[28] L. Sánchez, C. Chaouiya, and D. Thieffry. From gradients to segments: a logical analysis of the genetic network controlling early Drosophila. In *Bioinformatics of Genome Regulation and Structure II*, pages 379–90. Springer-Kluwer, 2006.

[29] L. Sánchez, C. Chaouiya, and D. Thieffry. Segmenting the fly embryo: a logical analysis of the segment polarity cross-regulatory module. *Int. J. Dev. Biol.*, 52(8):1059–75, 2008.

[30] Marvin Schulz, Jannis Uhlendorf, Edda Klipp, and Wolfram Liebermeister. SBMLmerge, a system for combining biochemical network models. *Genome informatics International Conference on Genome Informatics*, 17(1):62–71, 2006.

[31] Jacky L Snoep, Frank Bruggeman, Brett G Olivier, and Hans V Westerhoff. Towards building the silicon cell: a modular approach. *Bio Systems*, 83(2-3):207–216, February 2006.

[32] Jörg Stelling, Pedro Mendes, Frank Tonin, Edda Klipp, Riccardo Zecchina, Matthias Heinemann, Natasa Przulj, Judith Wodke, Szymon Stoma, Hand-Michael Kaltenbach, Joachim Almquist, Andreas Raue, Jonas Hagmar, Marcus Krantz, Andrea Pagnani, Sven Nelander, Marija Cvijovic, and Mats Jirstrand. Defining modeling strategies for Systems Biology. Technical report, FutureSysBio Workshop, 2011. Report for the European Union Commission.

[33] R. Thomas. Regulatory networks seen as asynchronous automata: A logical description. *J. Theor. Biol.*, 153:1–23, 1991.

[34] R. Thomas and R. D'Ari. *Biological Feedback*. CRC Press, 1990.

[35] R. J. van Glabeek and W. P. Weijland. Branching-time and abstraction in bisimulation semantics (extended abstract). CS R8911, cwi, Amsterdam, 1989.

[36] GP. Wagner, Mihaela M. Pavlicev, and J M. Cheverud. The road to modularity. *Nat Rev Genet*, 8(12):921–31, 2007.

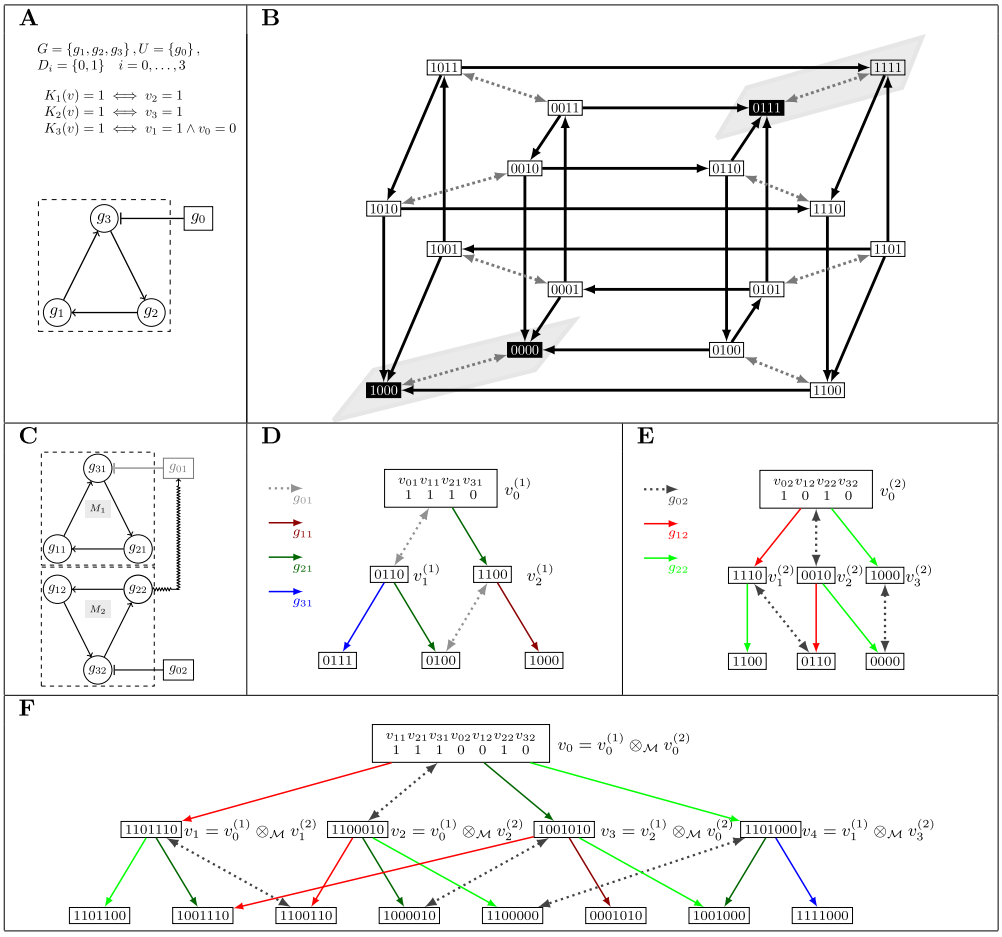
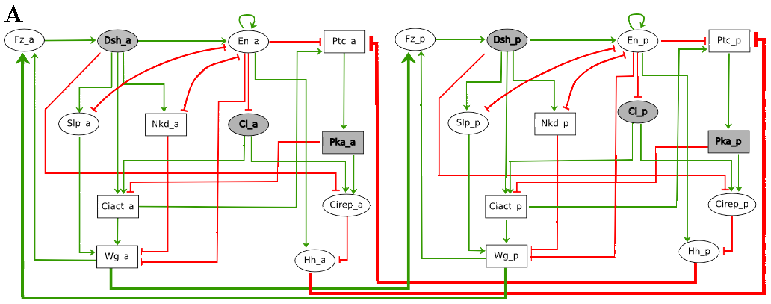


