Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 335 (2018) 67–90

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Unfolding of Parametric Boolean Networks

Juraj Kolčák[a,b] David Šafránek[b,1] Stefan Haar[a] Loïc Paulevé[c]

[a] *Inria and LSV*
École Normale Supérieure de Cachan and CNRS
Cachan, France

[b] *Systems Biology Laboratory (Sybila)*
Masaryk University
Brno, Czech Republic

[c] *LRI UMR CNRS 8623*
Univ. Paris-Sud – CNRS, Université Paris-Saclay
Orsay, France

Abstract

In systems biology, models of cellular regulatory processes such as gene regulatory networks or signalling pathways are crucial to understanding the behaviour of living cells. Available biological data are however often insufficient for full model specification. In this paper, we focus on partially specified models where the missing information is abstracted in the form of parameters. We introduce a novel approach to analysis of parametric logical regulatory networks addressing both sources of combinatoric explosion native to the model. First, we introduce a new compact representation of admissible parameters using Boolean lattices. Then, we define the unfolding of parametric Boolean networks. The resulting structure provides a partial-order reduction of concurrent transitions, and factorises the common transitions among the concrete models. A comparison is performed against state-of-the-art approaches to parametric model analysis.

Keywords: Boolean networks, parameters identification, asynchronous systems, concurrency, systems biology

1 Introduction

One of the main problems studied in computational systems biology is understanding of intracellular molecular interactions, often represented as networks. Two particular classes of processes are predominantly modelled, gene expression regulation (gene regulatory networks) and cell signalling [13].

The prime interests of gene regulatory networks are gene-protein and gene-gene interactions, the latter are generally facilitated by the proteins they encode. Cell signalling models usually consist of one or several signalling pathways. In simple terms chains of proteins provide information flow by means of sequential phosphorylation until some cellular process (such as gene expression) is influenced.

[1] This work has been partially supported by the Czech Science Foundation grant No. GA15-11089S.

https://doi.org/10.1016/j.entcs.2018.03.009

Although both of the described processes are quantitative in their nature, it is often the case that precise kinetic parameters of the reactions are unknown in biological context. As such, it is common to model genetic regulatory networks and signalling pathways by discrete models (logical regulatory networks) [1,6,15,18,19].

In the context of gene regulatory networks and signalling pathways it is often the case that one-to-one influences between species are known from *in vitro* experiments. The results of combinations of those influences are, however, largely unknown. In other words, it may be known that two species have both positive influence on the activity/population of a third species. However, it is rarely known if both of the activators must be present to activate the target or if just one is sufficient. In general, an arbitrary logical function may govern the joint influences. To cope with the problem technically, the individual target values of a species in possible combinations of their regulators activity are considered as unknown *parameters*.

The analysis of *parametric regulatory networks* (PRNs) is hindered by dual combinatorial explosion. Not only is the state space exponential in size of the networks, but the number of parametrisations is in the worst case doubly exponential in number of species. Combination of those factors often leads to the fact that analysis techniques of PRNs do not scale to larger networks.

Our Contribution. We introduce a new analysis framework for parametric logical regulatory networks addressing combinatorial explosion on two levels. First, we propose *a novel encoding of parametrisations using the inner structure of parametrisations*. The encoding is applied to mitigate the combinatorial explosion induced by parametrisations. Accompanying methods are provided allowing for efficient use of the encoding. Second, we *extend Petri net unfoldings to accommodate for the parametric setting*. The unfoldings are coupled with the encoding method for parametrisations to allow for compact representation of state space of the PRNs thanks to their ability to exploit concurrency. Finally, a prototype implementation is provided to compute the introduced unfoldings. Experiments are conducted comparing the results of our methods against state-of-the-art methods in parametric regulatory network analysis.

Related Work. The analysis of logical regulatory networks under parameter uncertainty is a field not yet largely explored. Recently, it is gaining popularity thanks to the importance and great promise to the field of systems biology. Computational Tree Logic (CTL) [2] has been used to enumerate all possible temporal properties (parametrisations) of Thomas networks, by Bernot et al. [4]. Methods based on LTL model checking [2] have also been introduced for Thomas networks [14,10]. In [14] the method called coloured model checking first introduced in [3] is used to capitalise on many parametrisations sharing some parts of their behaviour. The parametrisations are represented by colours (bits) in a binary vector and the model checking is extended to binary vector operations to keep track of the satisfying behaviours.

The approach in [10] explores the state space represented symbolically in form of execution trees. This approach is closest to our work since the symbolical representation of state space employed in [10] is acyclic, similar to unfoldings. Furthermore,

encoding of parametrisations is also performed in [10] using logical formulas. Contrary to our fixed-size encoding, however, the formula used in [10] continues to expand during the exploration as more detailed encoding of parametrisations is required.

Work was also done using constraint logic programming for parameter identification [5,9], again using Thomas networks. The approach in [5] encodes all available biological knowledge into logical constraints on the behaviour of the network while in [9] the constraint logic programming is used to pre-process the initial set of behaviours to filter out those in conflict with the constraints. Model checking is used on the smaller (filtered) set afterwards.

Ostrowski et al. [17] also introduce a method for restricting the initial set of possible behaviours for Boolean networks. Logical constraints are derived from time series data and answer set programming (ASP) is applied to compute a set of transient dynamics (parametrisations) best fitted to the measurements.

Paper Structure. In Section 2 we introduce the parametric regulatory networks including their semantics and parametrisation. Section 3 further expands the model by labels on the influences used to incorporate prior knowledge into the model. In Section 4 we address the potentially double exponential number of parametrisations by introducing a new encoding of parametrisations. This encoding is subsequently applied for unfoldings of parametric regulatory networks in Section 5. Section 6 features experimental results using the parametric unfoldings and comparison against methods relying on execution trees [10] is provided.

2 Parametric Regulatory Networks

In this section, we introduce *parametric regulatory networks* (PRN). Informally one can consider PRN as a standard regulatory network with unknown dynamics, namely transition relation. We can therefore capture the topology of a PRN using a directed graph, so-called Influence Graph, $G = (V, I)$ where V is the set of n vertices (components) and $I \subseteq V \times V$ is the set of directed edges (influences). We denote the set of all in-neighbours (regulators) of some $v \in V$ as $n^-(v) = \{u \in V \mid (u, v) \in I\}$ and the set of all out-neighbours (targets) as $n^+(v) = \{u \in V \mid (v, u) \in I\}$. The influence graph of our running example can be seen in Figure 1.

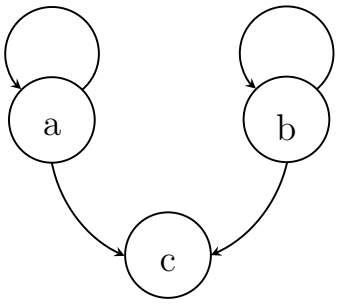

Figure 1. The influence graph of a simple three node regulatory network. We will further use this influence graph for our running example.

a	a
0	$P^a_\emptyset$
1	$P^a_{\{a\}}$

b	b
0	$P^b_\emptyset$
1	$P^b_{\{b\}}$

a	b	c
0	0	$P^c_\emptyset$
0	1	$P^c_{\{b\}}$
1	0	$P^c_{\{a\}}$
1	1	$P^c_{\{a,b\}}$

Table 1
The truth tables for the nodes of the running example influence graph depicted in Figure 1. Truth tables for all three nodes: a,b and c are listed left to right.

Generally, every component $v \in V$ is considered as a variable with a finite discrete domain (multivalued). In the scope of this article we limit ourselves to Boolean settings. Extension to multivalued variables is considered as further work. In the case that every variable $v \in V$ is Boolean we denote the PRN as a *parametric Boolean network* (PBN).

Viewing the components of the interaction graph as variables allows for a natural definition of a state of the PBN. By a state X of $G = (V, I)$ we mean any subset of V ($X \subseteq V$). We say that any component $v \in V$ is *active* (has value 1) in state X if $v \in X$ and, respectively, v is *inactive* (has value 0) if $v \notin X$. We denote the set of all possible states as $\mathcal{X} = 2^V$.

The nature of the interactions depends on the activity levels of the components in a given state. However, it is often the case in biology that exact effects of regulators on their targets are unknown. We therefore abstract these values by means of parameters.

A *parameter* represents a value of the target assigned to a given combination of active and inactive regulators as determined in the particular state. Naturally, there exists a parameter for any such combination of regulators. We denote such a combination as *regulatory context* (RC). Formal definition follows.

Definition 2.1 A *regulatory context* ω of component $v \in V$ is an arbitrary subset of the regulators of v. Formally, $\omega \subseteq n^-(v)$. Just as with the states of PBN we say that all components $u \in \omega$ are active and all components $u \in n^-(v) \setminus \omega$ are inactive.

The set of all combinatorially possible regulatory contexts of v will be further denoted as $\Omega_v = 2^{n^-(v)}$.

In that way, RCs correspond to the rows of the truth table for each component. The truth tables with the RCs and parameters for our running example can be seen in Table 1.

Every parameter (RC) can be assigned a target value 0 or 1. We denote such an assignment for all RCs as *parametrisation*.

Definition 2.2 A *parametrisation* $P \subseteq \Omega$, with $\Omega = \bigcup_{v \in V} (\{v\} \times \Omega_v)$, associates regulatory contexts to each component. We say that the target value of RC ω of component v under parametrisation P is 1 iff $(v, \omega) \in P$. We write $\omega \in P$ instead of $(v, \omega) \in P$ whenever the target v is known from the context.

We denote the set of all possible parametrisations of an influence graph $G = (V, I)$ as $\mathcal{P}_G = 2^{\Omega}$.

A PBN $\mathcal{B}$ is thus an influence graph G equipped with possible parametrisations:

Definition 2.3 A *Parametric Boolean Network (PBN)* $\mathcal{B}$ is a couple $(G, \mathcal{P})$ where G is an influence graph, and $\mathcal{P} \subseteq \mathcal{P}_G$ with $\mathcal{P} \neq \emptyset$ is a non-empty set of possible parametrisations of G.

If $\mathcal{P} = \mathcal{P}_G$, then we say that $\mathcal{B}$ is fully parametric. On the other hand, if $|\mathcal{P}| = 1$, then $\mathcal{B}$ is a simple Boolean network.

