Unfoldings and Abstract Interpretation for Parametric Biological Regulatory Networks

Juraj Kolčák

université
PARIS-SACLAY

Unfoldings and Abstract Interpretation for Parametric Biological Regulatory Networks

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n°580, sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique
Unité de recherche: Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France
Référent: ENS Paris-Saclay

Thèse présentée et soutenue en ligne, le 06/07/2021, par

Juraj KOLČÁK

Composition du jury:

Pascale Le Gall Professeure, Centrale Supélec	Présidente
Gilles Bernot Professeur, Université Nice Sophia Antipolis, Polytech Nice Sophia	Rapporteur
Paolo Zuliani Professeur associé, Newcastle University	Rapporteur
Barbara König Professeure, Universität Duisburg-Essen	Examinatrice
Heike Siebert Professeure, Freie Universität Berlin	Examinatrice
Stefan Haar Professeur, Université Paris-Saclay GS Informatique et science du numérique	Directeur
Loïc Paulevé Chargé de recherche, CNRS	Codirecteur

Thèse de doctorat

NNT: 2021UPASG048

Résumé

L'analyse de la dynamique des réseaux de régulation biologique, notamment des réseaux de signalisation et de régulation génique, fait face à l'incertitude du modèle de calcul exact. En effet, la plupart des connaissances disponibles concernent l'existence d'interactions (éventuellement indirectes) entre des entités biologiques (espèces), par ex. protéines, ARN, gènes, etc. Les détails sur la manière dont les différents régulateurs d'une même cible coopèrent, et plus encore sur les taux cohérents pour ces interactions, sont cependant rarement disponibles. A cet égard, des approches de modélisation qualitative sous forme de réseaux de régulation discrets, tels que les réseaux booléens et Thomas, offrir un niveau d'abstraction approprié pour la dynamique du réseau de régulation biologique. Les réseaux de régulation discrets étant basés sur un graphe d'influence, ils nécessitent peu de paramètres supplémentaires par rapport aux modèles quantitatifs classiques. Néanmoins, la détermination des paramètres discrets est un défi bien connu et un goulot d'étranglement majeur pour fournir des prédictions robustes à partir de modèles informatiques.

Le graphe d'influence d'un réseau de régulation établit des dépendances pour l'évolution de chaque espèce, spécifiées par les arêtes dirigées du graphe. Les dépendances seules, cependant, ne suffisent pas pour spécifier la fonction logique régissant l'évolution d'une espèce. Au lieu de cela, les fonctions logiques associées à chaque espèce, contraintes par le graphe d'influence, sont codées dans les paramètres d'un réseau de régulation discret. L'espace des fonctions logiques admissibles est alors représenté par un réseau de régulation paramétrique. D'une part, les réseaux de régulation paramétriques peuvent être utilisés pour l'identification de valeurs de paramètres pour lesquelles le réseau de régulation discret résultant satisfait des propriétés (dynamiques) données. L'identification des paramètres des réseaux de régulation peut ainsi être vue comme un exemple particulier de synthèse de modèle, dans le cadre contraint du graphe d'influence sous-jacent. D'autre part, les réseaux de régulation paramétriques peuvent être analysés comme un modèle autonome, pour faire des prédictions robustes vis-à-vis de la variabilité du réseau.

L'analyse de la dynamique du réseau de régulation paramétrique est entravée par la double explosion combinatoire, de l'espace d'états et de l'espace des paramètres. Dans cette thèse, nous développons de nouvelles méthodes d'analyse de réseau de régulation paramétrique, sous forme de sémantique spécialisée, visant à atténuer l'explosion combinatoire. Tout d'abord, nous in-

troduisons une interprétation abstraite de l'ensemble des évaluations de paramètres admissibles (paramétrisations). L'abstraction permet de représenter n'importe quel ensemble de paramétrisations par un encodage de taille constante, au prix d'une sur-approximation conservatrice. Deuxièmement, nous élevons la sémantique d'ordre partiel sous la forme d'un déploiement des réseaux de Petri vers des réseaux de régulation paramétriques. Les graphiques d'influence des réseaux de régulation biologique ont tendance à être relativement clairsemés, ce qui permet une grande concurrence. Cela peut être exploité par des méthodes de réduction d'ordre partiel pour produire des représentations d'espace d'état concises.

Les deux approches visent à aborder les deux aspects de la double explosion combinatoire et sont introduites de manière compatible, ce qui permet de les utiliser simultanément. Une telle application est soutenue par une implémentation prototype utilisée pour mener des expériences sur divers réseaux de régulation paramétriques. Nous considérons en outre des raffinements des méthodes, comme une méthode de réduction de modèle à la volée portée aux réseaux de régulation paramétriques à partir de réseaux d'automates.

Abstract

The analysis of dynamics of biological regulatory networks, notably signalling and gene regulatory networks, faces the uncertainty of the exact computational model. Indeed, most of the knowledge available concerns the existence of (possibly indirect) interactions between biological entities (species), e.g. proteins, RNAs, genes, etc. The details on how different regulators of a same target cooperate, and even more so on consistent rates for those interactions, however, are rarely available. In this regard, qualitative modelling approaches in the form of discrete regulatory networks, such as Boolean and Thomas networks, offer an appropriate level of abstraction for the biological regulatory network dynamics. As discrete regulatory networks are based on an influence graph, they require few additional parameters compared to classical quantitative models. Nevertheless, determining the discrete parameters is a well known challenge, and a major bottleneck for providing robust predictions from computational models.

The influence graph of a regulatory network establishes dependencies for the evolution of each specie, specified by the directed edges of the graph. The dependencies alone, however, do not suffice to specify the logical function governing the evolution of a specie. Instead the logical functions associated to each specie, constrained by the influence graph, are encoded within the parameters of a discrete regulatory network. The space of admissible logical functions is then represented by a parametric regulatory network. On the one hand, parametric regulatory networks can be used for identification of parameter values for which the resulting discrete regulatory network satisfies given (dynamical) properties. Parameter identification of regulatory networks can thus be seen as a particular instance of model synthesis, in the constrained setting of the underlying influence graph. On the other hand, parametric regulatory networks may be analysed as a stand-alone model, for making predictions that are robust with respect to variability in the network.

The analysis of parametric regulatory network dynamics is hampered by dual combinatorial explosion, of the state space and of the parameter space. In this thesis, we develop novel methods of parametric regulatory network analysis, in the form of specialised semantics, aimed at alleviating the combinatorial explosion. First, we introduce abstract interpretation for the set of admissible parameter evaluations (parametrisations). The abstraction allows us to represent any set of parametrisations by a constant size encoding, at

the cost of a conservative over-approximation. Second, we lift partial order semantics in the form of unfolding from Petri nets to parametric regulatory networks. The influence graphs of biological regulatory networks tend to be relatively sparse, allowing for a lot of concurrency. This can be harnessed by partial order reduction methods to produce concise state space representations.

The two approaches are aimed at tackling both aspects of the dual combinatorial explosion and are introduced in a compatible manner, allowing one to employ them simultaneously. Such application is supported by a prototype implementation used to conduct experiments on various parametric regulatory networks. We further consider refinements of the methods, such as an on-the-run model reduction method lifted to parametric regulatory networks from automata networks.

Contents

Chapter 1

Introduction

Modelling in systems biology is commonly conducted manually or semi-automatically using the available molecular interaction knowledge. Qualitative models are therefore often preferred for biological systems as they require comparatively little parametrisation on top of the knowledge available in literature and databases. The biological knowledge typically consists of one-on-one interactions, positive and negative, between species (molecules) within the system. Discrete regulatory networks are particularly well suited for representing and generalising such information, making them commonplace in modelling gene regulation and signalling pathways [52, 45, 1, 19, 71, 20] since their introduction in late 60s [44, 69]. Discrete regulatory networks are well known for being able to express complex emerging behaviour, owing in particular to loops on the inter-component influences (commonly known as feedback loops) [68, 4, 27].

The modelling of gene regulatory networks or signalling pathways as discrete regulatory networks presents a challenging task due to the high level of abstraction involved. The inference of discrete regulatory networks can be classically split into two phases. First, available data on specie interaction from databases and literature is used to deduce the topology of the network in the form of an influence graph. Influence graph is a directed graph whose nodes represent the variables (species) of the system and edges the pairwise influences, possibly marked as positive or negative. In the second step, a dynamical model is built from the influence graph by specifying a regulation function. The regulation function determines the variable value evolution across different states of the network, given as vectors of variable values.

While the topology of the discrete regulatory networks in the form of influence graph is often well supported by data from literature and databases, the specification of a regulation function requires additional parameter inference. Indeed, whereas the influence graph establishes the potential dependencies (possibly signed) between the variable value changes, the pairwise dependencies are not sufficient to determine the combined effect of several influences on a single variable. In other words, it is commonly known that two variables both have a positive influence on the value of a third variable. However, it is

rarely known if both of the variables have to be active, i.e. have value above a determined threshold, for the value of the third variable to increase or if just one is sufficient.

In general, the regulation function may take the form of an arbitrary logical function. The individual target values of a variable in possible combinations of the values of its regulator variables, the variables that have an influence on it, are hence discrete parameters. The regulation function is therefore equivalent to an assignment of a value to each such parameter, referred to as *parametrisation*. A discrete regulatory network can thus be specified by an influence graph, encoding the topology, paired with a parametrisation, giving the dynamics. Within the scope of this thesis we focus on discrete regulatory networks where each variable has a finite discrete domain. Finite variable domains guarantee that the number of parameters as well as each parametrisation are also finite, making it possible to enumerate them. The restriction to finite variable domains is in line with models studied in the literature, which typically utilise variable domains of very small size, often Boolean [69, 25, 51, 8, 72].

As biological knowledge on combined effects of two or more regulator variables is scarce, parametrisations cannot be easily derived from literature. We therefore avoid precise parametrisation specification, focusing instead on *parametric regulatory networks*. A parametric regulatory network is, similarly to discrete regulatory networks, based on an influence graph. However, parametric regulatory networks do not rely on a specific regulation function, or equivalently a single parametrisation. The dynamics instead encompass any parametrisation which is compatible with the influence graph (admissible), including any influence signs. A parametric regulatory network is thus a formal model constructed to represent exactly the available biological knowledge. It contains all the pairwise influence information available in the literature, however, no assumptions are made on the unknown regulator interplay, retaining all possibilities by the means of different parametrisations.

The analysis of parametric regulatory networks therefore does not consist merely of asking whether the model satisfies given dynamical properties (e.g. reachability), but with which parametrisations are the properties satisfied. The exploration of possible parametric regulatory network dynamics, however, suffers from dual combinatorial explosion, limiting the scalability. Not only do parametric regulatory networks experience the combinatorial explosion of the state space, where the number of states is exponential in the number of variables, a challenge also for discrete regulatory networks, but the number of parametrisations is in the worst case double exponential in the number of variables.

The dual combinatorial explosion arising from the combination of state space and number of parametrisations both being exponential in the number of variables forms the principal challenge tackled in this thesis. To this end, we propose new semantics for parametric regulatory networks.

First, we propose abstract semantics of parametric regulatory networks. The semantics rely on abstraction of the parametrisation space by the means of bounded convex sublattices of the parametrisation set with a preestablished

order. Our abstraction is not only compact, being represented by only two bounds, but can reflect possible state transitions without explicit enumeration of parametrisations. Furthermore, the abstraction is exact assuming no additional constraints (signs) on the influences.

The abstraction also extends to parametric regulatory networks with signed influences. The influence signs translate into monotonicity constraints on the influences. If a variable is influenced positively (resp. negatively) by a regulator, decrease (resp. increase) in the value of the regulator may not cause increase in the value of the variable. Different constraints are also captured by the abstraction, such as observability. If a variable is observably influenced by a regulator, then there must exist at least one state in which the sole change in the value of the regulator leads to a change in the value of the variable. Indeed, in the general case, we allow for an influence to have no impact on the regulation of a variable in spite of being specified.