Figure 1: **(A)** A simple, toy Logical Regulatory Module, where each proper component has one activator, g_3 being further inhibited by the input component g_0. **(B)** The corresponding STG; each node is a state (v_0, v_1, v_2, v_3); dotted arrows depict input transitions over g_0; plain arrows depict proper transitions; black nodes denote states in which proper components are stable (the sole outgoing transition refers to the input). The pair of states defined by $v_1 = v_2 = v_3 = 0$ (respectively $v_1 = v_2 = v_3 = 1$) define a strong (respectively weak) stable state with respect to the proper components. **(C-F)** Composition of two instances of this LRM. A relabelling of the components ensures their uniqueness: component g_i of module M_j (j-th instance of the LRM) becomes g_{ij}. In panel **C**, the input g_{01} of M_1 is mapped to the proper component g_{22} of module M_2: $\mathcal{M}(u_{01}) = (Z_{01}, h_{01})$, with $Z_{01} = \{g_{22}\}$ and h_{01} the identity. For the sake of brevity, the dynamics are truncated at depth 2 from the initial states. Panel **D** (respectively **E**) displays the (truncated) STG of M_1 (respectively M_2) from the initial state $v_0^{(1)} = (1, 1, 1, 0)$ (respectively $v_0^{(2)} = (1, 0, 1, 0)$). These states are compatible: $v_{01} = v_{22} = h_{01}(v_{22}) = 1$. Panel **F** displays the composed STG: transitions over g_{01} are lost (g_{01} is a mapped input), while transitions over g_{02} are preserved. The (red) transition that sets g_{12} to 1 from the initial state $v_0^{(2)}$ in panel **E** gives rise to 2 transitions in panel **F**, the first leaving the initial state v_0, the second leaving v_3. Likewise, the (dark green) transition that sets g_{21} to 0 from the initial state $v_0^{(1)}$ has 4 counterparts: (dark green) transitions from v_0, v_1, v_2 and v_4.



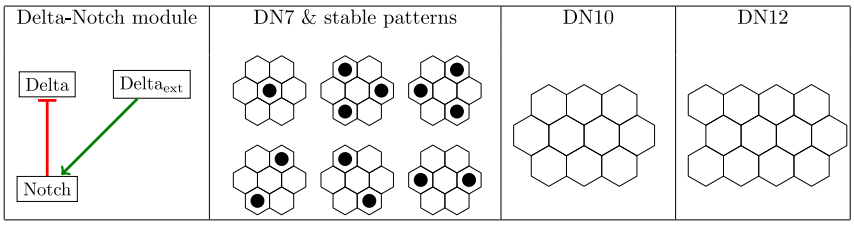
B Full model (12 components per cell)

	STG (GINsim)	LTS (CADP)
time	6.62 days	6.69 hours
size (states)		9 701
memory	out of memory (>50 GB)	27.66 GB

C Reduced model (9 components per cell)

	STG (GINsim)	LTS (CADP)
time	14.72 s	85 s
size (states)	160 580	1 645
memory	2.63 GB	0.78 GB

Figure 2: **(A)** Composition of two instances of the segment-polarity module, accounting for the two cells flanking the segmental border. Each module contains 12 components among which 7 are associated to Boolean variables (oval nodes) and 5 to ternary variables (rectangle nodes); suffix 'a' (respectively 'p') denotes components of the cell anterior (respectively posterior) to the border; regulatory interactions are denoted by arrows (activation) or flat-end edges (repression); inter-cellular interactions are denoted by thick edges. Additionally, grey nodes are reduced to obtain the 9-components version of the model. **(B–C)** Measures for the construction of the dynamics, in terms of size (number of states), time and memory use, for the full (panel B) and reduced (panel C) models.



DN7 topology

	STG (GINsim)	LTS (CADP)
time	< 1 s	2.25 minutes
size	16 024	2 290
memory	7.36 MB	2.46 GB

DN10 topology

	STG (GINsim)	LTS (CADP)
time	3.82 minutes	6.52 minutes
size	1 022 890	63 179
memory	555 MB	18.98 GB

DN12 topology

	STG (GINsim)	LTS (CADP)
time	–	1.54 hours
size	–	523 611
memory	out of memory	45.69 GB

Figure 3: Composition of the Delta-Notch module. Top left panel displays the Delta-Notch Logical Regulatory Module. All components are Boolean. Delta and Notch are proper components and Delta_ext is an input component integrating the influence of Delta from neighbouring modules (the integration function being a disjunction). In the initial state, all variables are set to 0. Three cases are analysed: DN7 referring to the composition of 7 instances (cells) of the module, following the neighbouring relations as illustrated (the cell at the centre receives a Delta signal from its 6 neighbouring cells, all the other cells have 3 neighbours), DN10 and DN12. For DN7 there are 6 (reachable) stable expression patterns as depicted (a black dot indicates a cell with active Delta); DN10 gives rise to 14 stable expression patterns and D12 to 22 stable expression patterns, all being reachable from the specified initial state. Performance results are given for the three cases.