Finally, we can define the dynamics of the PBN. As we already mentioned, the dynamics of a PBN equipped with a single parametrisation is identical to standard Boolean networks. There are, however, several ways to define dynamics of a Boolean network from the synchrony perspective. In the biological setting the individual reactions are often temporally independent from each other and no explicit synchrony exists. In the scope of this paper, we consider the usual asynchronous semantics of Boolean networks, where at most one component gets updated at a time.

The asynchronous dynamics are generally non-deterministic, however, it can be easily captured by means of the so-called *state transition graph* (STG) $S = (\mathcal{X}, \delta)$ where $\delta \subseteq \mathcal{X} \times \mathcal{X}$ is the state transition relation given by target values of RCs. Intuitively, the STG of a PBN $\mathcal{B} = (G, \mathcal{P})$ can be considered a natural composition (union on transitions) of STGs of Boolean networks $(G, \{P\})$ for every $P \in \mathcal{P}$. More formally, due to the asynchrony we only consider transitions between states that differ in exactly one element. Let $X_1, X_2 \in \mathcal{X}$ be two states that differ in a single element $v \in V$ ($X_1 \setminus X_2 = \{v\}$, respectively $X_2 \setminus X_1 = \{v\}$). The transition (X_1, X_2) belongs to δ only if at least one parametrisation exists that can reproduce it. More formally, for the case where v is assigned activity value 0 ($X_1 \setminus X_2 = \{v\}$) we require $\exists P \in \mathcal{P} : (n^-(v) \cap X_1) \notin P$. For the case where v is assigned activity value 1 ($X_2 \setminus X_1 = \{v\}$) we require $\exists P \in \mathcal{P} : (n^-(v) \cap X_1) \in P$.

3 Labels on Edges of Influence Graphs

In the previous section, we introduced PBNs and mentioned that the cause of parameter uncertainty comes from the lack of information on biological interaction. The information is, however, often partially available. Part of the biological knowledge can be compiled into two types of conditions on the edges of the influence graph: *monotonicity* and *observability*.

Monotonicity comes in two forms, either as *plus-monotonicity* or the dual *minus-monotonicity*. An edge $(u, v) \in I$ is plus-monotone under parametrisation P iff $\forall \omega \in \Omega_v : u \in \omega \Rightarrow (\omega \in P \vee \omega \setminus \{u\} \notin P)$. Analogically, an edge $(u, v) \in I$ is minus-monotone under P iff $\forall \omega \in \Omega_v : u \in \omega \Rightarrow (\omega \notin P \vee \omega \setminus \{u\} \in P)$. In other words, an edge is plus-monotone if the increase in the activity of the source cannot

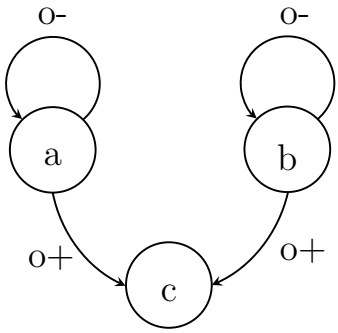

Figure 2. An example of a labelled influence graph (LIG) obtained by introducing a labelling function $\gamma = \{((a,a),\{-,o\}),((b,b),\{-,o\}),((a,c),\{+,o\}),((b,c),\{+,o\})\}$. The labelling strictly determines the Boolean function governing the self-regulation of a and b. In fact, from the initial $2^8 = 256$ parametrisations of the running example, only two parametrisations are possible with the labelling. Also notice that with the labelling γ every interaction of the running example is both observable and monotone. We refer to such labelling function as full labelling and to LIG with full labelling as fully labelled.

cause a decrease in the activity of the target and minus-monotone if the increase in the activity of source cannot cause an increase in the target activity.

On the other hand, an edge $(u,v) \in I$ is observable under parametrisation P if $\exists\omega \in \Omega_v : |\{\omega \setminus \{u\}, \omega \cup \{u\}\} \cap P| = 1$. In other words, an edge is observable if there exists a combination of regulators such that the change in the activity of the source causes a change in the activity of the target.

Naturally, monotonicity and observability may be used to restrict possible parametrisations. We therefore equip the influence graph with a labelling function $\gamma : I \to 2^{\{+,-,o\}}$ specifying the conditions imposed on every edge. A *Labelled Influence Graph* (LIG) is thus a tuple $\mathcal{G} = (V, I, \gamma)$. The set of possible parametrisations of $\mathcal{G}$ is $\{P \in \mathcal{P}_{(V,I)} \mid \forall i \in I : P \text{ satisfies the conditions imposed by } \gamma(i)\}$. An example of a labelling function and a LIG is given in Figure 2 using the running example as the original influence graph.

4 Parametrisation Encoding

In previous sections, we introduced the concept of parametrisation of Boolean networks and natural constraints to implement partial knowledge about the model. In practice, however, known methods are not scalable when applied to PBNs as introduced due to combinatorial explosion. In fact, the combinatorial explosion occurs for PBNs in two instances. First the state space is exponential in the number of components (note $\mathcal{X} = 2^V$). The state space explosion affects as well standard Boolean networks and equivalent models (this is addressed more closely in Section 5). Second, the number of possible parametrisations is also exponential with the RCs ($\mathcal{P}_G = 2^\Omega$). In this section, we dedicate ourselves to the second cause of combinatorial explosion, the number of parametrisations. Here we introduce a novel approach to encode some special sets of parametrisations relevant for our application.

The need to encode parametrisations is required especially for generating processes (possible behaviours) of the PBN. Although processes may be infinite, any reachable state is reachable by at least one finite process. We therefore only require

finite processes to be reachability-complete. A formal definition follows.

Definition 4.1 Let $(\mathcal{G}, \mathcal{P})$ be a PBN with STG $(\mathcal{X}, \delta)$. A process of length $k \in \mathbb{N}$ is a sequence of states $\pi = (X_1, \ldots, X_k) \in \mathcal{X}^k$ where $\forall i \in \{1, \ldots, k-1\} : (X_i, X_{i+1}) \in \delta$.

Let $\pi = (X_1, \ldots, X_k)$ be a process and $X \in \mathcal{X}$ be a state such that $(X_k, X) \in \delta$. Then $\pi' = \pi \cdot X$ is a process of length $k+1$ and we say π' is an extension of π.

PBNs represent several different model possibilities introduced by individual parametrisations. Whereas the individual parametrisations should be exclusive, the dynamics of a PBN (given by its STG) do not distinguish them. It is therefore possible to have processes in the PBN which mix different parametrisations. These processes may not correspond to a process of any individual parametrisation. In order to avoid exploring such incoherent processes we assign to every process (viewed either as a sequence of transitions, or as a partially ordered multi-set of transitions) a set of admissible parametrisations.

Let $(G, \mathcal{P}_G)$ be a PBN and $(X_1, X_2) \in \delta$ a transition of the respective STG such that $X_1 \setminus X_2 = \{v\}$ for some $v \in V$. We call such a transition the *inhibition* of v. Every parametrisation that allows the inhibition of v in state X_1 must necessarily assign the target value of $X_1 \cap n^-(v)$ to 0. Furthermore, it is sufficient for the target value of $X_1 \cap n^-(v)$ to be fixed to 0 for the parametrisation to allow transition (X_1, X_2). We apply a similar reasoning to *activations* ($X_2 \setminus X_1 = \{v\}$ for some $v \in V$). An activation requires the associated RC $X_1 \cap n^-(v)$ to have target value 1. As such, we can define an associated regulatory context of a transition $d = (X, X') \in \delta$ as $\omega_d = n^-(v) \cap X$ where $\{v\} = X \triangle X'$ (by $\triangle$ we mean symmetric difference).

Any transition changes the value of exactly one component. Thus any transition is either exclusively an activation or an inhibition of some component. An arbitrary set of transitions $D \subseteq \delta$ is therefore uniquely given as the union of set of inhibitions D_I and set of activations D_A ($D = D_A \cup D_I$). We now formalise the notion of feasible parametrisations of any set of transitions under the notion of *parameter context* (PC).

Definition 4.2 Let $(G, \mathcal{P})$ be a PBN with STG $(\mathcal{X}, \delta)$. We define a function $p : 2^\delta \to 2^\mathcal{P}$ that assigns to every set of transitions the set of parametrisations that allow all of the transitions. Formally, given any $D \subseteq \delta$, we set $p(D) = \{P \in \mathcal{P} | \forall d \in D_A : (\omega_d \in P) \wedge \forall d \in D_I : (\omega_d \notin P)\}$ where D_A and D_I are sets of activations and inhibitions (respectively) such that $D_A \cup D_I = D$. We call $p(D)$ the parameter context of D for any $D \subseteq \delta$. One can remark that $p(D) = \bigcap_{d \in D} p(\{d\})$.

We extend the definition to processes in a natural fashion. Let $\pi = (X_1, \ldots, X_k)$ be a process. By PC of π we mean $p(\pi) = p(\{(X_i, X_{i+1}) | i \in \{1, \ldots, k-1\}\})$.

A naive approach to computing the PC as defined above could be to enumerate all exponentially many parametrisations. It is, however, precisely thanks to $\mathcal{P}_G = 2^\Omega$ that by introducing the set-inclusion order to parametrisations, we obtain a Boolean lattice $(\mathcal{P}_G, \subseteq)$. We now provide intuition behind the use of lattices for PC encoding.

Let us consider a fully parametric PBN $(G, \mathcal{P}_G)$. As was mentioned above, the

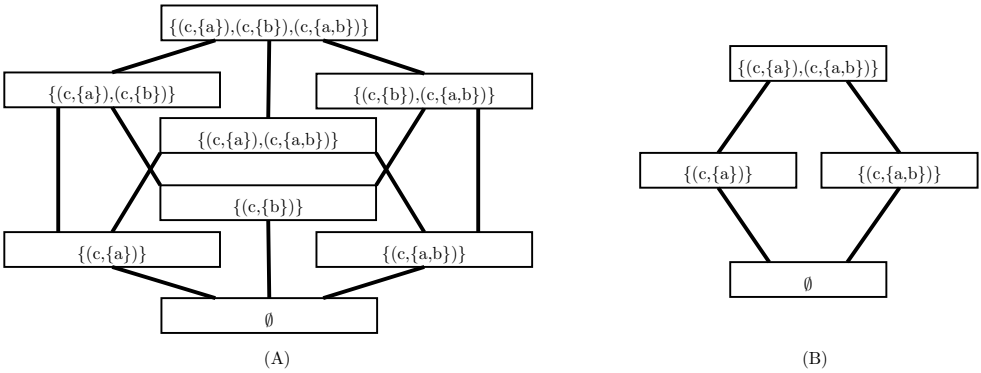

Figure 3. Hasse diagrams of the lattices representing parameter contexts for the regulation of component c in the unlabelled running example. **(A)** The PC of transition $(\{c\}, \emptyset)$, i.e., $p(\{(\{c\}, \emptyset)\})$. **(B)** The restricted PC after transition $(\{b, c\}, \{b\})$ is included $(p(\{(\{c\}, \emptyset), (\{b, c\}, \{b\})\}))$.