Such constraints on influences are handled by the parametrisation space abstraction at the cost of over-approximation. It is shown, however, that the abstraction is tight, i.e. the proposed abstraction is the smallest bounded convex sublattice containing all the admissible parametrisations. The tightness result ensures that if a state is reachable under the abstract semantics, there exists at least one parametrisation which allows a sequence of concrete transitions leading to the state. Thus, while the over-approximation allows for false positives in the form of falsely declaring a state reachable by a particular parametrisation, it may not introduce spurious transitions.

Second, we define partial order semantics for parametric regulatory networks in order to obtain a more compact representation of the reachable state space. The partial order semantics rely on constructing unfoldings, akin to Petri net unfoldings. The unfolding semantics may additionally by combined with either the concrete or abstract semantics of parametric regulatory networks, allowing us to use both the parametrisation space abstraction and partial order reduction at the same time, thus addressing both aspects of the dual combinatorial explosion.

Finally, we introduce a goal-oriented model reduction for parametric regulatory networks. The model reduction relies on polynomial static analysis methods to determine which transitions are guaranteed to not lead to a given target state. This allows us to prune the transitions which are known not to reach the goal, allowing us to avoid exploring dead end branches of the state space. Moreover, the method is compatible with both the abstract and the unfolding semantics of the parametric regulatory networks, allowing for the combination of all three methods.

This thesis expands upon results previously published in [50, 41]. Several discrete regulatory network applications have also benefited from the insights gathered working with the parametric regulatory networks. Namely in the area of concurrency [15] and a new symbolic semantics of Boolean networks that subsume any multivalued or continuous refinement [63].

Outline The thesis is partitioned into four main areas. In the first part, we introduce in detail the necessary theoretical background. Chapter 2 defines the regulatory network model and related concepts, forming the basis for the parametric regulatory network which is the object of our studies. Chapter 3 introduces the unfolding based partial order semantics for Petri nets, including the construction of complete finite prefixes.

The second area deals with our main contribution in the form both abstract semantics and partial order semantics of the parametric regulatory networks. Chapter 4 introduces the parametric regulatory networks as well as their concrete and abstract semantics. Chapter 5 extends both the concrete and abstract semantics of parametric regulatory networks to incorporate additional biological knowledge in the form of influence constraints. Chapter 6 introduces the partial order semantics of parametric regulatory networks based on the Petri net unfolding, including the complete finite prefix construction. Chapter 7 introduces an optimisation of the unfolding procedure for the parametric regulatory networks in a setting with predetermined target configuration.

The third part is dedicated to application areas of our parametric regulatory network semantics and related work. Chapter 8 explores related work on parametric regulatory networks and equivalent models within various application areas. Chapter 9 offers experimental results on the compactness of the state space representation with our parametric regulatory network semantics.

Finally, the last, fourth part gives a summary of our work and outlines possible future work. Chapter 10 offers a brief summary of our contributions. Chapter 11 describes the directions of the currently ongoing and future work related to semantics of parametric regulatory networks and regulatory networks in general.

Notations Π applied to sets denotes the Cartesian product. If the order of the elements matters, we write $\overset{\leq}{\underset{x \in X}{\Pi}} \ldots$ where $\leq$ is a total order on X.

Given a sequence of n elements $\pi = (\pi_i)_{1 \leq i \leq n}$, we write $\widetilde{\pi} \triangleq \{ \pi_i \mid 1 \leq i \leq n \}$ to denote the set of its elements.

Given a monotonic function f, we write $f^* (x)$ to denote the fixpoint of the iteration of the function f initially applied to x.

Given a vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$, we write $\boldsymbol{v} [i \mapsto y]$ to denote the vector identical to $\boldsymbol{v}$ except for the component i value, which is equal to y, $\boldsymbol{v} [i \mapsto y] \triangleq (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{i-1}, y, \boldsymbol{v}_{i+1}, \ldots, \boldsymbol{v}_n)$.

Part I

Background

Chapter 2

Discrete Regulatory Networks

In this chapter we introduce discrete regulatory networks, finite *transition systems* which are commonly employed for modelling biological systems, especially gene regulatory and signalling networks [25, 27, 68].

A discrete regulatory network consists of a finite number (n) of *variables* (*nodes*). Each variable v has associated a finite discrete domain X_v of possible values. For the sake of simplicity, we consider the finite set of variables to be indexed by $\{1, \ldots, n\}$ and by abuse of notation, we unify the variables with their respective indices. The set of *states* (state space) of the discrete regulatory network is then given as vectors of possible values for each variable, $\prod_{v=1}^{n} X_v$.

The dynamics of a discrete regulatory network are captured by a *regulation function* F which specifies new value each variable should take based on the current state. There are several ways to apply the regulation function, differing especially in simultaneity of updates of individual variables. To reflect the possibility of updating individual variables we decompose the function F into local functions f_v for each variable v.

As domain of the regulation function is the cartesian product of the domains of the individual variables, a discrete regulatory network is thus fully specified solely by the regulation function.

Definition 2.1 (Discrete Regulatory Network). A discrete regulatory network of a dimension n is a function $F\colon \prod_{v=1}^{n} X_v \to \prod_{v=1}^{n} X_v$, where, for every $v \in \{1, \ldots, n\}$, X_v is finite.

A state of F is a vector $\boldsymbol{x}$ which assigns each variable a value from the respective domain, $\boldsymbol{x} \in \prod_{v=1}^{n} X_v$.

The regulation functions of individual variables, $f_1, \ldots, f_n$, are obtained as projections of F to the respective variables, for all $u \in \{1, \ldots, n\}$:

$$f_u\colon \prod_{v=1}^{n} X_v \to X_u \quad f_u\colon \boldsymbol{x} \mapsto F(\boldsymbol{x})_u$$

In the rest of the chapter, we elaborate on discrete regulatory networks and introduce some commonly used variations. Namely, in Section 2.1 we

give a detailed description of multiple different semantics which can be used with discrete regulatory networks. Subsequently, in Section 2.2 we introduce additional representation and constraints for discrete regulatory networks. The introduced concepts are used heavily in the parametric version of the model, elaborated on in Chapter 4. In Section 2.3 we introduce some of the most commonly used configurations of discrete regulatory networks, such as Boolean networks or multivalued networks. Finally, for comparison and to give better intuition, we introduce *automata networks*, a model equivalent to the discrete regulatory networks which is defined using interacting automata instead of functions, in Section 2.4.

2.1 Semantics of Discrete Regulatory Networks

The definition of discrete regulatory networks is simple, but significantly flexible, making discrete regulatory networks suitable for a variety of modelling tasks. This flexibility is reflected in numerous different updating schemes – semantics used with discrete regulatory networks. The main distinction of the different semantics lies in *simultaneity* of variable updates. The variables of discrete regulatory networks may change value either simultaneously, all at the same time, (*synchronous* semantics) or individually, one at a time, (*asynchronous* semantics). Additionally, several variations of mixed semantics have been considered. We focus on the most universal of such mixed semantics, called *generalised asynchronous* semantics, which allows arbitrary combination of synchronous and asynchronous transitions.

Synchronous Semantics of Discrete Regulatory Networks

Using synchronous semantics, all variables are updated simultaneously, by the same transition. One can therefore envision the transitions as being simply the application of the function F on the current state of the discrete regulatory network.

Definition 2.2 (Synchronous Semantics). Let F be a discrete regulatory network of dimension n. The synchronous semantics of F is a relation $\xrightarrow[sync]{F} \subseteq \prod_{v=1}^{n} X_v \times \prod_{v=1}^{n} X_v$ defined as:

$$(\boldsymbol{x}, \boldsymbol{y}) \in \xrightarrow[sync]{F} \overset{\Delta}{\iff} \boldsymbol{y} = F(\boldsymbol{x})$$

We use the natural infix notation $\boldsymbol{x} \xrightarrow[sync]{F} \boldsymbol{y}$ to denote membership in the semantics relation, $(\boldsymbol{x}, \boldsymbol{y}) \in \xrightarrow[sync]{F}$.

We further use $\boldsymbol{x} \xrightarrow[sync]{F}{}^{*} \boldsymbol{y}$, where $\xrightarrow[sync]{F}{}^{*} \subseteq \prod_{v=1}^{n} X_v \times \prod_{v=1}^{n} X_v$ is the reflexive and transitive closure of $\xrightarrow[sync]{F}$ to denote reachability of state $\boldsymbol{y}$ from state $\boldsymbol{x}$ by synchronous transitions.

Under synchronous semantics, each state $\boldsymbol{x} \in \prod_{v=1}^{n} X_v$ has exactly one successor, namely $F(\boldsymbol{x})$. Synchronous semantics of discrete regulatory networks are thus fully deterministic. The determinism of the semantics allows for relatively simpler analysis of discrete regulatory networks such as reachability or attractor analysis. However, synchronism assumes strict timing constraints on the model by demanding value changes of each variable to have the same duration, or be otherwise synchronised.

Although several real world applications, such as electronic circuits, outright rely on synchronisation, the use of synchronous semantics is much more debatable in areas where variable updates can take different amounts of time. In particular, the timing of substrate concentration changes in biological systems is not necessarily uniform and the precise timing is often unpredictable or scarcely known. Simultaneity therefore cannot be guaranteed, leading us to explore the asynchronous semantics which assume temporal independence of variable updates.

Asynchronous Semantics of Discrete Regulatory Networks

In asynchronous semantics, the state is updated one variable at a time. Rather than the regulation function F, the transitions are given by application of the individual regulation functions $f_1, \ldots, f_n$. As any of the regulation functions $f_1, \ldots, f_n$ may be chosen to update the state, the asynchronous semantics is nondeterministic.

Definition 2.3 (Asynchronous Semantics). Let F be a discrete regulatory network of dimension n. The asynchronous semantics of $F = (f_1, \ldots, f_n)$ is a relation $\xrightarrow[async]{F} \subseteq \prod_{v=1}^{n} X_v \times \prod_{v=1}^{n} X_v$ defined as:

$$(\boldsymbol{x}, \boldsymbol{y}) \in \xrightarrow[async]{F} \overset{\Delta}{\Longleftrightarrow} D(\boldsymbol{x}, \boldsymbol{y}) = \{v\} \wedge \boldsymbol{y}_v = f_v(\boldsymbol{x})$$

where $D\colon \prod_{v=1}^{n} X_v \times \prod_{v=1}^{n} X_v \to 2^{\{1,\ldots,n\}}$ is the function computing the set of variables with different values in the two input states, $D\colon \boldsymbol{x}, \boldsymbol{y} \mapsto \{v \in \{1, \ldots, n\} \mid \boldsymbol{x}_v \neq \boldsymbol{y}_v\}$.

Similar to the synchronous semantics, we use the infix notation $\boldsymbol{x} \xrightarrow[async]{F} \boldsymbol{y}$ for membership in the semantics relation and the reflexive and transitive closure $\boldsymbol{x} \xrightarrow[async]{F}{}^{*} \boldsymbol{y}$ to denote reachability of state $\boldsymbol{y}$ from state $\boldsymbol{x}$ in F.

Since $D(\boldsymbol{x},\boldsymbol{y})$ is a singleton set for asynchronous transitions $t = (\boldsymbol{x},\boldsymbol{y}) \in \xrightarrow[async]{F}$, we denote the unique element, the only variable that changes value, as $v(t)$ $(D(\boldsymbol{x},\boldsymbol{y}) = \{v((\boldsymbol{x},\boldsymbol{y}))\})$.

One may be led to believe the asynchronous semantics are a refinement of the synchronous semantics. Although true for some discrete regulatory networks, the two semantics are incomparable in the general case. The synchronous semantics often allow *normal transitions* – transitions which are not decomposable into a sequence of asynchronous transitions, thus allowing the synchronous semantics to exhibit behaviours unreachable in the asynchronous case. The existence of normal transitions has been linked to simple structural elements of discrete regulatory networks, called *NOPE cycles*, by Noual et al. [60].