PC of arbitrary single transition d only contains parametrisations that set target value of ω_d to the same value. If d is an inhibition of some $v \in V$, then we know $\forall P \in p(\{d\}) : \omega_d \notin P$. Keeping the set-inclusion order in mind the largest parametrisation in $p(\{d\})$ is $\Omega \setminus \{(v, \omega_d)\}$. In fact, $p(\{d\})$ is a prime ideal of the lattice $(\mathcal{P}_G, \subseteq)$ with sole principal (maximal) element $\Omega \setminus \{(v, \omega_d)\}$ (an ideal I of lattice L is prime if it is proper and for any $a, b \in L$ such that $a \wedge b \in I$ either $a \in I$ or $b \in I$ holds). Analogously, if d is an activation of some $v \in V$, then the PC $p(\{d\}) = \{P \in \mathcal{P}_G | \omega_d \in P\}$ is a prime filter of $(\mathcal{P}_G, \subseteq)$ with sole principal (minimal) element $\{(v, \omega_d)\}$ (a filter is prime under the same conditions as an ideal but instead of infimum $(\wedge)$, supremum is used $(\vee)$).

Since $p(D \cup D') = p(D) \cap p(D')$ (Def. 4.2), one can remark that the PC of any set $D_I \subseteq \delta$ such that $\forall d \in D_I : d$ is an inhibition is an ideal of $(\mathcal{P}_G, \subseteq)$. Respectively, the PC of any set of activations $D_A \subseteq \delta$ is a filter. It is well known that the intersection of an arbitrary ideal and arbitrary filter is either empty or a convex sub-lattice. Moreover, any convex sub-lattice can be uniquely represented by intersection of an ideal and a filter [12]. This allows us to represent any convex sub-lattice of $(\mathcal{P}_G, \subseteq)$ by only the maximal element (ideal) and minimal element (filter). As any set of transitions can be split into a set of inhibitions and a set of activations, it is clear that any PC can be encoded by minimal and maximal elements. An example of the PCs represented as convex sub-lattices is visualised in Figure 3.

The results we have provided hold for fully parametric PBN $(G, \mathcal{P}_G)$. In case of a LIG $\mathcal{G}$, however, the lattice $(\mathcal{P}_\mathcal{G}, \subseteq)$ is not guaranteed to exists. This is as there may exist two parametrisations in $\mathcal{P}_\mathcal{G}$ such that their infimum (supremum) does not belong to $\mathcal{P}_\mathcal{G}$. For illustration consider the running example with one observable interaction, e.g. labelling $\gamma = \{((a, c), \{o\})\}$. The interaction (a, c) is observable under parametrisations $\{(c, \emptyset)\}$ and $\{(c, a)\}$ and thus both $\{(c, \emptyset)\}, \{(c, a)\} \in \mathcal{P}_\mathcal{G}$. No interaction is observable under the intersection (infimum) $\emptyset = \{(c, \emptyset)\} \cap \{(c, a)\}$ and namely (a, c) is not observable meaning $\emptyset \notin \mathcal{P}_\mathcal{G}$.

To address this issue we propose an over-approximation of a PBN $\mathcal{B} = (\mathcal{G}, \mathcal{P}_\mathcal{G})$ constructed as $\mathcal{B}' = (\mathcal{G}, [\mathcal{P}_\mathcal{G}])$ where we use $[\mathcal{P}]$ to denote the smallest convex sub-

lattice such that $\mathcal{P} \subseteq [\mathcal{P}]$. On a similar note, we introduce an over-approximative PC $p' : 2^{\delta} \to 2^{\mathcal{P}_G}$ such that $p'(D) = [p(D)]$ or $p'(D) = \emptyset$ if $p(D) = \emptyset$. The labelling function γ introduces dependencies between target values of individual RCs and therefore computing $p'(D) \cap p'(D')$ may not be sufficient to obtain the correct $p'(D \cup D')$ contrary to p. However, we can resolve this issue with the following method.

Our method relies on knowledge of $p'(\pi)$ for some process $\pi = (X_1, \ldots, X_k)$ to compute the PC of an arbitrary extension $\pi \cdot X$ where $X \in \mathcal{X}$ is a compatible state. Since we know that $p(\pi \cdot X) \subseteq p(\pi)$ and in conjecture $p'(\pi \cdot X) \subseteq p'(\pi)$ the PC of the extension can only be smaller than the PC of π. The method thus continuously removes elements from $p'(\pi)$ until $[p(\pi \cdot X)]$ is reached. The elements are removed by successively applying *restrictions*. A restriction is the combination of a regulatory context $\omega \in \Omega_v$ for some $v \in V$ and a value $i \in \{0, 1\}$. A restricted PC is then a PC $\mathcal{P}$ such that $\forall P \in \mathcal{P} : \omega \in P$ if $i = 1$ and $\omega \notin P$ if $i = 0$. In other words, a restriction ensures all parametrisations in the restricted PC to have the same target value for a given RC.

The method recognises two causes of restriction. First, the extension itself requires the transition (X_k, X) to be allowed. Second, the edge labels introduce dependencies between target values of individual RCs. The method detects these dependencies and restricts the PC accordingly. For a more detailed explanation of the method see Appendix A.1.

One of the most important properties of the method is the preservation of reachability. Since the method guarantees that $p'(\pi \cdot X) = [p(\pi \cdot X)]$ and namely $p'(\pi \cdot X) = \emptyset$ if $p(\pi \cdot X) = \emptyset$, any process π such that $p'(\pi) \neq \emptyset$ is guaranteed to also have $p(\pi) \neq \emptyset$ and vice versa. This property becomes important in Section 5 where we construct a compact representation of reachable state space. Thanks to the reachability being preserved any state reached by the over-approximation p' is guaranteed to be also reachable by p and vice versa. This allows us to compute the reachable states within the over-approximation p'. Reachability is, however, only guaranteed to be preserved if the input $p'(\pi)$ of the method is correct. Cases exist where the initial $[\mathcal{P}_\mathcal{G}] \neq \mathcal{P}_G$. A pre-computation is therefore necessary to determine $[\mathcal{P}_\mathcal{G}]$. The pre-computation itself is detailed in Appendix A.2.

5 Parametric Unfolding

Previously, we introduced an encoding of parametrisations to alleviate the combinatorial explosion induced by all possible combinations of RCs. In this section we address the combinatorial explosion of the state space of PBNs and standard Boolean networks accordingly. Biological networks are often considerably sparse in nature and contain a high amount of concurrent interactions. Partial order reduction approaches are therefore meaningful for dealing with the state space explosion in case of standard networks. Petri net unfoldings are a prime example of a structure exploiting the concurrency of transitions. This section is therefore dedicated to application of unfoldings to PBNs and parametric setting in general.

We will now introduce unfoldings for PBNs using the PCs given by p'. Intuitively, the unfolding is an acyclic (tree-like) representation of all the processes of the PBN starting in a given initial state. Although an equivalent Petri net can be constructed for any PBN, we do not require this Petri net explicitly to be able to unfold the PBN. We define the (parametric) unfolding of a PBN as an event structure. Hence, our construction is similar to Petri net unfoldings [8,7]. The only difference is the source of the events, in our case the PBN versus a Petri net. Thus, PBN unfolding and Petri net unfolding are structurally identical. A special treatment was required for construction of complete finite prefixes of the PBN unfoldings when determining which branches can be cut-off without loss of reachability.

In general, an event structure is a triplet $\mathcal{E} = (E, \leq, \#)$ where E is the set of events, $\leq \subseteq E \times E$ is a partial order relation on E called *causality relation* and $\# \subseteq E \times E$ is an antisymmetric, irreflexive relation called *conflict relation* satisfying:

(i) $\forall e \in E : \{e' \in E \mid e' \leq e\}$ is finite.

(ii) $\forall e, e', e'' \in E : (e \# e' \wedge e' \leq e'') \Rightarrow e \# e''$.

For our purposes we extend the event structure by a set of conditions B (we adopt the Petri net unfolding notation) to provide better intuition behind causality and conflict relations in our setting. First let us define the set of all events $\overline{E}$ and conditions $\overline{B}$ possible for a given PBN. As the definitions of events and conditions are interdependent, we define a hierarchy of sets E_i and B_i. First let $B_0 = \{(\bot, v, j) \mid v \in V \wedge j \in \{0, 1\}\}$. We then define $E_i = \{(\beta, v) \mid v \in V \wedge \beta \subseteq \bigcup_{j \in \{0,\ldots,i-1\}} B_j \wedge \beta \cap B_{i-1} \neq \emptyset\}$ and $B_i = \{(e, v, j) \mid v \in V \wedge j \in \{0, 1\} \wedge e \in E_i\}$ for all $i \in \mathbb{N}$. The desired $\overline{E} = \bigcup_{i \in \mathbb{N}} E_i$ and $\overline{B} = \bigcup_{i \in \mathbb{N}_0} B_i$ thus become the infinite unions.

Every condition $b \in \overline{B}$ is of the form $b = (e, v, i)$, where $e \in \overline{E} \cup \{\bot\}$ is a predecessor (parent) event of b if it exists, or $\bot$; otherwise, $v \in V$ is the component of PBN represented by condition b and $i \in \{0, 1\}$ is the value of v in b. Intuitively, a condition represents the possibility of a process reaching a state where component v has value i by following event e. Analogically, every event $e \in E$ is of the form $e = (\beta, v)$ where $\beta \subseteq \overline{B}$ is the set of predecessors (*preset*) of e and $v \in V$ is the component whose value changes by firing e. Intuitively, an event e represents the possibility of component v changing value under influence of regulators in β.

Events closely resemble the transitions of the STG (δ). In fact, if the event $e = (\beta, v)$ is *well-formed* (satisfies $n^-(v) \cup \{v\} = \{u \mid (e', u, i) \in \beta\}$ and $|n^-(v) \cup \{v\}| = |\beta|$) we can define an associated RC $\omega_e = \{u \in n^-(v) \mid \exists (e', u, i) \in \beta : i = 1\}$ much like for transitions. We can also make a distinction between activations and inhibitions between the events. We say that an event $e = (\beta, v)$ is an activation of v if $\exists (e', v, 0) \in \beta$ and analogously, e is an inhibition of v if $\exists (e', v, 1) \in \beta$. Any well-formed event is exclusively either activation or inhibition. This allows us to extend the PC function p and in turn also p' to well-formed events in the natural fashion. Formally, let $E \subseteq \overline{E}$ be a set of well-formed events. Then $p(E) = \{P \in \mathcal{P} \mid \forall e \in E_A : (\omega_e \in P) \wedge \forall e \in E_I : (\omega_e \notin P)\}$ where E_A is a set of activations and E_I is a set of inhibitions such that $E = E_A \cup E_I$.

We can now define causality and conflict relations. Let $e, e' \in E$ be arbitrary.

We say that event $e = (\beta, v)$ is causally dependent on event $e' = (\beta', u)$ $(e' \leq e)$ if $e = e'$ or there exists $b = (e'', w, i) \in \beta$ such that $e' \leq e''$. In other words, e is causally dependent on e' if there exists a directed path from e to e' defined by the parents and presets of conditions and events. If $\neg(e \leq e')$ and $\neg(e' \leq e)$ we say that e and e' are causally independent. Similarly, e is in conflict with e' $(e \# e')$ if there exist events $(\beta, v), (\beta', u) \in E$ such that $(\beta, v) \neq (\beta', u)$, $(\beta, v) \leq e$, $(\beta', u) \leq e'$ and $\beta \cap \beta' \neq \emptyset$. In other words, two events are in conflict if they (or their causal predecessors) use the same condition by two different events.

We can also naturally extend the relations of causality and conflict to conditions simply by setting $\forall(\beta, v) \in \overline{E} : \forall b \in \beta : b < (\beta, v)$ and $\forall(e, v, i) \in \overline{B} : e < (e, v, i)$ and computing the reflexive and transitive closure. The conflict relation is adjusted extending its domain to $\overline{E} \cup \overline{B}$. Additionally, let $x, y \in \overline{E} \cup \overline{B}$ such that x and y are causally independent and not in conflict. We then say x and y are concurrent.

One can notice that $(\overline{B}, \overline{E}, \leq, \#)$ may not be an extended event structure as it is not guaranteed to satisfy the constraints on the causality and conflict relations. We therefore construct the unfolding using subsets of $B \subseteq \overline{B}$ and $E \subseteq \overline{E}$ on which the relations $\leq$ and $\#$ satisfy the desired properties. Let $(\mathcal{G}, \mathcal{P}_\mathcal{G})$ be a PBN with STG $(\mathcal{X}, \delta)$ and $X_0 \in \mathcal{X}$ an initial state. Then the unfolding $\mathcal{U} = (B, E, \leq, \#)$ of $(\mathcal{G}, [\mathcal{P}_\mathcal{G}])$ in state X_0 is constructed as follows.

(i) Start with empty $B = \emptyset$ and $E = \emptyset$.

(ii) For every $v \in V$ add a condition $(\bot, v, i)$ to B such that $i = 1$ if $v \in X_0$ and $i = 0$ otherwise.

(iii) For every $v \in V$ find all sets $\mathcal{C} \subseteq B$ of concurrent conditions (*cosets*) such that $\{u \mid (e, u, i) \in \mathcal{C}\} = n^-(v) \cup \{v\}$ and $|\mathcal{C}| = |n^-(v) \cup \{v\}|$. For every such $\mathcal{C}$ create an event $e = (\mathcal{C}, v)$. If $e \notin E$ then compute $p'(e)$ using the algorithm in Appendix A.1. If $p'(e) \neq \emptyset$ add e to the set E and for every $(e', u, i) \in \mathcal{C}$ add new condition $b = (e, u, j)$ to B where $j = i$ if $u \neq v$ and $j = (1 - i)$ for $u = v$. If at least one event was added to E repeat step (iii).

Although B and E are subsets of $\overline{B}$ and $\overline{E}$ it is apparent from the construction of the unfolding that they are infinite in the general case. We have mentioned in Section 4 that any reachable state is reachable by a finite process. As the unfolding is a representation of all the processes of the network and the number of states is finite, there exist finite prefixes of the unfolding from which all the reachable states can be recovered. We refer to such a prefix as a *complete finite prefix* (CFP), and show below a possible construction. First, we define an equivalent of a process within the unfolding traditionally referred to as configuration.

Definition 5.1 A set $C \subseteq E$ is a configuration if it is conflict free $(\nexists e, e' \in C : e \# e')$ and causally closed $\forall : e \in C, e' \in E : e' \leq e \Rightarrow e' \in C$.

By $[e]$ we denote a special configuration called local configuration of e, $[e] = \{e' \in E \mid e' \leq e\}$.

Any configuration C corresponds to at least one process of the PBN given by completing the partial order on the transitions equivalent to events in C. We can

therefore assign to every finite configuration a terminal (final) state $X_C = X_0 \triangle \{v_1\} \triangle \cdots \triangle \{v_k\}$ where $C = \{e_1 = (\beta_1, v_1), \ldots, e_k = (\beta_k, v_k)\}$ and $k \in \mathbb{N}_0$.

Let C be a configuration and $F \subseteq E$ such that $C \cap F = \emptyset$. We say that F is an extension of C if $C \cup F$ is a configuration. Let us now consider all possible extensions of C. Any extension of C corresponds to a process in the original PBN starting in state X_C and every such process is represented by an extension. As such, we can see that all extensions of C define an unfolding of the PBN in state X_C and initial parametrisation $p'(C)$. Furthermore, any extension possible in $\mathcal{P} \subseteq p'(C)$ is also possible under $p'(C)$.

Let us now consider two configurations $C, C' \subseteq E$ such that $X_C = X_{C'}$ and $p'(C) \subseteq p'(C')$ and an extension $F \subseteq E$ of C. Since F is an extension of C it surely belongs to the unfolding of the PBN in state X_C with $p'(C)$ and namely

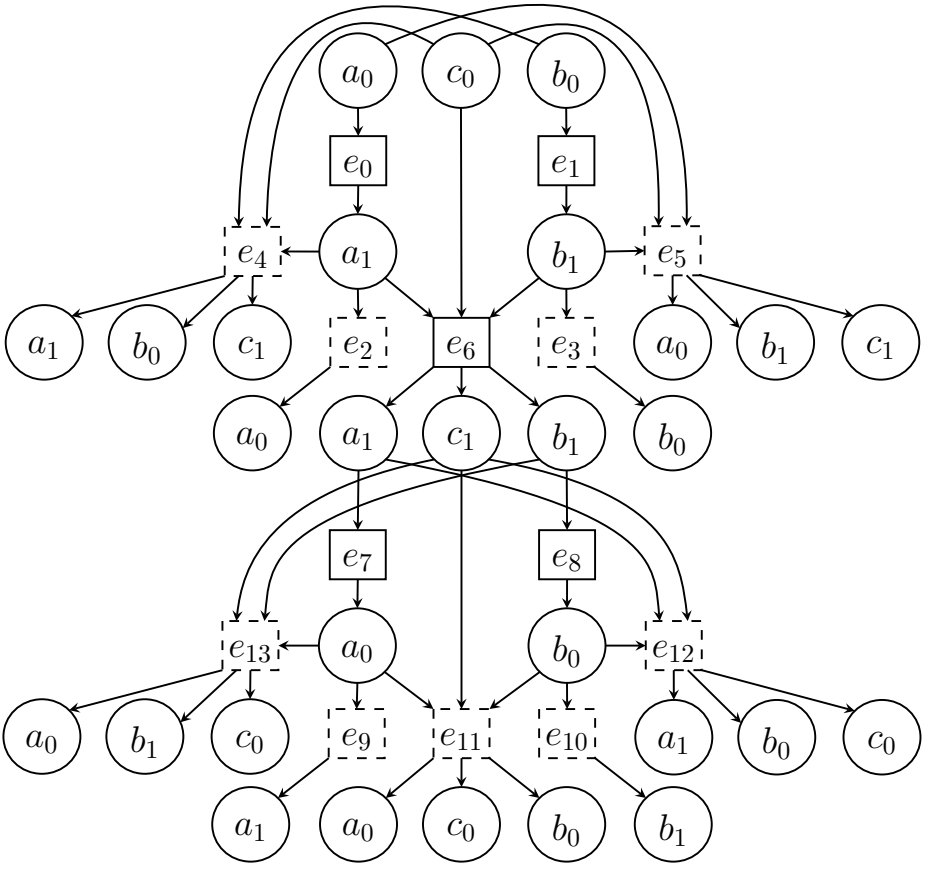

Figure 4. The complete finite prefix obtained by unfolding the running example equipped with full labelling function as illustrated in Figure 2. The complete finite prefix is visualised as a Petri net. The conditions, labelled by the component they represent and the value of the component (v_i for (e, v, i)), are represented by circles. The events, labelled by numbering (e_i for the i-th event), are represented by rectangles. Cut-off events are additionally marked with dashed borders. Notice that the order of concurrent activation/inhibition of species a and b is abstracted.

it also belongs to the unfolding in X_C with $p'(C')$. Unfolding of a PBN is given uniquely by an initial state and set of feasible parametrisations. The unfolding in X_C with $p'(C')$ must therefore be isomorphic to the unfolding in $X_{C'}$ with $p'(C')$. And especially, there must be an extension $F' \subseteq E$ of C' isomorphic to F. This holds for any extension of C meaning any information captured by the extensions of C is already covered by extensions of C' and is redundant from the reachability point of view.

This result is interesting especially for local configurations. Let $e, e' \in E$ be such that $X_{[e]} = X_{[e']}$ and $p'([e]) \subseteq p'([e'])$. As there is no need to explore extensions of $[e]$ we omit them from the CFP. We formalise this by the notion of *cut-off* events. Once an event e is marked cut-off, no other event $e' \in E$ such that $e < e'$ is added to the CFP. Esparza et al. [8] show for Petri net unfoldings that a specific order on the configurations called adequate order is required to guarantee that no reachability is lost by cut-offs. As our unfolding notion is equivalent to Petri net unfoldings we have the same requirement on the order. By $\prec$ we will further understand the total adequate order as defined in [8] adjusted for our definition of unfolding (see Appendix B for a definition). A formal definition of a cut-off event follows.