Apart from the two extreme cases, the fully synchronous and the fully asynchronous semantics, other simultaneity restrictions might be considered depending on the domain. As the fully synchronous and fully asynchronous semantics are incomparable with regards to expressivity (reachability) in the general case, it is natural to consider their combination. To this end, we consider *generalised asynchronous* semantics, which allow any subset of variables to be updated at a time, thus allowing the fully asynchronous behaviour as well as normal transitions.

Other approaches are possible, such as globally asynchronous locally synchronous (GALS) employed extensively in circuit design [13], or the, essentially dual concept, of bounded asynchrony [33]. As local synchrony is subsumed by the generalised asynchronous semantics and bounded asynchrony can be modelled with a suitable fairness criterion, we do not treat either in detail.

Generalised Asynchronous Semantics of Discrete Regulatory Networks

In the generalised asynchronous semantics, each transition updates a (nonempty) subset of variables synchronously.

Definition 2.4 (Generalised Asynchronous Semantics). Let F be a discrete regulatory network of dimension n. The generalised asynchronous semantics of $F = (f_1, \ldots, f_n)$ is a relation $\xrightarrow[gen]{F} \subseteq \prod_{v=1}^{n} X_v \times \prod_{v=1}^{n} X_v$ defined as:

$$(\boldsymbol{x},\boldsymbol{y}) \in \xrightarrow[gen]{F} \overset{\Delta}{\iff} \forall\, v \in D(\boldsymbol{x},\boldsymbol{y}) \neq \emptyset,\ \boldsymbol{y}_v = f_v(\boldsymbol{x})$$

where D is again the function computing the set of variables which differ in values between the input states.

Similarly to the synchronous and the asynchronous semantics, we use the infix notation $\boldsymbol{x} \xrightarrow[gen]{F} \boldsymbol{y}$ for membership and the reflexive and transitive closure $\boldsymbol{x} \xrightarrow[gen]{F}{}^{*} \boldsymbol{y}$ to denote reachability of state $\boldsymbol{y}$ from state $\boldsymbol{x}$ in F.

By definition, any behaviour exhibited by either the synchronous or the asynchronous semantics is reproducible in the generalised asynchronous semantics. However, the generalised asynchronous semantics not only retains the nondeterminism of the asynchronous semantics, it allows up to exponentially more transitions [1].

We briefly revisit the question of discrete regulatory network semantics, especially for *Boolean networks* (see Section 2.3), in Chapter 11, where we make connection with the *most permissive* semantics of Boolean networks.

2.2 Influence Graphs

The regulation functions as presented in the definition of discrete regulatory networks (Definition 2.1) take all variables of the system as the input. In real world applications, however, such 'dense' interdependency is rare. In particular, the direct interaction of components of gene regulatory networks and other biological systems is often considerably sparse.

The (in)significance of some inter-variable dependencies introduced topology to discrete regulatory network in the form of a directed graph, called influence (or interaction) graph, whose nodes represent the variables of the network. The edges between the variables then denote the significant influences.

For the purposes of discrete regulatory networks, an influence of variable u on variable v is significant if there exists at least one state, in which the sole change in the value of variable u changes the result of the partial regulation function f_v. We relax this definition, however, allowing also other variables to be declared significant. This is to accommodate for the uncertainty involved with the design of parametric regulatory networks (Chapter 4), the definition of which is strongly tied to the influence graphs. The above criterion then corresponds to the smallest influence graph.

Definition 2.5 (Influence Graph). Let F be a discrete regulatory network of dimension n.

Then a graph $G = \{V, I\}$ such that $V = \{1, \ldots, n\}$ is an influence graph of F if, for each pair of states $\boldsymbol{x}, \boldsymbol{y} \in \prod_{w=1}^{n} X_w$ such that $D(\boldsymbol{x}, \boldsymbol{y}) = \{u\} \wedge f_v(\boldsymbol{x}) \neq f_v(\boldsymbol{y})$, $(u, v) \in I$.

We use $G(F)$ to denote the smallest influence graph of the regulatory network F.

The influence graph allows us to specify which variable values does the target value of a variable v directly depend on. We call such variables the regulators of v.

Definition 2.6 (Regulator). Let $G = (V, I)$ be an influence graph of discrete regulatory network F of dimension n and let $v \in \{1, \ldots, n\}$ be an arbitrary variable of F.

[1] While in asynchronous semantics, there is one transition per variable, the generalised asynchronous semantics have one transition for each subset of variables.

Then, variable $u \in \{1, \ldots, n\}$ is a regulator of v according to G, if $(u, v) \in I$.
We write $R(v)$ to denote the set of all regulators of the variable v.

As the output of the regulation function f_v does not change with different values of variables $u \notin R(v)$, the domain of the regulation function may be restricted to $R(v)$ without loss of information. We thus redefine regulation functions f_v for *regulator states*, which are projections of global states to regulators of v.

Definition 2.7 (Regulator State). Let F be a discrete regulatory network of dimension n, G an influence graph of F and $v \in \{1, \ldots, n\}$ an arbitrary variable of F.

A regulator state of v is a vector $\boldsymbol{\omega} \in \prod_{u \in R(v)} X_u$.

We use $\Omega_v = \prod_{u \in R(v)} X_u$ to denote the set of all regulator states of a variable $v \in \{1, \ldots, n\}$.

We further use $\Omega(F, G)$ (or simply Ω where F and G are obvious from the context) to denote the set of all regulator states of all variables of F, annotated with the respective variables, $\Omega(F, G) = \{ (v, \boldsymbol{\omega}) \mid v \in \{1, \ldots, n\} \wedge \boldsymbol{\omega} \in \Omega_v \}$.

Finally, given an arbitrary state $\boldsymbol{x} \in \prod_{w=1}^{n} X_w$, we use $\omega_v(\boldsymbol{x}) = \boldsymbol{\omega}$ to denote the regulator state $\boldsymbol{\omega} \in \Omega_v$ of variable $v \in \{1, \ldots, n\}$ such that $\forall\, u \in R(v),\ \boldsymbol{\omega}_u = \boldsymbol{x}_u$.

As regulator states are merely projections of the states, the restriction of the regulation functions $f_1, \ldots, f_n$ to variable regulators is straightforward. For all $v \in \{1, \ldots, n\}$:

$$f_v \colon \Omega_v \to X_v \quad f_v \colon \boldsymbol{\omega} \mapsto f_v(\boldsymbol{x})$$

where $\boldsymbol{x} \in \prod_{u=1}^{n} X_u$ is arbitrary such that $\omega_v(\boldsymbol{x}) = \boldsymbol{\omega}$.

On the one hand, smaller domains of regulation functions directly translate into less parameters one needs to evaluate to fully specify the network. Thus, even if the exact regulation functions are unknown, one can greatly simplify the model inference task by considering only interactions which are known to be important in the system. The knowledge of one-to-one interaction between species is much more common in the biological setting than the complex interplay of the regulators which constitutes the regulation function. We thus assume the knowledge of the influence graph and utilise regulator states for the definition of parametric regulatory networks in Chapter 4.

On the other hand, smaller domains of regulation functions allow us to clearly capture independence of the individual variable regulation functions. E.g. if $v \in \{1, \ldots, n\}$ is not a regulator of $u \in \{1, \ldots, n\}$ and vice versa, the variable u is not a regulator of the variable v, the two regulation functions f_v and f_u are independent and can be executed concurrently (the order of their execution is irrelevant). The concurrency in discrete regulatory networks can be exploited for smaller representations of the state space, which is generally exponential in the number of variables. We present such a partial order reduction method in Chapter 6.

2.3 Multivalued Networks

We have presented discrete regulatory networks in the their most generic form. In practice, however, the variables of regulatory networks are commonly an abstraction (discretisation) of a quantitative value, e.g. concentration or production rate of a certain protein [69, 37, 73]. Although continuous models, such as differential equation systems, are generally more suited for quantitative data, the use of discrete abstraction is well justified in the biological setting. As data available for biological systems is sparse, constructing a precise continuous model with all the necessary kinetic parameters is often impossible or requires many design decisions which are ad-hoc by nature. A discrete model requires less parameters whose impact on the system is easier to estimate, making them more suited for the reverse engineering scenario common for systems biology.

In this section we introduce the subclass of discrete regulatory networks suited for the aforementioned task of quantitative variable interpretation, commonly referred to as *multivalued networks*.[2] Multivalued networks are distinguished by having a total order associated with each variable domain. For simplicity, we can assume the variable domains to be downward closed subsets of natural numbers with zero ($X_v = {\downarrow}x$ for some $x \in \mathbb{N}_0$) without loss of generality.

Definition 2.8 (Multivalued Network). A multivalued network F of dimension n is a function $F\colon \prod_{v=1}^{n} {\downarrow}\boldsymbol{m}_v \to \prod_{v=1}^{n} {\downarrow}\boldsymbol{m}_v$ where ${\downarrow}x = \{\, y \in \mathbb{N}_0 \mid y \leq x \,\}$ is the smallest downward closed subset of natural numbers with zero containing x, And $\boldsymbol{m} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n) \in \mathbb{N}^n$ is the vector of maximum values of the individual variables.

To simplify notation, we write $X_{\boldsymbol{m}} = \prod_{v=1}^{n} {\downarrow}\boldsymbol{m}_v$ to denote the state space of multivalued networks.

Multivalued networks allow for all the semantics we introduced for the general discrete regulatory networks (Section 2.1). However, multivalued networks are most often used as a discretisation of a continuous system. To emulate the underlying continuous evolution, we restrict the semantics to only allow the variables to change value stepwise along the total order on their respective domains. In case of subsets of natural numbers, the variables are only allowed to change value by steps of size 1. Note that the restriction has little impact on multivalued networks with asynchronous semantics, as discrete regulatory networks inherently impose no timing information[3]. Replacing one transition

[2] The term multivalued networks was adopted to distinguish them from *Boolean networks* which, although technically a subclass of multivalued networks, are chronologically older as they were used in the first discrete regulatory network application [44].

[3] Modifications exist for models where timing constraints are critical. E.g. models containing both metabolic pathways, where reactions happen within fractions of seconds, and gene regulation, which, being reliant on protein synthesis, takes time in the order of tens of minutes to hours.

changing value by an absolute value of 2 by two transitions changing the value by absolute value of 1 (in the same direction) thus results in the same behaviour.

We only present the necessary modification of generalised asynchronous semantics as it subsumes both synchronous and asynchronous semantics.

Definition 2.9 (Generalised Asynchronous Semantics of Multivalued Networks). Let F be a multivalued network of dimension n and $\boldsymbol{m}$ the vector of maximum values of the individual variables.

The multivalued generalised asynchronous semantics of $F = (f_1, \ldots, f_n)$ is a relation $\xrightarrow[gen]{F} \subseteq X_{\boldsymbol{m}} \times X_{\boldsymbol{m}}$ defined as:

$$(\boldsymbol{x}, \boldsymbol{y}) \in \xrightarrow[gen]{F} \overset{\Delta}{\Longleftrightarrow} \forall v \in D(\boldsymbol{x}, \boldsymbol{y}),\ f_v(\boldsymbol{x}) \geq \boldsymbol{y}_v = \boldsymbol{x}_v + 1 \vee f_v(\boldsymbol{x}) \leq \boldsymbol{y}_v = \boldsymbol{x}_v - 1$$

Having total order on variable domains allows us to express several useful properties in multivalued networks. A prime example of such a property is *monotonicity* of influences (or local monotonicity). As is standard, monotonicity comes in two forms. An influence $(u, v) \in I$ is said to be *positive monotonic* or *activation* if an increase in the value of the regulator u cannot cause a decrease of the value of the target variable v. Similarly, an influence $(u, v) \in I$ is said to be *negative monotonic* or *inhibition* if an increase in the value of the regulator u cannot cause an increase of the value of the target variable v.