Definition 5.2 An event $e \in E$ is considered a cut-off event if there exists a different event $e' \in E$ such that $X_{[e]} = X_{[e']}$, $[e'] \prec [e]$ and $p'([e]) \subseteq p'([e'])$.

The total adequate order $\prec$, however, does not correlate with inclusion order over PCs. In other words, $C \prec C'$ does not guarantee $p(C') \subseteq p(C)$ and vice versa. As such a situation may occur where local configurations of two events $e, e' \in E$ lead to the same state $X_{[e]} = X_{[e']}$ and $p([e]) \subseteq p([e'])$ also holds. Ideally, in such a situation, the extensions of e should not be explored as they are all redundant with some extensions of e'. However, due to $[e] \prec [e']$, e will not be designed as cut-off (as e' does not exist yet) and some extensions of e may have been explored before e'. In our algorithm, when e' is added to the CFP, e is marked as cut-off, and we remove its (unnecessarily explored) extensions.

Standard complete finite prefixes of Petri nets computed using a total adequate order for extensions have a number of non-cut-off events which does not exceed the size of the marking graph (state space) of the Petri net [8]. This claim does not hold in our setting, because several events with the same marking can exist in our CFP of PBNs (the cut-offs depend also on the PC). However, because of the resulting partial ordering of transitions in the CFP, one can easily argue that the number of processes in the CFP is smaller than the number of traces in the STG.

The CFP computed for the fully labelled version of the running example is shown in Figure 4.

6 Experiments

In this section we present some initial results on biological models and compare with the symbolic representation employed in [10]. The results have been obtained

by a prototype implementation in Python of a **pa**rametric **un**folder named *Pawn*.[2]

The comparison is done regarding the size of the structure representing all the possible traces. We therefore compare the size of the unfolding – typically represented as the number of non-cut-off events with the number of states in the complete symbolic execution tree, obtained with the tool `SPuTNIk` [10]. Therefore, the difference accounts for the partial-order reduction implicitly achieved by the unfolding, in the scope of Parametric Biological Regulatory Networks.

The experiments were conducted for two different biological models. First we use a boolean model of the gene regulatory network underlying mammalian cortical area development [11]. The model is depicted in Figure 5 (A). The reachable state space has been explored with respect to two defferent initial states. First, all species has been considered inactive. Second, all species has been considered inactive with the only exception of Fgf8.

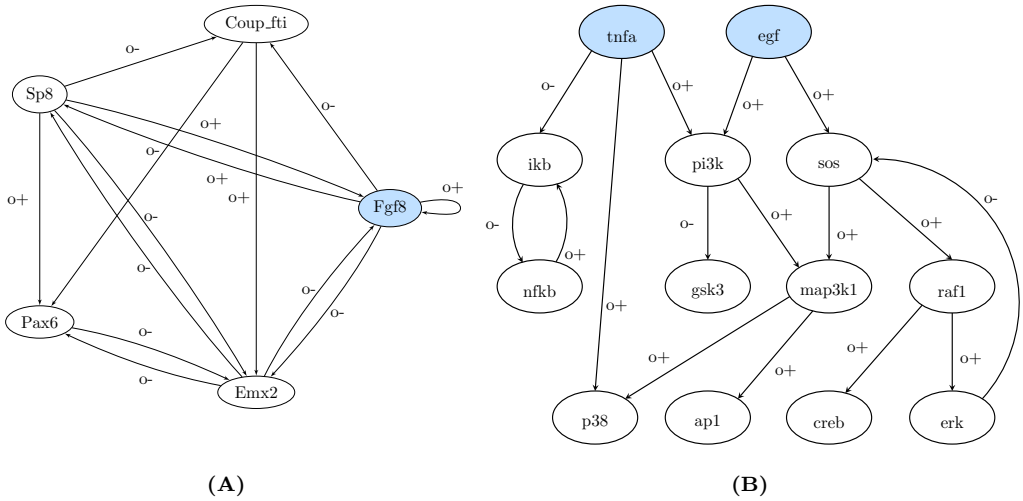

Figure 5. **(A)** A genetic regulatory network controlling the cortical area development. The state marked in blue has been set to initial value 1 in one of the experiments. **(B)** Model of signalling pathway of EGF-TNFα. The only two states that start with initial value 1 are marked in blue.

The second model is a signalling pathway, EGF-TNFα, as studied in [16,17] (Figure 5 (B)). In this case the initial state was for every specie inactive save for the two input species, *tnfa* and *egf*, which were active ($X = \{tnfa, egf\}$). The results are summarised in Table 2. The execution times for *Pawn* were less than a second ($\sim 0.4s$) and around 8.5 seconds for the bacteriophage λ (both variants) and EGF-TNFα respectively. `SPuTNIk` on the other hand took $\sim$ 1 min and $\sim$ 10 min for the bacteriophage λ with and without Min/Max respectively, and $\sim$ 30 min for EGF-TNFα.

Although no theoretical estimate on the size of the parametric unfolding can be given, it is apparent from the results that exploiting the concurrency allows for considerably smaller representations of the parametric state space in practice. The difference in size is derived mainly from the capability of the unfoldings to

[2] *Pawn* is available online: https://github.com/GeorgeKolcak/Pawn

Model	Unfolding Events	Unfolding Events (with cut-offs)	Symbolic Execution Size
Cortical (Fgf8=1)	1,054	3,530	8,312
Cortical (Fgf8=0)	554	1,939	8,312
EGF-TNFα	1,057	2,658	534,498

Table 2
Comparison of the size of the obtained structures between unfolding and the symbolic representation for different models.

exploit concurrency. Therefore, if there are n different concurrent transitions firing, unfolding does not distinguish the order in which they are fired and only explores one possibility. The symbolic execution on the other hand explores all $n!$ possible firing sequences only to obtain the same result each time. This is especially apparent in sparse networks that contain a high number of concurrent transitions.

In case of the two initial conditions experimented in the cortical development model, the same state space is reachable. However, different sizes of unfoldings in the two cases show that our technique is sensitive with respect to the concrete initial state determining the construction of the unfolding.

7 Conclusion

We offer a new platform for parameter identification of logical regulatory networks based on Petri net unfoldings. Our contribution addresses several issues. First, we introduce a novel approach to encoding parametrisations allowing for efficient analysis of parametric Boolean networks. We employ the encoding in practice for computing feasible parametrisations for all possible behaviours of the system. Accompanying methods are also presented for efficiently computing the feasible parametrisations of extensions of behaviours within the encoding. This set of parametrisations can be efficiently constrained by monotonous and observability criteria. Future work may also consider taking into account other constraints, such as the existence of particular fixed points. Although only an over-approximation of the set of feasible parametrisations is given in the general case, reachability preservation is guaranteed by our method. The refinement of the over-approximation is considered for future work.

Next we analyse the possibility of using Petri net unfoldings to exploit concurrency in parametric models of biological networks. We present a modification to allow for unfolding of parametric Boolean networks and couple the unfoldings with the encoding of parametrisations to neutralise both sources of combinatorial explosion in parametric regulatory networks.

In this article, the presentation of the framework is focused on Boolean networks. The formalization of an extension of our parameter space encoding and unfolding to parametric multivalued regulatory networks is currently under investigation.

Last but not least, we provide a prototype implementation of the introduced methods and compare with existing methods on relevant biological examples. The comparison proves our methods capable of representing the reachable state space of parametric regulatory networks in much smaller structures than previous approaches. This compression opens the possibilities of efficient further analysis of

parametric networks via the parametric unfoldings.

References

[1] Albert, R. and H. G. Othmer, *The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster*, Journal of Theoretical Biology **223** (2003), pp. 1 – 18.

[2] Baier, C. and J. Katoen, "Principles of Model Checking," MIT Press, 2008.

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

[4] Bernot, G., 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** (2004), pp. 339 – 347.

[5] Corblin, F., E. Fanchon and L. Trilling, *Applications of a formal approach to decipher discrete genetic networks*, BMC Bioinformatics **11** (2010), pp. 1–21.

[6] de Jong, H., *Modeling and simulation of genetic regulatory systems: A literature review*, Journal of Computational Biology **9** (2002), pp. 67–103.

[7] Esparza, J. and K. Heljanko, "Unfoldings: A Partial-Order Approach to Model Checking," Monographs in Theoretical Computer Science. An EATCS Series, Springer Berlin Heidelberg, 2010.

[8] Esparza, J., S. Römer and W. Vogler, *An improvement of McMillan's unfolding algorithm*, Formal Methods in System Design **20** (2002), pp. 285–310.

[9] Fromentin, J., J. P. Comet, P. L. Gall and O. Roux, *Analysing gene regulatory networks by both constraint programming and model-checking*, in: *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2007, pp. 4595–4598.

[10] Gallet, E., M. Manceny, P. Le Gall and P. Ballarini, "Formal Methods and Software Engineering: 16th International Conference on Formal Engineering Methods, ICFEM 2014, Luxembourg, Luxembourg, November 3-5, 2014. Proceedings," Springer International Publishing, Cham, 2014 pp. 155–170.

[11] Giacomantonio, C. E. and G. J. Goodhill, *A boolean model of the gene regulatory network underlying mammalian cortical area development*, PLOS Computational Biology **6** (2010), pp. 1–13.

[12] Grätzer, G., "Lattice Theory: Foundation," SpringerLink : Bücher, Springer Basel, 2011.

[13] Kitano, H., *Computational systems biology*, Nature **420** (2002), pp. 206–210.

[14] Klarner, H., A. Streck, D. Šafránek, J. Kolčák and H. Siebert, "Computational Methods in Systems Biology: 10th International Conference, CMSB 2012, London, UK, October 3-5, 2012. Proceedings," Springer Berlin Heidelberg, Berlin, Heidelberg, 2012 pp. 207–226.

[15] Laubenbacher, R. and B. Stigler, *A computational algebra approach to the reverse engineering of gene regulatory networks*, Journal of Theoretical Biology **229** (2004), pp. 523 – 537.

[16] MacNamara, A., C. Terfve, D. Henriques, B. P. Bernabé and J. Saez-Rodriguez, *State–time spectrum of signal transduction logic models*, Physical Biology **9** (2012), p. 045003.