Definition 2.10 (Positive Monotonicity). Let F be a multivalued network of dimension n, $\boldsymbol{m}$ the vector of maximum values of the individual variables and let $G = (V, I)$ be an influence graph of F.

An influence $(u, v) \in I$ is positive monotonic (activation) if for any two states $\boldsymbol{x}, \boldsymbol{y} \in X_{\boldsymbol{m}}$ such that $D(\boldsymbol{x}, \boldsymbol{y}) = \{u\}$,

$$(\boldsymbol{x}_u < \boldsymbol{y}_u) \Longrightarrow (f_v(\boldsymbol{x}) \leq f_v(\boldsymbol{y}))$$

u is then a positive regulator (activator) of v.

Definition 2.11 (Negative Monotonicity). Let F be a multivalued network of dimension n, $\boldsymbol{m}$ the vector of maximum values of the individual variables and let $G = (V, I)$ be an influence graph of F.

An influence $(u, v) \in I$ is negative monotonic (inhibition) if for any two states $\boldsymbol{x}, \boldsymbol{y} \in X_{\boldsymbol{m}}$ such that $D(\boldsymbol{x}, \boldsymbol{y}) = \{u\}$,

$$(\boldsymbol{x}_u < \boldsymbol{y}_u) \Longrightarrow (f_v(\boldsymbol{x}) \geq f_v(\boldsymbol{y}))$$

u is then a negative regulator (inhibitor) of v.

Since data on biological systems is sparse, it is often difficult or outright impossible to infer the exact regulation functions. Local monotonicity, however, describes the relationship of only couple variables in isolation, as opposed to the

complex interplay of regulators. Thanks to the relative simplicity, information on influence monotonicity is widely available. Monotonicity is thus a powerful constraint on admissible multivalued networks in place of the exact regulation functions for the purposes of network inference. The exact uses and benefits of influence monotonicity information are explored in Chapter 5.

Due to the difficulty of finding correct regulation functions for multivalued networks, networks with simpler domains are often preferred in practice. In particular, two subclasses of multivalued networks with Boolean vector domain for F are commonly employed, *Boolean networks* and *Thomas networks*.

Boolean Networks

Boolean networks are the most fundamental multivalued networks and as the name suggests, they are characterised by only being composed of Boolean variables, $\forall\, v \in \{1, \ldots, n\},\ X_v = \mathbb{B}$.

Definition 2.12 (Boolean Network). A Boolean network F of dimension n is a function $F\colon \mathbb{B}^n \to \mathbb{B}^n$ on Boolean vectors of length n.

The Boolean domain is the smallest reasonable[4] domain a variable of a multivalued network can have. Boolean networks of dimension n are thus the simplest (smallest) discrete regulatory networks of the dimension n. Although the Boolean domains directly translate to smaller number of possible and thus also reachable states as well as smaller number of regulation states and thus the number of parameters (more in Chapter 4), the reduction is not enough to break out of the combinatorial explosion in either case and the number of both the states and parameters is in the general case exponential in n.

Despite the minimalistic variable domains, Boolean networks exhibit relatively high expressiveness leading to their widespread use [25, 1, 19].

Thomas Networks

Thomas networks allow the use of Boolean regulation functions while maintaining arbitrary (natural) bounds for the individual variable domains. This is achieved by introduction of regulation thresholds in the form of labelling on influences. More precisely, each influence $e = (u, v) \in I$ is assigned a regulation threshold $t(e) \in X_u$. The threshold acts as a binarisation delimiter when feeding the multivalued state to the Boolean regulation function. Any value of the regulator u smaller than $t(e)$ is binarised to 0, while any value of the regulator u larger than or equal to $t(e)$ is treated as 1.

Due to the use of thresholds, Thomas networks are also known as *multivalued threshold networks* or simply *multivalued networks with threshold*. We, however, stick with the traditional nomenclature of Thomas networks, legacy of their first use for modelling cellular regulation by René Thomas [69] as it

[4] Excluding the pathological case of singleton sets.

allows for a clearer distinction between Thomas networks and general multi-valued networks.

Definition 2.13 (Thomas Network). A Thomas network of dimension n is a couple (F, t) where $F\colon \mathbb{B}^n \to X_{\boldsymbol{m}}$ is the regulation function and $t\colon \{1,\dots,n\}^2 \to \mathbb{N}$ is a threshold labelling function such that for any pair of variables $u, v \in \{1,\dots,n\}$, $t((u,v)) \leq \boldsymbol{m}_u$.

Although all discrete regulatory network semantics are applicable to Thomas networks, we modify the standard definition to account for the thresholds. We present only the generalised asynchronous semantics of Thomas networks as it subsumes both the synchronous and asynchronous semantics.

Definition 2.14 (Generalised Asynchronous Thomas Semantics). Let (F, t) be a Thomas network of dimension n and $\boldsymbol{m}$ the vector of maximum values.

The generalised asynchronous semantics of (F, t) is a relation $\xrightarrow[gen]{(F,t)} \subseteq X_{\boldsymbol{m}} \times X_{\boldsymbol{m}}$ defined as:

$$(\boldsymbol{x}, \boldsymbol{y}) \in \xrightarrow{(F,t)} \overset{\Delta}{\iff} \forall\, v \in D(\boldsymbol{x}, \boldsymbol{y}),\ f_v(\boldsymbol{b}_v) \geq \boldsymbol{y}_v = \boldsymbol{x}_v + 1 \vee f_v(\boldsymbol{b}_v) \leq \boldsymbol{y}_v = \boldsymbol{x}_v - 1$$

where for each $v \in \{1,\dots,n\}$, $\boldsymbol{b}_v \in \mathbb{B}^n$ is a binarisation of $\boldsymbol{x}$ according to the relevant thresholds, $\forall\, u \in \{1,\dots,n\},\ \boldsymbol{b}_v[u] = 1 \iff t((u,v)) \leq \boldsymbol{x}_u$.

Thomas networks naturally subsume Boolean networks and offer more expressivity while maintaining the same complexity of the regulation function. The extra expressivity, however, translates into the need to determine the regulation threshold for each influence. The regulation thresholds essentially dictate which variables respond fastest to a monotonic change of the value of a common regulator. In other words, thresholds determine the *sensitivity* of variables to their regulators changing value. Such sensitivity information, however, is seldom readily available in the biological setting.

2.4 Discrete Regultory Networks as Automata Networks

In this section we approach discrete regulatory networks from a different standpoint. In particular, we provide an alternative definition of discrete regulatory networks in the form of automata networks making use of finite automata instead of regulation function to describe the dynamics. Although we speak of alternative definition automata networks may be independently studied as a standalone model [34].

An automata network is a collection of finite automata which take the current state of their neighbours as the input. Automata networks are thus closely related to cellular automata, however, instead of a grid, the topology of an automata network is given by an arbitrary directed graph. In general, automata networks may be infinite. Since the number of automata in an automata

network corresponds to the dimension of the equivalent discrete regulatory network[5], however, we focus on finite automata networks to match the restriction on the dimension of discrete regulatory networks.

Although we introduce automata networks as collections of finite automata, it is usual for automata networks to produce infinite executions. The final states of the individual automata are therefore omitted in the definition. Additionally, since discrete regulatory networks do not have an explicitly defined initial state, we do the same for the automata networks and omit also the initial states of the automata. The notions of final states and initial states may be employed for specification of a concrete, typically reachability, problem.

Definition 2.15 (Automata Network). An automata network $N = (A_1, \ldots, A_n)$ is a collection of simplified finite automata such that for each $i \in \{1, \ldots, n\}$, $A_i = (Q_i, \Sigma_i, \delta_i)$, where $\Sigma_i \subseteq 2^{\bigcup_{j \neq i} Q_j}$ and $\delta_i \colon Q_i \times \Sigma_i \to Q_i$.

Note that the alphabet Σ_i does not necessarily contain the state of every other automaton in the network. Having a restricted alphabet identifies dependencies between the automata (in-neighbours), thus defining the (graph) topology of the automata network.

One can easily find a correspondence between automata networks with n states and discrete regulatory networks of dimension n. The individual automata $A_1, \ldots, A_n$ correspond to the variables of the discrete regulatory network. The states of the automaton are then the elements of the domain of the variable, $Q_i = X_i$, and the transition relation δ_i corresponds to the individual regulation function f_i, $\delta_i(x_i, \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}) = f_i((x_1, \ldots, x_n))$[6]. Following the above correspondence, one can easily apply any of discrete regulatory network semantics for the automata networks.

2.5 Examples

In this section we provide a few examples of the different discrete regulatory networks including the influence graphs and dynamics for different semantics types. As the associated structures, such as the state space graph, tend to grow in size very quickly for larger discrete regulatory networks, we present toy examples rather than actual models from biology. While our toy examples are considerably minimalistic, they suffice to illustrate the important properties in this as well as following chapters, where we repurpose them as running examples.

Example 2.1. *In the following we give an example of a multivalued network F_A with three variables $a(=1), b(=2), c(=3)$. We consider only one variable,*

[5] This is true for networks of deterministic automata. Nondeterministic finite automata in the network should be converted to deterministic automata first as nondeterminism on the level of automata is equivalent to F being a regulation relation instead of a function.

[6] For convenience, we list a value for each automaton in the network in the input of A_i. However, only a subset of the automata (the in-neighbours) has to be considered in general. The automata which are not in-neighbours of A_i have no direct impact on δ_i.

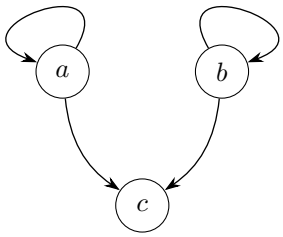


Figure 2.1: The smallest influence graph of the multivalued network F_A.

a, with a domain of size three, $X_a = [0, 2]$. The remaining variables b and c are considered with only Boolean domains, $X_b = X_c = \mathbb{B}$.

$$F_A \colon \boldsymbol{x} \mapsto \left((\boldsymbol{x}_a + 1) \bmod 3, 1 - \boldsymbol{x}_b, \left\lfloor \frac{\boldsymbol{x}_a + \boldsymbol{x}_b}{2} \right\rfloor \right)$$

F_A can be equivalently specified using the individual regulation functions $F_A = (f_a, f_b, f_c)$:

$$\begin{aligned} f_a &\colon \boldsymbol{x} \mapsto (\boldsymbol{x}_a + 1) \bmod 3 \\ f_b &\colon \boldsymbol{x} \mapsto 1 - \boldsymbol{x}_b \\ f_c &\colon \boldsymbol{x} \mapsto \left\lfloor \frac{\boldsymbol{x}_a + \boldsymbol{x}_b}{2} \right\rfloor \end{aligned}$$

Note that according to F_A, respectively f_a, the value of the variable a tends to increase to the maximum $\boldsymbol{m}_a = 2$ and tends to decrease back to 0 once the maximum value is achieved. However, as the multivalued network semantics are restricted to updating variable values by steps of size 1, the value of variable a must decrease to 1 before reaching 0, at which point the tendency changes towards increase. Variable a thus cannot reach the value 0 once it reaches 1 or 2. This is also illustrated in Figure 2.2 and Figure 2.3 showing the state space graph of F_A with synchronous semantics, $\xrightarrow[sync]{F_A}$, and asynchronous semantics, $\xrightarrow[async]{F_A}$, respectively.

The smallest influence graph $G(F_A)$ of F_A is given in Figure 2.1.

The state space graph of F_A with the synchronous semantics in Figure 2.2 features states with simplified notation. Instead of writing the full state notation, e.g. $(0, 1, 0)$ for $\boldsymbol{x}$ with $\boldsymbol{x}_a = \boldsymbol{x}_c = 0$ and $\boldsymbol{x}_b = 1$, we use a shorthand notation with the variable values only, e.g. 010, for simplicity. The same notation is also adopted in the state space graph of F_A with the asynchronous semantics Figure 2.3. Since each asynchronous transition t changes value of a unique variable $v = v(t)$, we additionally annotate the transitions with $v(t)$ and the direction of the value change ($+$ for value increase and $-$ for value decrease) in Figure 2.3.

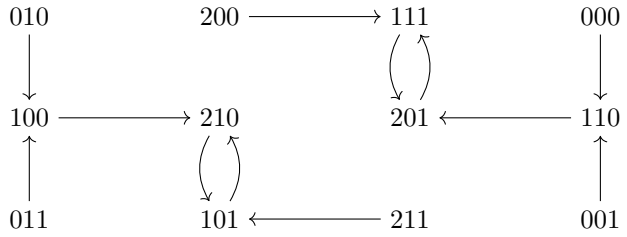


Figure 2.2: The state space of the multivalued network F_A with the synchronous semantics. States are represented by concatenation of variable values in the natural order.

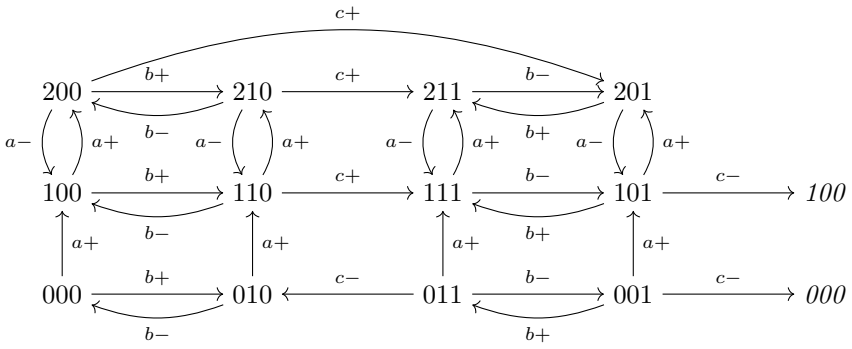


Figure 2.3: The state space of the multivalued network F_A with the asynchronous semantics. States are represented by concatenation of variable values in the natural order and transitions are annotated by the unique variable changing value ($v(t)$) and the direction of the change.

Observe that the choice of semantics has a significant impact on the dynamics of the regulatory network. In the case of F_A, the difference is mainly caused by the variables a and b being regulated only by themselves, $R(a) = \{a\}$ and $R(b) = \{b\}$. Thus, while in the asynchronous semantics either variable a or variable b is allowed to change value independently of each other, their value updates are synchronised under the synchronous semantics. Such synchronisation is responsible for the state space graph in Figure 2.2 being split into two connected components. One component drains into the loop $101 \leftrightarrow 210$ *(a and b change value in the same direction) while the other drains into the loop* $111 \leftrightarrow 201$ *(a and b change value in the opposite direction).*

Example 2.2.

$$F_B : \boldsymbol{x} \mapsto ((\boldsymbol{x}_b \vee \boldsymbol{x}_c) \wedge \neg(\boldsymbol{x}_d), \neg(\boldsymbol{x}_b), \boldsymbol{x}_b, \neg(\boldsymbol{x}_d))$$

F_B is an example of a Boolean network with four variables $a(=1), b(=2), c(=3), d(=4)$. The Boolean network F_B can be equivalently expressed using the

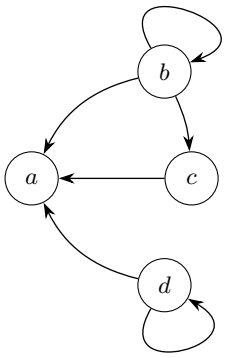


Figure 2.4: The smallest influence graph of the Boolean network F_B.

individual regulation functions f_a, f_b, f_c, f_d*:*

$$f_a \colon \boldsymbol{x} \mapsto (\boldsymbol{x}_b \vee \boldsymbol{x}_c) \wedge \neg (\boldsymbol{x}_d)$$
$$f_b \colon \boldsymbol{x} \mapsto \neg (\boldsymbol{x}_b)$$
$$f_c \colon \boldsymbol{x} \mapsto \boldsymbol{x}_b$$
$$f_d \colon \boldsymbol{x} \mapsto \neg (\boldsymbol{x}_d)$$

Since all variable domains are Boolean, the regulation functions can be specified within the Boolean algebra.

The smallest influence graph of F_B *is given in Figure 2.4. While variables* b, c, d *only depend on a single regulator (*$|R(b)| = |R(c)| = |R(d)| = 1$*), the value of variable* a *evolves according to* 3 *regulators (*$R(a) = \{b, c, d\}$*).*

Figure 2.5 and Figure 2.6 illustrate the state space graphs of F_B *with synchronous semantics* $\xrightarrow[sync]{F_B}$ *and asynchronous semantics* $\xrightarrow[async]{F_B}$*, respectively. The sole regulators of variables* b *and* d *are the variables themselves,* $R(b) = \{b\}$*,* $R(d) = \{d\}$*. Moreover, since the variables* b *and* d *have negative monotonic influence on themselves, they are frustrated in each state,* $f_b(\boldsymbol{x}) \neq \boldsymbol{x}_b$ *and* $f_d(\boldsymbol{x}) \neq \boldsymbol{x}_d$ *for any state* $\boldsymbol{x}$*. The variables* b *and* d *wanting to change value in any state leads to significant number of transitions in the asynchronous semantics. The transitions updating the value of the variables* b *and* d *are thus abstracted in Figure 2.6 to improve readability.*

Observe that similarly to Example 2.1, the choice of semantics has a significant impact on the dynamics of the network. In particular, while the state space graph of F_B *is a single connected component under asynchronous semantics, it consists of several components for the synchronous case. This can be partially attributed to the synchronisation of the variable* b *and* d *value updates disallowing transitions from states where the variables* b *and* d *have the same value to states in which their values differ and vice versa.*

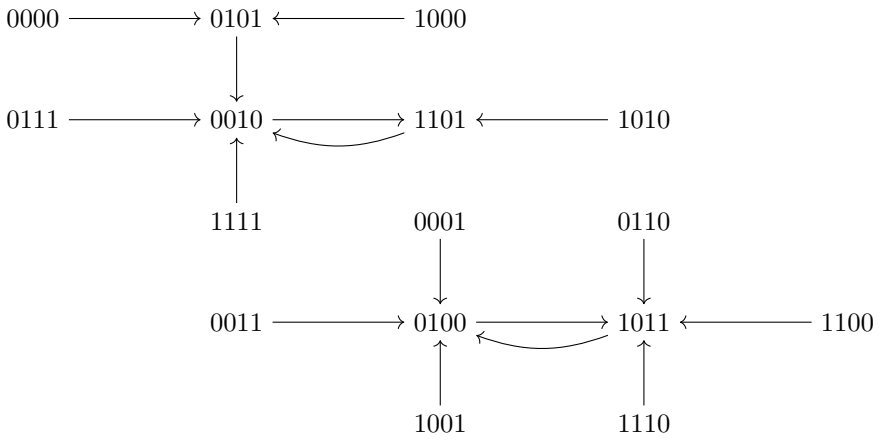


Figure 2.5: The state space graph of the Boolean network F_B with synchronous semantics. States are represented by concatenation of variable values in the natural order.

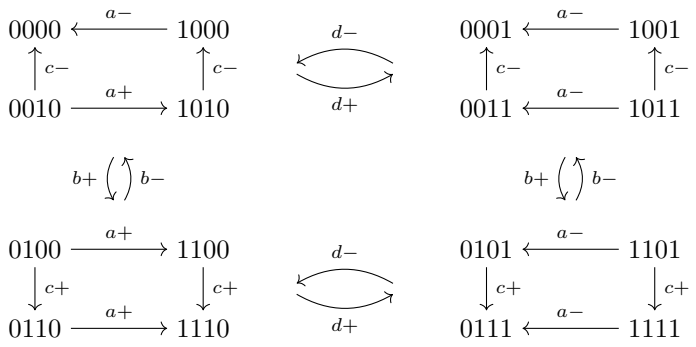


Figure 2.6: The state space graph of the Boolean network F_B with asynchronous semantics. States are represented by concatenation of variable values in the natural order and transitions are annotated by the unique variable changing value ($v(t)$) and the direction of the change. The transitions updating the value of variables b and d are given only schematically for the sake of clarity.

Example 2.3. *Consider the influence graph $G(F_A) = (V, I)$ (Figure 2.1) from Example 2.1 and a threshold function $t\colon I \to [0, 2]$ defined as:*

$$t\colon (a, a) \mapsto 2$$
$$t\colon (b, b) \mapsto 1$$
$$t\colon (a, c) \mapsto 2$$
$$t\colon (b, c) \mapsto 1$$

We use the threshold function t to define a Thomas network (F_C, t) on the same variables a, b and c with the same domains, $X_a = [0, 2]$, $X_b = X_c = \mathbb{B}$, and the same smallest influence graph $G(F_C) = G(F_A)$ as the multivalued network F_A from Example 2.1.

$$F_C\colon \boldsymbol{b} \mapsto \left(2\left(1 - \boldsymbol{b}_a\right), \neg\left(\boldsymbol{b}_b\right), \boldsymbol{b}_a \vee \boldsymbol{b}_b\right)$$

Or equivalently using the individual regulation functions f_a, f_b, f_c:

$$f_a\colon \boldsymbol{b} \mapsto 2\left(1 - \boldsymbol{x}_a\right)$$
$$f_b\colon \boldsymbol{b} \mapsto \neg\left(\boldsymbol{x}\right)_b$$
$$f_c\colon \boldsymbol{b} \mapsto \boldsymbol{x}_a \vee \boldsymbol{x}_b$$

The states depend on the variable domains and are thus identical for (F_C, t) and F_A, the regulatory function itself, however, is defined on Boolean vectors for Thomas networks, $F_C\colon \mathbb{B}^3 \to [0, 2] \times \mathbb{B}^2$. While the simplification to the Boolean domain is often desired, the resulting regulatory function is not as expressive as the original F_A.

Indeed, one may note that although the regulatory function F_C has been constructed to emulate F_A, there are differences in the dynamics of (F_C, t) and F_A. In particular, the value of variable c tends to decrease when $\boldsymbol{x}_a = 0$ regardless of the value of $\boldsymbol{x}_b$ under F_A. In the case of (F_C, t), however, the value of variable c tends to increase when $\boldsymbol{x}_b = 1$ regardless of the value of $\boldsymbol{x}_a$. This is also illustrated in the state space graph of (F_C, t) in Figure 2.7.

While the difference on the target value of variable c in state $(0, 1, 0)$, respectively $(0, 1, 1)$, can be amended by a different Thomas network, the exact behaviour of F_A cannot be replicated by any Thomas network with the same variables and influence graph. This follows from the fact that the impact of variable b on variable c is different for each possible value of variable a (see Table 2.1).

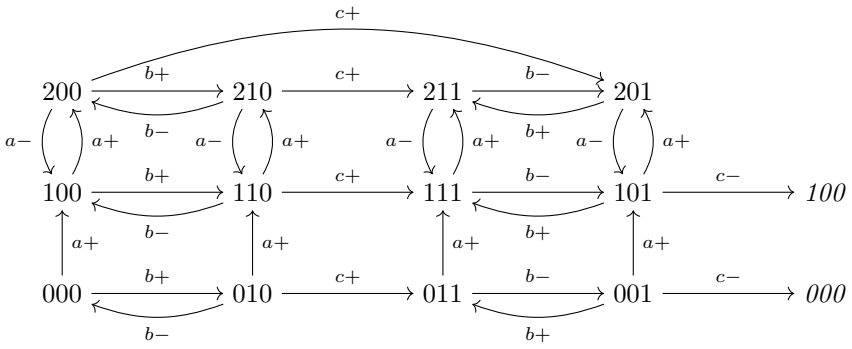


Figure 2.7: The state space of the Thomas network (F_C, t) with the asynchronous semantics. States are represented by concatenation of variable values in the natural order and transitions are annotated by the unique variable changing value ($v(t)$) and the direction of the change.