[17] Ostrowski, M., L. Paulevé, T. Schaub, A. Siegel and C. Guziolowski, "Computational Methods in Systems Biology: 13th International Conference, CMSB 2015, Nantes, France, September 16-18, 2015, Proceedings," Springer International Publishing, Cham, 2015 pp. 170–181.

[18] Thomas, R., *Boolean formalization of genetic control circuits*, Journal of Theoretical Biology **42** (1973), pp. 563 – 585.

[19] Wang, R.-S., A. Saadatpour and R. Albert, *Boolean modeling in systems biology: an overview of methodology and applications*, Physical Biology **9** (2012), p. 055001.

Appendices

A Parameter Context of Extensions

Here we present in detail the method to compute the overapproximation PC p' of process extensions expanding on the intuition given in Section 4.

A.1 Method for Parameter Context Restriction

Let us first fix a notation for minimum and maximum of the PCs. For $[\mathcal{P}]$ we denote the minimum as $0_{\mathcal{P}} = \bigcap_{P \in \mathcal{P}} P$ and the maximum as $1_{\mathcal{P}} = \bigcup_{P \in \mathcal{P}} P$. Next we define restriction formally and denote some special relationships between PCs and RCs.

Definition A.1 A restriction is a tuple $r = (v, \omega, i)$ where $\omega \in \Omega_v$ is RC of $v \in V$ and $i \in \{0, 1\}$ is the new target value of ω.

Let $\mathcal{P}$ be a PC (convex lattice). $\mathcal{P}$ can be restricted by restriction $r = (v, \omega, i)$ to obtain a restricted PC $\mathcal{P}' = \{P \in \mathcal{P} \mid \omega \in P \text{ if } i = 1 \text{ or } \omega \notin P \text{ if } i = 0\}$

Using the notation with minimal and maximal element a restriction $(v, \omega, 0)$ results in $\omega \notin 1_{\mathcal{P}'}$ and $(v, \omega, 1)$ leads to $\omega \in 0_{\mathcal{P}'}$. As long as $0_{\mathcal{P}'} \subseteq 1_{\mathcal{P}'}$ holds ($\mathcal{P}' \neq \emptyset$) the above constraints enforce $\omega \notin 0_{\mathcal{P}'} \triangle 1_{\mathcal{P}'}$.

Definition A.2 Let $\omega \in \Omega_v$ be RC of some $v \in V$ and let $u \in n^-(v)$ be an arbitrary regulator of v. We say that $\omega' \in \Omega_v$ such that $\omega' = \omega \triangle \{u\}$ is a u-pair regulatory context (u-pair) of ω. In other words, two RCs of v are reciprocal u-pairs of each other if they differ only in activity of regulator u. We will denote the u-pair of ω as ω^u.

Let $[\mathcal{P}]$ be some PC. We say that ω is restricted under $[\mathcal{P}]$ iff $\omega \notin 0_{\mathcal{P}} \triangle 1_{\mathcal{P}}$. In other words, ω is said to be restricted under $[\mathcal{P}]$ if all parametrisations in $\mathcal{P}$ assign ω the same target value. In such case we say $[\mathcal{P}]$ fixes the target value of ω. On the other hand, if the target values for ω differ, then we say ω is free under $[\mathcal{P}]$.

Additionally, if both ω and the u-pair of ω are restricted under some $[\mathcal{P}]$ we say that ω and the u-pair are u-restricted under $[\mathcal{P}]$.

As mentioned in Section 4, the method works with two causes of restrictions. In fact, the first cause – the transition itself is always the first restriction to happen. Let $\pi = (X_1, \ldots, X_k)$ be a process and transition (X_k, X) an inhibition of v. If there is indeed no restriction necessary for all parametrisations to allow the transition, then we know that $X_k \cap n^-(v) \notin 1_{p'(\pi)}$ and thus especially $X_k \cap n^-(v) \notin 1_{p(\pi)}$. Surely then $p((X_k, X)) \subseteq p(\pi)$ must hold leading to $p(\pi \cdot X) = p(\pi)$. We may even extend this reasoning as it can be shown that any edge label based restriction is a result of a prior restriction and dependency introduced by labelling function.

Let us first consider a case when a restriction occurs due to monotonocity. Let $(u, v) \in I$ be an interaction such that $+ \in \gamma((u, v))$. (Once again we assume plus-monotonocity without loss of generality as the reasoning is analogously for minus-monotonocity.) We now define necessary and sufficient conditions for the monotonocity of interaction (u, v) to introduce a restriction $(v, \omega, 1)$: $u \in \omega$ and

$\omega^u \in 0_{p(\pi \cdot X)}$. Clearly these conditions are sufficient as using the definition of monotonocity we have $u \in \omega \Rightarrow (\omega \in P \vee \omega^u \notin P)$ thus obviously $\omega \in P$. In fact, the given conditions are not necessary strictly speaking. However, under the assumption that it was indeed the monotonocity of the (u, v) interaction that requires the restriction $(v, \omega, 1)$, it becomes easy to see that they are indeed necessary. In other words, those are the necessary conditions for a plus-monotonocity criterion to introduce a new restriction. Let us therefore assume restriction $(v, \omega, 1)$ was required. As we are working with plus-monotonocity, the only way to enforce restriction on ω is when $u \in \omega$. Moreover, the restriction with value 1 can only be imposed if $\omega^u \in P$ for all P giving us $\omega^u \in 0_{p(\pi \cdot X)}$. Clearly then restriction $(v, \omega, 1)$ enforced by plus-monotonocity requires a prior restriction $(v, \omega^u, 1)$ to occur.

Similarly we may consider a restriction enforced by observability. Let $(u, v) \in I$ be an interaction such that $o \in \gamma((u, v))$. Just as in the monotonocity case we define necessary and sufficient conditions for the observability of interaction (u, v) to introduce restriction $(v, \omega, 1)$: $\forall \omega' \in \Omega_v \setminus \{\omega, \omega^u\} : (\omega' \in 0_{p(\pi \cdot X)} \wedge \omega'^u \in 0_{p(\pi \cdot X)}) \vee (\omega' \notin 1_{p(\pi \cdot X)} \wedge \omega'^u \notin 1_{p(\pi \cdot X)})$ and $\omega^u \notin 1_{p(\pi \cdot X)}$. Although the conditions are more complex to formalise compared to monotonocity, they are fundamentally very straightforward, the first condition requires any u-pair of regulatory contexts other than ω to be u-restricted to the same value in $p(\pi \cdot X)$ while the second condition simply requires the u-pair of ω to have target value fixed to 0. Again we show that the given conditions are sufficient. As any $\omega' \in \Omega_v$ except ω and ω^u has target value equal to the u-pair ω'^u it is apparent that to satisfy the existential condition in the definition of observability we need target values of ω and ω^u to differ. Thus $\omega \in 0_{p(\pi \cdot X)}$ as the value of ω^u is fixed to 0 and the desired restriction happened. The conditions again, are not necessary in the general sense, however it can be shown they are necessary if the restriction $(v, \omega, 1)$ is imposed by the observability of (u, v). Let therefore $\omega \in 0_{p(\pi \cdot X)}$ be enforced by observability of (u, v). For ω and the u-pair of ω to be able to satisfy observability is is clearly necessary that their target values differ and as ω is fixed to 1 we get $\omega^u \notin 1_{p(\pi \cdot X_{k+1})}$. Moreover for observability to strictly determine the target value of ω it must hold for all the other u-pairs that they do not satisfy observability of (u, v) giving us the first criterion $(\forall \omega' \in \Omega_v \setminus \{\omega, \omega^u\} : (\omega' \in 0_{p(\pi \cdot X)} \wedge \omega'^u \in 0_{p(\pi \cdot X)}) \vee (\omega' \notin 1_{p(\pi \cdot X)} \wedge \omega'^u \notin 1_{p(\pi \cdot X)}))$ as if any of those u-pairs differed in target value the observability would be satisfied with arbitrary target value of ω. Again we can draw the conclusion that there must have been a prior restriction that allowed the sufficient and necessary conditions to become true before the restriction $(v, \omega, 1)$ was enforced by observability.

We can conduct analogical reasoning for restrictions with target values 0. As any restriction with the exception of the first one imposed by the transition itself has one prior restriction acting as a cause, we can define a causal partial order on the restrictions. It is easy to see that such a partial order defines a tree topology on the restrictions. We will now proceed with the definition of the method itself. Let $\pi = (X_0, \ldots, X_k)$ be a process and $X \in \mathcal{X}$ a state such that $(X_k, X) \in \delta$. The following method computes $p'(\pi \cdot X)$ using $p'(\pi)$.

(i) Set $0_{p(\pi \cdot X)} = 0_{p(\pi)}$ and $1_{p(\pi \cdot X)} = 1_{p(\pi)}$ and initialise an empty FIFO queue of

restrictions.

(ii) Push $(v, X_k \cap n^-(v), i)$ where $i = 0$ if (X_k, X) is inhibition of v or $i = 1$ if (X_k, X) is activation of v to the queue of restrictions.

(iii) While the queue of restrictions is not empty, pop (v, ω, i) from the queue and execute the following:
   (a) If $i = 0$ set $1_{p(\pi\cdot X)} = 1_{p(\pi\cdot X)} \setminus \{\omega\}$ else $0_{p(\pi\cdot X)} = 0_{p(\pi\cdot X)} \cup \{\omega\}$. If no change occurred by the previous operation, then skip to next element in the queue of restrictions.
   (b) For every monotone influence $(u, v) \in I$ push (v, ω_u, i) where ω_u is the u-pair of ω to the queue of restrictions if one of the following is true:
   $i = 0$, $+ \in \gamma((u, v))$ and $u \in \omega$.
   $i = 1$, $+ \in \gamma((u, v))$ and $u \notin \omega$.
   $i = 0$, $- \in \gamma((u, v))$ and $u \notin \omega$.
   $i = 1$, $- \in \gamma((u, v))$ and $u \in \omega$.
   (c) If there exists only one regulatory context $\omega' \in \Omega_v$ such that ω' is open under $p'(\pi \cdot X)$ as defined by $0_{\pi\cdot X}$ and $1_{\pi\cdot X}$ then for every observable influence $(u, v) \in I$ such that there does not exist an u-closed pair of regulatory contexts with different target values push $(v, \omega', 1 - j)$ where j is the fixed target value of u-pair of ω' to the queue of restrictions.