$\boldsymbol{x}_a$	0	0	1	1	2	2
$\boldsymbol{x}_b$	0	1	0	1	0	1
$f_c(\boldsymbol{x})$	0	0	0	1	1	1

Table 2.1: The table of target values of variable c (f_c) in the multivalued network F_A for all possible combinations of variable a and b values. Note that different target values are observed for all three values of variable a.

Chapter 3

Partial Order Semantics of Transition System Products

The state space of discrete regulatory networks is in the general case exponential in the dimension (number of variables). As such, discrete regulatory network analysis more often than not suffers from combinatorial explosion. Gene regulatory networks and other biological models that constitute our primary application domain, however, are generally sparsely connected, leading to a high degree of concurrency. For this reason, it is natural to consider partial order semantics for discrete regulatory networks, in order to benefit from a more compact representation of the state space.

A well established and extensively studied partial order semantics for transition systems exists for Petri nets in the form of unfoldings, or more precisely *branching processes*. Thanks to their high expressivity, the unfolding semantics of Petri nets can be used for any general transition system, including discrete regulatory networks [30]. Petri net unfolding therefore appears to be the ideal partial order semantics candidate to study in relation to discrete regulatory networks, as well as their parametric extension introduced in Chapter 4.

3.1 Petri Nets

Prior to starting on the unfolding itself, we briefly recall the definition of a Petri net. A Petri net is a directed bipartite graph between places and transitions. Each place can hold any natural number of tokens. *Marking* is a function specifying the number of tokens for each place. A transition can fire (is enabled) if a token is present in all places in the set of its in-neighbours (*preset*). Firing a transition consumes a token from each place in the preset and produces a token in each place in the set of out-neighbours (*poset*) of the transition[1] thus

[1] In the general case, the arcs of a Petri net may have labels specifying the number of tokens consumed/produced by a transition. The same quantities of tokens then apply to both enabling and firing of the transition. Within our work, however, we assume all arcs to be unlabelled (labelled with 1).

reaching a new marking.

In this work we limit ourselves to safe Petri nets. A Petri net is safe (or 1-safe) if any place has at most one token at a time in any reachable marking. The limitation is well justified as domains of all the variables of discrete regulatory networks are guaranteed to be finite. A safe Petri can thus emulate a Discrete regulatory network by representing each value of each variable by a unique place. Similarly, we consider only finite Petri nets as the number of variables in discrete regulatory networks is also finite. Observe that this representation closely resembles the automata networks.

Definition 3.1 (Petri Net). A (1-safe) Petri net is a tuple (P, T, W, M), where $P \cap T = \emptyset$ are finite sets of places and transition, respectively, $W \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (edges) between places and transitions and $M \colon P \to \mathbb{B}$ is the initial marking.

For each node $x \in P \cup T$ we write ${}^\bullet x = \{ y \in P \cup T \mid (y, x) \in W \}$ to denote the preset of x and $x^\bullet = \{ y \in P \cup T \mid (x, y) \in W \}$ to denote the poset of x.

Note that for safe Petri nets, a marking M can be easily represented as a set of the places which contain a token, $M = \{ p \in P \mid M(p) = 1 \}$. Remark further that $(P \cup T, W)$ is a directed bipartite graph between places and transitions by definition and we employ it as such. Finally, we omit the initial marking from the definition of a Petri net where convenient. When omitted, we consider the Petri net with any initial marking which supports the safeness criteria.

3.2 Unfolding

Petri net unfolding relies on *branching processes*, a partial order semantics of Petri nets. Intuitively, a Petri net unfolding is similar to an unfolding of a graph. Given an initial vertex, any directed graph can be unfolded into a tree whose nodes represent the paths leading from the initial vertex. Petri nets can also be unfolded into labelled *occurrence nets*, a subclass of Petri nets benefitting from simple structure similar to trees. The resulting occurrence net is called a branching process. The nodes of the branching process are labelled with places and transitions of the original Petri net. Unfolding of a Petri net may be stopped at any time, yielding many different branching processes. The Petri net *unfolding* refers to the maximal (generally infinite) branching process which unfolds as much as possible.

Petri net unfolding differs from the unfolding of the state space graph of the underlying transition systems. The Petri net transitions are local – they affect only a subset of the places of the Petri net, allowing for two or more transition to act independently of each other. Such transitions are called *concurrent*. By exploiting *concurrency*, Petri net unfolding results in a more compact structure. This is achieved by allowing "merging" concurrent branches within the unfolding. Petri net unfoldings therefore avoid representing equivalent branches twice at the expense of the tree structure. We give the formal notion of concurrency below, using the *causal* and *conflict* relations on the nodes of a Petri net.

Definition 3.2 (Causal Relation). Let $x, y \in P \cup T$ be two nodes of the Petri net (P, T, W, M_0).

x and y are in causal relation, $x < y$, if there exists a directed path from x to y in the graph $(P \cup T, W)$.

Definition 3.3 (Conflict Relation). Let $x, y \in P \cup T$ be two nodes of the Petri net (P, T, W, M_0).

x and y are in conflict relation, $x \# y$, if there exist two paths $(a, t_0, \ldots, x)$ and $(a, t_1, \ldots, y)$ in the graph $(P \cup T, W)$ leading to x and y, respectively, which start from the same place and subsequently diverge.

Definition 3.4 (Concurrency Relation). Let $x, y \in P \cup T$ be two nodes of the Petri net (P, T, W, M_0).

x and y are in concurrency relation, $x \parallel y$, if $\neg(x < y)$, $\neg(y < x)$ and $\neg(x \# y)$.

As the causal, conflict and concurrency relations suggest, occurrence nets are indeed event structures. We adapt the usual notation and call the places of an occurrence net *conditions* (the b notation comes from Petri's original 'Bedingungen') and the transitions of an occurrence net *events*.

Definition 3.5 (Occurrence Net). An occurrence net $O = (B, E, F)$ is a Petri net such that:

1. $\forall b \in B,\ |{}^\bullet b| \leq 1$;
2. The causal relation is a partial order (O is acyclic);
3. For every $x \in B \cup E$, the set $\{ y \in B \cup E \mid y < x \}$ is finite (O is finitely preceded);
4. $\forall b \in B,\ \neg(b \# b)$ (O is conflict-free).

As aforementioned, a branching process is an occurrence net whose conditions and events are labelled by the places and transitions, respectively, of the original Petri net they represent. Unlike an occurrence net, the branching process is a "proper" Petri net, including the initial marking. To ease the notation, we override the min function for occurrence nets to produce the causality-minimal subsets of nodes, $\min\colon (B, E, F) \mapsto \{ x \in B \cup E \mid \forall y \in B \cup E,\ x \leq y \}$.

Definition 3.6 (Branching Process). A branching process of a Petri net (P, T, W, M_0) is an occurrence net O labelled with function $\beta\colon B \cup E \to P \cup T$ such that:

1. $\beta(B) \subseteq P$ and $\beta(E) \subseteq T$ (β preserves the nature of nodes).
2. Given an arbitrary event $e \in E$. $\forall b \in {}^\bullet e,\ \beta(b) \in {}^\bullet\beta(e)$ and vice versa, $\forall p \in {}^\bullet\beta(e)$, there exists a unique $b = \beta^{-1}(p)$ and $b \in {}^\bullet e$. (β restricted to ${}^\bullet e$ is a bijection.) Similarly, $\forall b \in e^\bullet,\ \beta(b) \in \beta(e)^\bullet$ and vice versa, $\forall p \in \beta(e)^\bullet,\ b = \beta^{-1}(p)$ is unique and $b \in e^\bullet$. (β restricted to $e^\bullet$ is a bijection.)

3. $\forall b \in \min((O)),\ \beta(b) \in M_0$ and $\forall p \in M_0,\ b = \beta^{-1}(p)$ is unique and $b \in \min((O))$. (β restricted to causality-minimal conditions is a bijection with the initial marking.)

4. $\forall e_0, e_1 \in E,\ {}^\bullet e_0 = {}^\bullet e_1 \wedge \beta(e_0) = \beta(e_1) \Longrightarrow e_0 = e_1$ (No duplicate transitions).

$\min((O))$ is the initial marking of the branching process.

Many (possibly infinitely many) different branching processes may be constructed for a single Petri net. However, all the branching processes are constructed by the same process, unfolding, differing essentially on 'how much' are they unfolded. Thanks to the acyclic structure of occurrence nets and thus branching processes, everything captured by a branching process which has been unfolded less (to a lesser depth) is also captured by a larger, 'more unfolded', branching process (with larger depth). The smaller, 'less unfolded', branching process can thus be called a *prefix* of the larger branching process.

Definition 3.7 (Branching Process Prefix). Let (O, β) and (O', β') be two branching processes of the same Petri net (P, T, W, M).

Then, (O, β) is a prefix of (O', β') if the following conditions are satisfied:

1. $B \subseteq B'$, $E \subseteq E'$ and $F \subseteq F'$ (O is a subnet of O');

2. $\min((O')) \subseteq B$ (The natural initial marking is the same for both branching processes);

3. For each condition $b \in B$ and the single event $e \in E'$ such that $e \in {}^\bullet b$ (if it exists), $e \in E$;

4. Similarly, for each event $e \in E$ and each condition $b \in B'$ such that $b \in {}^\bullet e \cup e^\bullet$, $b \in B$;

5. For each $x \in B \cup E$, $\beta(x) = \beta'(x)$ (β is the restriction of β' to O).

The notion of prefixes gives a partial order structure on branching processes allowing us to formally capture the notion of 'unfolding more'. In [29], it has been shown that there exists a unique maximal branching process of a Petri net, up to isomorphism. The maximal branching process is known as the Petri net unfolding.

Definition 3.8 (Petri Net Unfolding). Let (P, T, W, M) be a Petri net.

The unfolding of (P, T, W, M) is a branching process (O, β) of (P, T, W, M) such that any other branching process (O', β') of (P, T, W, M) is a prefix of (O, β).

The unfolding of a Petri net is unique up to isomorphism.

3.3 Behavioural Equivalence

Although the unfolding of a Petri net is in the general case infinite, it gives a complete acyclic representation of the behaviour of the original Petri net. The behavioural equivalence is captured by the isomorphism of (graph) unfoldings of the reachability graphs of a Petri net and its unfolding. In particular, the set of reachable markings of a Petri net contains exactly the markings obtained as $\beta(M)$ where M is reachable in the unfolding of the Petri net. Moreover, the possible firing sequences of the Petri net are also reproduced by the unfolding. Given a marking M reachable in the unfolding, another marking M' and a transition $t \in T$ of the original Petri net, An event $e \in E$ such that $M \xrightarrow{e} M'$ and $\beta(e) = t$ exists if and only if $\beta(M) \xrightarrow{t} \beta(M')$.

To capture the notion of behavioural equivalence of Petri nets and their branching processes formally, we introduce the concepts of *configuration* and *cut*, which represent the firing sequence and marking of the original Petri net, respectively, within the branching process.

Definition 3.9 (Configuration). A configuration of a branching process (O, β) is a set of events $C \subseteq E$ such that:

1. $e \in C \Longrightarrow \forall e' < e,\ e' \in C$ (C is causally closed);
2. $\forall e, e' \in C,\ \neg(e \# e')$ (C is conflict-free).

Configuration is a set of events which can be fired within one run of the branching process. More precisely, for each configuration there exists a firing sequence which allows each event in the configuration to execute exactly once starting from the natural initial state and respecting the causal relation. These firing sequences mirror the firing sequences of the original Petri net through the labelling function β.

Thanks to being conflict free and causally closed, firing each event in a configuration results in a set of concurrent conditions (*coset*). Since we assume our input Petri net to be 1-safe, any two conditions b, b' of a branching process such that $\beta(b) = \beta(b')$ are either in causal relation or conflict. As such, β restricted to any coset of the branching process is injective. A maximal coset of a branching process is called a cut and corresponds to a marking of the original Petri net when projected via β. Of particular interest are then cuts produced by configurations, as they represent the reachable markings.

Definition 3.10 (Cut). A set of conditions $\gamma \subseteq B$ is a cut, if $\forall b, b' \in \gamma,\ b \parallel b'$ and $\forall b \in B \setminus \gamma,\ \exists b' \in \gamma,\ \neg(b \parallel b')$.

Given a finite configuration C, the set of conditions obtained by executing every event in C from the initial state is a cut of the following form $Cut(C) = (\min((O)) \cup C^\bullet) \setminus {}^\bullet C$.

We have already stated, in less formal terms, that a marking M is reachable in the original Petri net if and only if the unfolding of said Petri net contains a

configuration C such that $\beta(Cut(C)) = M$. Petri net unfoldings therefore offer a comprehensive way to study the reachable state space of transition systems in a concurrency aware environment. This property is captured in the notion of *completeness* of branching processes.

Definition 3.11 (Complete Branching Process). Let (P, T, W, M_0) be a Petri net.

We say that a branching process (O, β) of (P, T, W, M_0) is complete, if for every marking M reachable in (P, T, W, M_0), there exists a configuration C in (O, β), satisfying:

1. $\beta(Cut(C)) = M$ (M is represented in the branching process);
2. For every transition $t \in T$ enabled in M, there exists an event $e \in E \setminus C$ such that $\beta(e) = t$ and $C \cup \{e\}$ is a configuration of (O, β) (all enabled transitions can be reproduced in the branching process).

3.4 Complete Finite Prefix

Unfolding of a Petri net is trivially a complete branching process, however, the applicability of Petri net unfoldings is largely impeded by the unfoldings being infinite in the general case. As the reachable state space of a finite (safe) Petri net is also finite, envisioning a branching process which is both complete and finite is far from absurd. A technique to construct such a representation, called a *complete finite prefix* of the unfolding, has been introduced by Kenneth L. McMillan et al. [58] and later improved by Javier Esparza et al. [31], including an upper bound on the size of the constructed prefix.

The construction of the complete finite prefix is based on identifying events, called *cut-off* events, at which the unfolding procedure can be stopped while guaranteeing the resulting branching process is complete. Determining whether an event can be declared a *cut-off* or not relies on a key observation about the isomorphism of extensions of configurations whose cuts have the same projection via β.

More precisely, let (O, β) be a branching process of a Petri net (P, T, W, M) with a configuration C. We write $\Uparrow C = (O', \beta')$ to denote the part of (O, β) which 'comes after' C. Formally, $O' = (B', E', F')$ where $B' = \{b \in B \setminus {}^\bullet C \mid \forall\, e \in C,\ \neg(b \# e)\}$, $E' = \{\, e \in E \setminus C \mid \forall\, e' \in C,\ \neg(e \# e')\,\}$ and F', β' are restrictions of F and β, respectively, to $B' \cup E'$. Then, (O', β') is a branching process of the Petri net $(P, T, W, \beta(Cut(C)))$. In particular, if (O, β) is the unfolding of the Petri net (P, T, W, M), (O', β') is the unfolding of $(P, T, W, \beta(Cut(C)))$. Thus, as the unfolding is unique up to isomorphism, for any two configurations C and C' such that $\beta(Cut(C)) = \beta(Cut(C'))$, we have $\Uparrow C = I(\Uparrow C')$ where I is the isomorphism of the unfoldings.

We therefore have an isomorphism of configuration extensions provided the configurations reach the same marking (projected via β). This very isomorphism is exploited in the construction of the complete finite prefix to determine

which branches of the unfolding can be omitted without loss of completeness. To be able to apply the reasoning about isomorphic extensions directly to events, McMillan et al. [58] associate a *local configuration* to each event. A local configuration of an event e is the minimal configuration which allows e to fire.

Definition 3.12 (Local Configuration). Let (O, β) be a branching process with $O = (B, E, F)$ and let $e \in E$ be an arbitrary event.

The local configuration $[e] \subseteq E$ of event e is the downward closure, with respect to the causality relation, of the singleton set $\{e\}$ within the set of events E, $[e] = \{\, e' \in E \mid e' \leq e \,\}$.

The local configuration of any event $e \in E$ is trivially a configuration of (O, β).

Any configuration C containing an event e is necessarily an extension of the local configuration, $e \in C \Longrightarrow [e] \subseteq C$. Thus, shall there exist a different event e' with $\beta\,(\mathit{Cut}\,([e])) = \beta\,(\mathit{Cut}\,([e']))$, there must exist an extension of $[e']$ isomorphic to C, or more precisely to $C \setminus [e]$.

The unfolding procedure detailed in [58, 31] constructs the branching process by including individual events and their posets one-by-one, selecting them from a set of *possible extensions*. A possible extension is an event of the unfolding whose preset is already included in the prefix constructed thus far and which can therefore be included in the prefix with no further modification necessary.

Definition 3.13 (Possible Extension). Let (O, β) be the unfolding of a Petri net (P, T, W, M) and let (O', β') be a prefix of the unfolding.

An event $e \in E \setminus E'$ is a possible extension of the prefix (O', β') if and only if $\forall\, b \in {}^\bullet e \subseteq B,\ b \in B'$.

The set of all possible extensions of a prefix (O', β') is denoted by $PE\,((O', \beta'))$.

We now have all the tools needed to define the cut-off criterion for events, which determines when to stop the unfolding procedure.

Definition 3.14 (Cut-Off Event). Let (O, β) be a finite branching process of a Petri net (P, T, W, M) and let (O', β') be another branching process of the same Petri net such that $B' = B \cup e^\bullet$ and $E' = E \cup \{e\}$, where $e \in PE\,((O, \beta))$ is a possible extension of the branching process (O, β).

Then the event e is a cut-off in (O', β'), $e \in \mathit{cutoffs}\,((O', \beta'))$, if and only if there exists an event $e' \in E$ such that $\beta\,(\mathit{Cut}\,([e])) = \beta\,(\mathit{Cut}\,([e']))$.

Notice that the definition of a cut-off event depends heavily on the possible extension chosen. More precisely, which events are marked cut-off depends on the order in which the unfolding procedure adds events into the constructed branching process. The influence of the order in which potential extensions are chosen is significant enough to separate between complete and incomplete finite prefixes while using the same cut-off criterion.

Esparza et al. [31] identify a class of *adequate orders* on configurations of the branching process and prove that including events in an order that aligns with an adequate order on the local configurations guarantees the produced finite prefix to be complete. A concrete example of an adequate order based on counting transitions of the configuration (as projected via β) in a manner similar to Parikh vectors is also provided in [31].

Definition 3.15 (Adequate Order). Let (O, β) be the unfolding of a Petri net (P, T, W, M).

A (partial) order $\prec$ on the configurations of the unfolding (O, β) is an adequate order if it satisfies the following requirements:

1. $\prec$ is well founded.
2. Given two configurations C_0, C_1, $C_0 \subset C_1 \Longrightarrow C_0 \prec C_1$.
3. Given two configurations C_0, C_1 with $\beta\left(Cut\left(C_0\right)\right) = \beta\left(Cut\left(C_1\right)\right)$ and $C_0 \prec C_1$, for any extension $C' \subseteq E \backslash C_0$ of the configuration C_0, $C_0 \cup C' \prec C_1 \cup I\left(C'\right)$ where I is the isomorphism of the extensions of C_0 and C_1 (unfoldings of $(P, T, W, \beta\left(Cut\left(C_0\right)\right))$, respectively, $(P, T, W, \beta\left(Cut\left(C_1\right)\right))$).

Esparza et al. [31] further prove that if the used adequate order is also a total order, the number of non cut-off events in the constructed complete finite prefix is bounded by the number of reachable states of the original Petri net. This result is underlined by an example of a total adequate order, which being an extension of the partial adequate order example again counts transitions, however, does so per the causal layers of the configuration, utilising the Foata normal form.

The complete finite prefix is thus guaranteed to not be (asymptotically) larger than the reachable state space graph. Moreover, by keeping track of the concurrency in the system, unfoldings and their complete finite prefixes benefit from partial order reduction. The complete finite prefix of a highly concurrent Petri net is therefore a very compact way to represent the reachable state space.

Part II

Theoretical Contributions

Chapter 4

Parametric Regulatory Networks

In this chapter we introduce parametric regulatory networks, a parametric version of discrete regulatory networks where the exact regulation function is unknown – represented by parameters.

Parametric regulatory networks are motivated by the lack of precise information on the complex regulations of genes and other molecular interaction in biological systems. Parametric regulatory networks aim at extracting the available regulation information while making no assumption on the unknown interactions. This is achieved by the use of parameters to represent the target values of the individual regulation functions, where unknown.

Standing for a target value of an individual regulation function, a parameter $K_{v,\boldsymbol{x}} \in X_v$ represents the value variable v takes in state $\boldsymbol{x}$ ($f_v(\boldsymbol{x})$). Having a parameter for each variable and each state thus allows us to completely parametrise the regulation function, yielding a parametric regulatory network.

The parameters, as illustrated above, depend on the states of the network. It is thus easy to see that the total number of parameters is exponential in the number of variables, possibly limiting the tractability of parametric regulatory network analysis. However, each state and variable combination has to be used only if no prior knowledge is considered. Not only would such construction yield an impractical network, as mentioned in Section 2.2, the knowledge of one-on-one interactions between biological species is far more widespread than the information on the complex regulation. It is therefore reasonable to consider an influence graph to be part of the input.

Having an influence graph allows us to restrict the parameters to the regulator states of individual variables without loss of information, much like we illustrated for regulation functions of discrete regulatory networks in Section 2.2. We therefore consider parameters of the form $K_{v,\boldsymbol{\omega}} \in X_v$ to represent the target value of variable v in the regulator state $\boldsymbol{\omega} \in \Omega_v$. Note that using regulator states does not improve on the number of parameters asymptotically, as the number of regulator states is still exponential in the number of variables in the general case. The size of the exponent, however, depends on the number of regulators of individual variables rather than the total number of variables.

In practice, this allows sparse influence graphs to exponentially decrease the number of regulator states, and by extension parameters, compared to the theoretical maximum.

Similarly to a discrete regulatory network, which can be defined by a single function, parametric regulatory network is fully captured by the parameters that build up said function. Unlike for discrete regulatory networks, however, where the variable domains are inherently captured within the domain of the function, we have to explicitly specify the variable domains, or parameter ranges, for parametric regulatory networks. Instead of listing all the parameters in the definition, we can simply utilise the influence graph, which by itself fully characterises all the regulator states and thus the parameters.

Definition 4.1 (Parametric Regulatory Network). A parametric regulatory network G_d of dimension n is a directed graph $G = (V, I)$ such that $n = |V|$ coupled with a function $d\colon v \mapsto X_v$ mapping each variable (vertex) $v \in V$ to the corresponding domain X_v.

Definition 4.1 captures the most generic parametric regulatory network, corresponding to the most generic discrete regulatory network in Definition 2.1. All the restrictions we discussed for discrete regulatory networks in Section 2.3, whether being a restriction on the variable domains (Boolean and multivalued networks) or also on the regulation function (Thomas networks) can be lifted to the parametric regulatory networks.

In here we focus on the multivalued restriction only, as it subsumes the Boolean networks and is a prerequisite for the Thomas networks. As the variable domains of a multivalued network must come with a total order, they can always be represented as intervals of natural numbers (with zero), ranging from zero to a given maximum.

Definition 4.2 (Parametric Multivalued Network). A multivalued parametric regulatory network $G_{\boldsymbol{m}}$ of dimension n is a directed graph $G = (V, I)$ such that $n = |V|$ coupled with a vector $\boldsymbol{m} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n) \in \mathbb{N}^n$ of maximum values for each variable.

An example of a parametric multivalued network is given in Example 4.1.

One may observe that Definition 4.2 does not need any further modifications to work for parametric Thomas networks as the restriction to Boolean regulatory states is pushed to the inference of the regulator states from the influence graph.