(iv) Output $0_{\pi\cdot X}$ and $1_{\pi\cdot X}$ as minimum and maximum of $p'(\pi \cdot X)$ respectively. If $0_{\pi\cdot X} \leq 1_{\pi\cdot X}$ does not hold we consider the result to be empty ($p'(\pi \cdot X) = \emptyset$) in accordance with the definition as intersection of ideal and filter.

In simpler terms, the aforementioned method traverses the tree of restrictions with a breadth-first search while constructing it on the run. Monotonocity (point (iii)(b)) is enforced straightforwardly as the universal quantifier in the definition requires every u-pair to satisfy the condition. Observability (point (iii)(c)) definition contains an existential quantifier and is therefore enforced only if no other u-pair can satisfy the condition. The method always terminates and has a polynomial complexity of $\mathcal{O}(\Omega^2)$ in the worst case as every RC can have it's value restricted at most twice.

Assuming that the input is correct ($p'(\pi) = [p(\pi)]$) the method computes correct $p'(\pi\cdot X) = [p(\pi\cdot X)]$ ($p(\pi\cdot X) = \emptyset \Rightarrow p'(\pi\cdot X) = \emptyset$). Let us first assume $p(\pi\cdot X) \neq \emptyset$. As $[p(\pi \cdot X)]$ is by definition the smallest convex sub-lattice containing $p(\pi \cdot X)$ it is enough to prove $p(\pi \cdot X) \subseteq p'(\pi \cdot X) \subseteq [p(\pi \cdot X)]$.

Let us first show $p(\pi \cdot X) \subseteq p'(\pi \cdot X)$. Let (v, ω, i) be an arbitrary restriction that gets added to the restriction queue during the algorithm. If the restriction does not change $0_{p'(\pi\cdot X)}$ and $1_{p'(\pi\cdot X)}$ within the algorithm then $p'(\pi \cdot X) = p'(\pi)$ and the condition is trivially satisfied since we know $p(\pi \cdot X) \subseteq p(\pi) \subseteq [p(\pi)] = p'(\pi)$. Let us therefore expect that the restriction (v, ω, i) had an effect on $p'(\pi \cdot X)$. Thanks to the self-duality of the Boolean matrix, the duality of plus-/minus-monotonocity and symmetry of observability it is enough to consider $i = 1$ without loss of generality. Thus we get that $0_{p(\pi\cdot X)}$ gets extended by ω by the restriction. Let us now discussion the cause of this restriction.

(i) The restriction $(v, \omega, 1)$ is added to the queue due to transition (X_k, X) such that $\{v\} = X \setminus X_k$ and $n^-(v) \cap X_k = \omega$. By definition $p((X_k, X)) = \{P \in \mathcal{P}_\mathcal{G} \mid \omega \in P\}$ and $p(\pi \cdot X) = p(\pi) \cap p((X_k, X))$ it thus follows that $\forall P \in p(\pi \cdot X) : \omega \in P$ and therefore $\omega \in 0_{p(\pi \cdot X)}$.

(ii) The restriction $(v, \omega, 1)$ is enforced by some prior restriction r and monotonocity of $(u, v) \in I$ (plus-monotonocity without loss of generality). If we apply this reasoning to restriction in the partial order given by the restriction tree we can assume r does not break the desired $p(\pi \cdot X) \subseteq p'(\pi \cdot X)$ ($0_{p'(\pi \cdot X)} \subseteq 0_{p(\pi \cdot X)}$ is enough to consider for the case $i = 1$). We can now use the sufficient and necessary conditions for restriction $(v, \omega, 1)$ to happen. Thus we get $u \in \omega$ and $\omega^u \in 0_{p'(\pi \cdot X)}$ (by extension $\omega^u \in 0_{p(\pi \cdot X)}$). Since $p(\pi \cdot X) \subseteq \mathcal{P}_\mathcal{G}$ the monotonocity restrictions must hold for every $P \in p(\pi \cdot X)$. Thus namely for any $P \in p(\pi \cdot X)$ such that $\omega^u \in P$ is must also hold $\omega \in P$ giving us the desired $\omega \in 0_{p(\pi \cdot X)}$.

(iii) The restriction $(v, \omega, 1)$ is enforced by some prior restriction r and observability of $(u, v) \in I$. Just as in the previous case we can assume r does not break the desired $0_{p'(\pi \cdot X)} \subseteq 0_{p(\pi \cdot X)}$. Again, we use the sufficient and necessary conditions for $(v, \omega, 1)$ giving us $\forall \omega' \in \Omega_v \setminus \{\omega, \omega^u\} : (\omega' \in 0_{p(\pi \cdot X)} \wedge \omega'^u \in 0_{p(\pi \cdot X)}) \vee (\omega' \notin 1_{p(\pi \cdot X)} \wedge \omega'^u \notin 1_{p(\pi \cdot X)})$ and $\omega^u \notin 1_{p(\pi \cdot X)}$. Applying the same reasoning and $p(\pi \cdot X) \subseteq \mathcal{P}_\mathcal{G}$ we know the target value of ω must be different from ω^u for every parametrisation in $p(\pi \cdot X)$.

We will now show $p'(\pi \cdot X) \subseteq [p(\pi \cdot X)]$. If $[p(\pi)] = [p(\pi \cdot X)]$, then we know $p'(\pi \cdot X) \subseteq p'(\pi) = [p(\pi)] = [p(\pi \cdot X)]$ and the result is trivial. Let us thus assume $[p(\pi \cdot X)] \subseteq [p(\pi)]$. This clearly means there exists at least one regulatory context ω such that $\omega \in 0_{p(\pi \cdot X) \setminus 0_{p(\pi)}}$ or $\omega \in 1_{p(\pi)} \setminus 1_{p(\pi \cdot X)}$. Just as in the previous case we can assume $\omega \in 0_{p(\pi \cdot X)} \setminus 0_{p(\pi)}$ without loss of generality thanks to the duality and symmetry properties. We will now show that for any such ω the algorithm definitely adds restriction $(v, \omega, 1)$ to the restriction queue where $v \in V$ is such that $\omega \in \Omega_v$. Analogically to the above discussion on the restriction we do a discussion on the nature of ω. Since we know that any parametrisation with target value of ω equal to 0 does not belong to $p(\pi \cdot X)$ despite belonging to $p(\pi)$ there are only few possible explanations of this occurrence (we once again use the sufficient and necessary conditions for monotonocity and observability enforcement).

(i) $\omega = X_k \cap n^-(v)$ is the regulatory context used by the activation (X_k, X). The corresponding restriction $(v, \omega, 1)$ is always added to the restriction queue. Thus $p'(\pi \cdot X)$ is restricted appropriately thus retaining the desired $0_{p(\pi \cdot X)} \subseteq 0_{p'(\pi \cdot X)}$.

(ii) $u \in \omega$ and ω^u also belongs to $0_{p(\pi \cdot X)} \setminus 0_{p(\pi)}$ and $(u, v) \in I$ is plus-monotone (without loss of generality). We may assume that restriction $(v, \omega^u, 1)$ is in the queue by conducting this discussion on ω^u. By restriction tree we are guaranteed to eventually have $X_k \cap n^-(v)$ as the cause RC. Since $(v, \omega^u, 1)$ is in the restriction queue it will eventually be handled by the algorithm. Since $\omega^u \notin 0_{p'(\pi)}$ the restriction will make a difference leading to monotonocity and

observability check. By the plus-monotonocity of (u, v) the restriction $(v, \omega, 1)$ will be enqueued.

(iii) There exists some $\omega' \in 0_{p(\pi \cdot X)} \subseteq 0_{p'(\pi \cdot X)}$ (without loss of generality although it could also belong to $1_{p(\pi)} \setminus 1_{p(\pi \cdot X)}$) such that $\omega \neq \omega'$ and $\omega' \in \Omega_v$ and $(u, v) \in I$ is observable. In this case we have two different possibilities for how the sufficient and necessary conditions are satisfied.

(a) $\omega' = \omega^u$. In this case we get $\forall \omega'' \in \Omega_v \setminus \{\omega, \omega^u\} : (\omega'' \in 0_{p(\pi \cdot X)} \wedge \omega''^u \in 0_{p(\pi \cdot X)}) \vee (\omega'' \notin 1_{p(\pi \cdot X)} \wedge \omega''^u \notin 1_{p(\pi \cdot X)})$ and $\omega^u \notin 1_{p(\pi \cdot X)}$. We can thus assume $(v, \omega^u, 0)$ to be in the restriction queue. From the first condition we know that every RC is u-restricted under $p'(\pi \cdot X)$ by the time restriction $(v, \omega^u, 0)$ is handled and the observability of (u, v) is yet to be satisfied. As such ω is left as the last free context by restriction of ω^u and to ensure observability the algorithm adds $(v, \omega, 1)$ to the restriction queue.

(b) Otherwise we get $\forall \omega'' \in \Omega_v \setminus \{\omega, \omega^u, \omega', \omega'^u\} : (\omega'' \in 0_{p(\pi \cdot X)} \wedge \omega''^u \in 0_{p(\pi \cdot X)}) \vee (\omega'' \notin 1_{p(\pi \cdot X)} \wedge \omega''^u \notin 1_{p(\pi \cdot X)})$, $\omega'^u \notin 1_{p(\pi \cdot X)}$ and $\omega^u \notin 1_{p(\pi \cdot X)}$. And our assumption becomes existence of restriction $(v, \omega', 1)$ in the restriction queue. As we know $\omega' \notin 0_{p(\pi)}$ the restriction will make a difference and as it leaves behind ω as the last free RC and (u, v) with still unfulfilled observability requirement the $(v, \omega, 1)$ restriction must be added to the restriction queue.

Finally, let us assume $p(\pi \cdot X) = \emptyset$ (and $p(\pi) \neq \emptyset$). We now show that in such a case the algorithm computes $\neg(0_{p'(\pi \cdot X)} \subseteq 1_{p'(\pi \cdot X)})$ we interpret as $p'(\pi \cdot X) = \emptyset$. It is easy to see, as $0_{p'(\pi \cdot X)}$ is only extended and $1_{p'(\pi \cdot X)}$ only reduced that the method cannot recover from such a malformed state.

Let us again assume (X_k, X) to be activation of some $v \in V$ without loss of generality. We know $p(\pi \cdot X) = p(\pi) \cap p((X_k, X)) = \emptyset$. Since $p((X_k, X))$ contains all parametrisations $P \in \mathcal{P}_\mathcal{G}$ such that $X_k \cap n^-(v) \in P$ and $p(\pi) \subseteq \mathcal{P}_\mathcal{G}$ it must hold that $\forall P \in p(\pi) : X_k \cap n^-(v) \notin P$ and by extension $X_k \cap n^-(v) \notin 1_{p(\pi)}$. $p'(\pi) = [p(\pi)]$ further gives us $\forall P \in p'(\pi) : X_k \cap n^-(v) \notin P$. By the structure for the algorithm $1_{p'(\pi \cdot X)} \subseteq 1_{p'(\pi)}$ and therefore $X_k \cap n^-(v) \notin 1_{p'(\pi \cdot X)}$. Furthermore the algorithm will for sure handle restriction $(v, X_k \cap n^-(v), 1)$ due to the transition (X_k, X) resulting in $X_k \cap n^-(v) \in 0_{p'(\pi \cdot X)}$ giving us $\neg(0_{p'(\pi \cdot X) \subseteq 1_{p'(\pi \cdot X)}})$.

A.2 The Initial Parameter Context Overapproximation

As the method relies on the knowledge of $p'(\pi)$ to compute $p(\pi \cdot X)$ it is necessary to be able to compute the initial $p'(\emptyset) = [\mathcal{P}_\mathcal{G}]$. Although $[\mathcal{P}_\mathcal{G}] = \mathcal{P}_G$ often holds, there are some cases when a restriction is viable.

As regulations of individual components are independent (note that both monotonocity and observability only speak about RCs of the same component) it is enough to consider restrictions for regulation of each component separately. Let thus $v \in V$ be an arbitrary component. Let $A = \{u \in n^-(v) \mid + \in \gamma((u, v))\}$ be the set of all regulators of v with plus-monotone interaction and $I = \{u \in n^-(v) \mid - \in \gamma((u, v))\}$ be the set of all regulators of v with minus-monotone interaction. We

now consider two parametrisations $P_s = \Omega \setminus \{I\}$ and $P_i = \{A\}$. We can show easily that any monotonocity criterion on interaction with v is satisfied by both P_s and P_i. Monotonocity for (u, v) is satisfied by default if both ω and u-pair of ω have the same target value. As such only the context I has to be analysed. Let $u \in A$ be arbitrary. We get $u \in I_u$ by definition and $I \notin P_s$ as well as $I \notin P_i$ thus monotonocity of (u, v) is satisfied by both P_s and P_i. Now let $u \in I$ be arbitrary. By definition $u \in I$ and thus by $I \notin P_s$ and $I \notin P_i$ we again satisfy monotonocity of (u, v) for both P_s and P_i. Let us now discuss observability. For any $u \in A$ observability is trivially satisfied under both P_s and P_i as $I \notin P_s$ and $I \cup \{u\} \in P_s$ and similarly $A \in P_i$ and $A \setminus \{u\} \notin P_i$. Similarly let $u \in I$ be arbitrary then $I \notin P_s$ and $I \setminus \{u\} \in P_s$ make (u, v) observable under P_s and $A \in P_i$ and $A \cup \{u\} \notin P_i$ makes (u, v) observable also under P_i. Finally, observability is also satisfied for any other $u \in (n^-(v) \setminus A) \setminus I$ as $I \notin P_s$ and $I \cup \{u\} \in P_s$ and $A \in P_i$ and $A \cup \{u\} \notin P_i$. It thus becomes clear that P_s and P_i satisfy both monotonocity and observability for arbitrary labelling function γ. As such, edge labelling cannot reduce the initial lattice beyond the maximum P_s and minimum P_i for each $v \in V$ $(\forall v \in V : [\{P_s, P_i\}] \subseteq [\mathcal{P}_\mathcal{G} \cap (\{v\} \times \Omega_v)])$.

Let us now consider a set of regulators $N \subseteq n^-(v)$ such that $N \cap A = \emptyset$ and $N \cap I = \emptyset$ and parametrisations $P_{ns} = \Omega \setminus \{N \cup I\}$ and $P_{ni} = \{A \cup N\}$. It is easy to see that both P_{ns} and P_{ni} have the same properties as P_s and P_i and thus satisfy arbitrary γ. In fact, if $N = \emptyset$, then we trivially get $P_{ns} = P_s$ and $P_{ni} = P_i$. Let us now consider $N \neq \emptyset$. Then clearly both P_s and P_{ns} are valid parametrisations. Moreover the supremum of P_{ns} and P_s according to the set inclusion order is $P_ns \cup P_s = \Omega$. Similarly both P_i and P_{ni} are also valid and their infimum $P_{ni} \cap P_i = \emptyset$. Thus, if there exists any interaction (u, v) such that $+ \notin \gamma((u, v))$ and $- \notin \gamma((u, v))$ then the initial lattice of parametrisations for regulations of v will be equal to the fully parametrised case $([\mathcal{P}_\mathcal{G} \cap (\{v\} \times \Omega_v)] = \mathcal{P}_G \cap (\{v\} \times \Omega_v))$. And we can safely conclude that for $[\mathcal{P}_\mathcal{G}] \subset \mathcal{P}_G$ there must exist at least one $v \in V$ such that $\forall u \in n^-(v) : + \in \gamma((u, v)) \vee - \in \gamma((u, v))$.

Since the parametrisations Ω and $\emptyset$ are both plus-monotone and minus-monotonous for all interactions, it is obvious that just monotonocity is not enough for $[\mathcal{P}_\mathcal{G}] \subset \mathcal{P}_G$. In fact, Ω and $\emptyset$ are not observable for any interaction (an interaction that is both plus and minus-monotone is anti-observable) and thus a single observable interaction (u, v) is sufficient to remove Ω and $\emptyset$ from the set of feasible parametrisation. In fact, if every interaction $(u, v) \in I$ is monotone for some $v \in V$ and at least one of them also observable, the initial PC can be restricted to maximum P_s and minimum P_i $([\mathcal{P}_\mathcal{G} \cap (\{v\} \times \Omega_v)] = [P_s, P_i])$. To prove this let us consider any parametrisation P such that $I \in P$. Since (u, v) is minus-monotone for every $u \in I$ every RC of the form $I \setminus \{u\}$ must also belong to P and by iterative application of the minus-monotonocity condition $\forall J \subseteq I : J \in P$. Similarly (u, v) is plus-monotone for any other $u \in A$. As such, for any $J \subseteq I$ we get $J \cup \{u\} \in P$ and by iterative application $P = \Omega$ and the observability is not satisfied. Thus we may conclude that no parametrisation P such that $I \in P$ is feasible. An analogical construction can be conducted for P such that $A \notin P$ reaching $P = \emptyset$.

The construction of the initial PC $p'(\emptyset) = [\mathcal{P}_\mathcal{G}]$ can thus be done algorithmically

as follows.

(i) Start with $[\mathcal{P}_\mathcal{G}] = \mathcal{P}_G$.

(ii) Find all components $v \in V$ such that $\forall u \in n^-(v) : + \in \gamma((u,v)) \vee - \in \gamma((u,v))$ and $\exists u \in n^-(v) : o \in \gamma((u,v))$.

(iii) For every such v construct the sets $A = \{u \in n^-(v) \mid + \in \gamma((u,v))\}$ and $I = \{u \in n^-(v) \mid - \in \gamma((u,v))\}$ and restrict the minimum by $(v, A) \in 0_{\mathcal{P}_\mathcal{G}}$ and the maximum by $(v, I) \notin 1_{\mathcal{P}_\mathcal{G}}$.

The computation of the initial PC is similar in the general case of multivalued PRNs. The only difference is the target value limitation for I for every suitable component $v \in V$ (target value of A is still set to be at least 1). Instead of setting target value of I to 0, in general the value is limited to be at most $k_v - 1$ where $k_v \in \mathbb{N}$ is the maximum value of component v.

B Total Adequate Order

Here we extend the total adequate order $\prec_F$ introduced in [8] to our formalism of unfoldings as introduced in section 5. Let $\mathcal{U} = (B, E, \leq, \#)$ be an unfolding of some PRN and let $C_1, C_2 \subseteq E$ be two finite configurations of $\mathcal{U}$.

Let $\ll$ be an arbitrary total order on all RCs $(v, \omega) \in \Omega$. We can always find such an order as Ω is finite. Then we define a function φ such that for every finite configuration C, $\varphi(C) = (\omega_{e_1}, \ldots, \omega_{e_k})$ is a vector of regulatory contexts such that $C = \{e_1, \ldots, e_k\}$ ordered by $\ll$. $\varphi(C)$ is thus a variant of a Parikh vector of associated RCs of C. We extend the order $\ll$ to ordered vectors as a lexicographical order. Essentially the same construction has been used for Petri net unfoldings in [8] using transitions of the original Petri net instead of RCs.

Furthermore let us define the Foata normal form of configuration C, $FC(C)$ as follows.

(i) $FC_1(C) = \varphi(\{e \in C \mid \forall e' \in C : \neg(e' < e)\})$ is the φ of all events in C minimal with respect to the causality relation.

(ii) For $1 < i \in \mathbb{N} : FC_i(C) = \varphi(\{e \in C \mid \forall e' \in C : e' < e \Rightarrow e' \in \bigcup_{j<i-1} FC_j(C)\})$ is the φ of all events in C without all the previous $FC_j(C)$ minimal in respect to the causality relation.

(iii) $FC(C) = (FC_1(C), \ldots, FC_k(C))$ where $k \in \mathbb{N}$ is the largest natural number such that $FC_k(C) \neq \emptyset$. Such k is guaranteed to exist as C is finite.

Intuitively, the Foata normal form $FC(C)$ is a layered representation of C in respect to causality relation and represents steps in which events of C can fire if all concurrent events fire synchronously. We again extend $\ll$ to the Foata normal forms as a lexicographical order on the vectors of $FC(C)$. The construction is again equivalent to the one proposed in [8] for Petri net unfoldings. Finally, the order $\prec$ as used in Section 5 is defined as follows.

Definition B.1 Let $\mathcal{U} = (B, E, \leq, \#)$ be an unfolding of some PRN and $C_1, C_2 \subseteq$

E two finite configurations of $\mathcal{U}$. We say that $C_1 \prec C_2$ if one of the following conditions holds.

- $|C_1| < |C_2|$
- $|C_1| = |C_2| \wedge \varphi(C_1) \ll \varphi(C_2)$
- $|C_1| = |C_2| \wedge \varphi(C_1) = \varphi(C_2) \wedge FC(C_1) \ll FC(C_2)$