In further work, we focus almost entirely on the modelling of gene regulatory networks. As the values of variables in a model gene regulatory network most commonly represent the concentration level of a protein or another molecule within the cell, models of gene regulatory networks are a prime example of networks whose variables are discrete abstractions of real-valued measurements. As mentioned in Section 2.3, the multivalued restriction of discrete regulatory networks, and in turn of parametric regulatory networks is highly suitable for these types of models. We will thus limit ourselves to parametric multivalued

networks when referring to parametric regulatory networks in further text, unless explicitly states otherwise.

4.1 Parametrisations

Semantics of parametric regulatory networks are closely tied to the semantics of discrete regulatory networks. To be able to give a formal definition of parametric regulatory network semantics, we first have to formalise the connection between parametric and discrete networks. Intuitively, by replacing the target values of the regulation function by parameters, parametric regulatory networks obtain freedom to choose between multiple different regulation functions to use. As such, a parametric regulatory network can be understood as a set of discrete regulatory networks sharing common topological properties (influence graph and variable domains). To capture this relationship formally, we employ the concept of a *parametrisation*.

A parametrisation is essentially a function assigning each parameter a concrete value from the associated variable domain. For practicality of presentation, however, we prefer to think of a parametrisation as a vector of the values of individual parameters. To this end we assume an arbitrary but fixed total order $\blacktriangleleft \subseteq \Omega_v \times \Omega_v$ on the regulator states of each variable $v \in V$. Such an order is guaranteed to exist for each variable as there is always only finitely many regulator states, lexicographic order is a natural example. We then use the total order $\trianglelefteq$ on regulator states in combination with the natural total order $\trianglelefteq$ on variables, first utilised in Section 2.2 to represent regulator states as vectors, to obtain a total order on parameters $\preceq$ of a parametric regulatory network:

$$K_{v,\boldsymbol{\omega}} \preceq K_{w,\boldsymbol{\omega}'} \overset{\Delta}{\iff} (v \triangleleft w) \vee ((v = w) \wedge (\boldsymbol{\omega} \trianglelefteq \boldsymbol{\omega}'))$$

Using $\preceq$ as a fixed order on parameters, definition of a parametrisations as vectors becomes straightforward.

Definition 4.3 (Parametrisation). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network.

Then, a parametrisation of $G_{\boldsymbol{m}}$ is a vector $\boldsymbol{P}$ of length $|\Omega(G)|$ and of the following form:

$$\boldsymbol{P} \in \overset{\preceq}{\prod_{(\boldsymbol{\omega},v)\in\Omega(G)}} \{0,\ldots,\boldsymbol{m}_v\}$$

We further use $\mathbb{P}(G_{\boldsymbol{m}}) = \overset{\preceq}{\prod_{(\boldsymbol{\omega},v)\in\Omega(G)}} \{0,\ldots,\boldsymbol{m}_v\}$ to denote the set of all parametrisations of the parametric regulatory network $G_{\boldsymbol{m}}$.

Finally, to ease notation, we write simply $\boldsymbol{P}_{v,\boldsymbol{\omega}}$ instead of $\boldsymbol{P}_{K_{v,\boldsymbol{\omega}}}$ to denote the value of parameter $K_{v,\boldsymbol{\omega}}$ in parametrisation $\boldsymbol{P}$.

The parameters stand for the values of the regulation function at individual inputs. Thus, by specifying a concrete value for each parameter, a parametrisation effectively describes a full regulation function of a discrete regulatory

network. A parametric regulatory network overlaid by a parametrisation thus defines a single discrete regulatory network, or *parametrised network.*

Definition 4.4 (Parametrised Network). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network of dimension n and let $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$ be a parametrisation of $G_{\boldsymbol{m}}$.

Then, the multivalued network $F_{\boldsymbol{P}} = (f_1, \ldots, f_n)$ with

$$f_v \colon \Omega_v \to \{1, \ldots, \boldsymbol{m}_v\}$$
$$f_v \colon \boldsymbol{\omega} \mapsto \boldsymbol{P}_{v,\boldsymbol{\omega}}$$

for variable each $v \in \{1, \ldots, n\}$, is the parametric regulatory network $G_{\boldsymbol{m}}$ parametrised by $\boldsymbol{P}$.

We can now expand on the previous intuition of parametric regulatory networks being sets of discrete regulatory networks. Although a simple union is unsatisfactory, as is elaborated in Section 4.2 where the formal definition of parametric regulatory network semantics is given, the semantics of parametric regulatory networks may be understood as an aggregation over the semantics of all the parametrised networks $F_{\boldsymbol{P}}$ of all the parametrisations $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$.

Example 4.1. *Let $G = G(F_A)$ be the minimal influence graph of the multivalued network F_A from Example 2.1 and let $\boldsymbol{m} = (2, 1, 1)$. $G_{\boldsymbol{m}}$ is then the parametric regulatory network with the same influence graph and variable domains as the multivalued network F_A.*

The parametrisations of $G_{\boldsymbol{m}}$ assign a value to each regulator state in $\Omega(G_{\boldsymbol{m}})$. This gives us three parameters for the variable a, two parameters for the variable b and six parameters for the variable c. All the parameters, their domains inherited from the respective variable and an example parametrisation $\boldsymbol{P}$ are given in Table 4.1

The total number of parametrisations of the parametric network $G_{\boldsymbol{m}}$ is $3^3 \times 2^2 \times 2^6 = 6912 = |\mathbb{P}(G_{\boldsymbol{m}})|$. One should note that the influence graph G is relatively sparse (variables a and b only have one regulator, $|R(a)| = |R(b)| = 1$), a trait common for influence graphs of gene regulatory networks. The number of parametrisations is thus not particularly high among other graphs of similar size and with the same variable domains. As biological examples naturally tend to have higher numbers of variables, the enumeration of all parametrisations is not viable for most computations.

Parametrising $G_{\boldsymbol{m}}$ by $\boldsymbol{P}$ gives us the discrete regulatory network, in our case multivalued network, $F_{\boldsymbol{P}}$ defined as,

$$\begin{aligned} F_{\boldsymbol{P}}(\boldsymbol{x}) &= \left(\boldsymbol{P}_{a,\omega_a(\boldsymbol{x})}, \boldsymbol{P}_{b,\omega_b(\boldsymbol{x})}, \boldsymbol{P}_{c,\omega_c(\boldsymbol{x})}\right) \\ &= \left((\boldsymbol{x}_a + 1) \bmod 3, 1 - \boldsymbol{x}_b, \left\lfloor \frac{\boldsymbol{x}_a + \boldsymbol{x}_b}{2} \right\rfloor\right) \\ &= F_A \end{aligned}$$

coinciding with the definition of the multivalued network F_A from Example 2.1.

		$\boldsymbol{P}$
$K_{a,(a=0)}$	$\in X_a$	1
$K_{a,(a=1)}$		2
$K_{a,(a=2)}$		0
$K_{b,(b=0)}$	$\in X_b$	1
$K_{b,(b=1)}$		0
$K_{c,(a=0,b=0)}$	$\in X_c$	0
$K_{c,(a=0,b=1)}$		0
$K_{c,(a=1,b=0)}$		0
$K_{c,(a=1,b=1)}$		1
$K_{c,(a=2,b=0)}$		1
$K_{c,(a=2,b=1)}$		1

Table 4.1: Table of all parameters of the parametric multivalued network $G_{\boldsymbol{m}}$ with their respective domains. An example assignment of parameter values, a parametrisation $\boldsymbol{P}$ is given in the last column.

Before proceeding with the definition of parametric regulatory network semantics, we introduce a partial order on the parametrisations of a parametric regulatory network. The *parametrisation order* is given as a piecewise order on vectors of length $|\Omega|$.

Definition 4.5 (Parametrisation Order). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network.

Then the parametrisation order on the parametrisations of $G_{\boldsymbol{m}}$ is the partial order $\leq_{G_{\boldsymbol{m}}}$ defined as

$$\boldsymbol{P} \leq_{G_{\boldsymbol{m}}} \boldsymbol{P}' \overset{\Delta}{\iff} \forall\, (v, \boldsymbol{\omega}) \in \Omega,\ \boldsymbol{P}_{v,\boldsymbol{\omega}} \leq \boldsymbol{P}'_{v,\boldsymbol{\omega}}$$

for all $\boldsymbol{P}, \boldsymbol{P}' \in \mathbb{P}\,(G_{\boldsymbol{m}})$.

The parametrisation order allows us to showcase some nice structural properties of the parametric regulatory network semantics and forms the basis for the abstract regulatory network semantics introduced in Section 4.3.

4.2 Concrete Semantics of Parametric Regulatory Networks

We have already given the intuition of the parametric regulatory network semantics being akin to union over the semantics of all possible parametrised networks. A simple union, however, allows the parametric regulatory network to behave according to a different parametrised network after each transition. This would, in particular, allow the same state visited more than once to be followed by a transition from different parametrised network semantics each time. To eliminate such inconsistent behaviour, instead of taking the union

over all the semantics, we consider the union over all the behaviours (traces) of the parametrised networks. In other words, given a trace of the parametric regulatory network, there should exists at least one parametrisation $\boldsymbol{P}$, such that the parametrised network $F_{\boldsymbol{P}}$ can reproduce the trace.

In this section we examine the semantics of parametric regulatory networks in detail and give a formal definition satisfying the outlined consistency condition. To this end, for each transition in the union of parametrised network semantics, we identify parametrisations which enable said transition. Intuitively, a transition is enabled (allowed) by a parametrisation if it belongs to the semantics of the corresponding parametrised network.

Definition 4.6 (Parametrisation Set Enabling a Transition). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $\xrightarrow{F_{\boldsymbol{P}}}$ be a multivalued network semantics of an arbitrary but fixed type for all $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$.

Then, for any transition $t = (\boldsymbol{x}, \boldsymbol{y}) \in \bigcup_{\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})} \xrightarrow{F_{\boldsymbol{P}}}$, the parametrisation set $p(t)$ enabling t is defined as follows:

$$p(t) \triangleq \left\{ \boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}}) \;\middle|\; t \in \xrightarrow{F_{\boldsymbol{P}}} \right\}$$

The definition of $p(t)$ can be naturally extended to sets of transitions T as the intersection of $p(t)$ for each transition $t \in T$.

Definition 4.7 (Parametrisation Set Enabling a Transition Set). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $\xrightarrow{F_{\boldsymbol{P}}}$ be a multivalued network semantics of an arbitrary but fixed type for all $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$.

Then for any set of transitions $T \subseteq \bigcup_{\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})} \xrightarrow{F_{\boldsymbol{P}}}$, the parametrisation set enabling T is defined as follows:

$$p(\emptyset) \triangleq \mathbb{P}(G_{\boldsymbol{m}}),$$

$$p(T) \triangleq \bigcap_{t \in T} p(t) \text{ if } T \neq \emptyset.$$

By definition, all transitions in a set T belong to the semantics of the parametrised network $F_{\boldsymbol{P}}$ ($T \subseteq \xrightarrow{F_{\boldsymbol{P}}}$) if and only if $\boldsymbol{P} \in p(T)$. Thus, taking $T = \widetilde{\pi}$ to be the set of transitions of a trace π over the union of parametrised network semantics, $p(\widetilde{\pi})$ becomes the set of all parametrisations $\boldsymbol{P}$ such that π is a trace of the parametrised network $F_{\boldsymbol{P}}$. Putting the consistency condition on parametric regulatory networks semantics outlined in the beginning of the section to formal terms, a trace π over the union of parametrised network semantics is a trace of the parametric regulatory network (is *realisable*) if $p(\widetilde{\pi}) \neq \emptyset$.

By only considering realisable traces, we prevent the network from behaving differently in the same state over the course of a single trace. However, separate traces are still allowed to display different behaviours in the same state. Parametric regulatory network semantics thus cannot be defined on the states alone. To this end, we annotate the states with an information on past choices to disqualify inconsistent behaviour. As the required information is

independent of the order of past transitions, the set of transition taken, or its parametrisation set directly, is sufficient.

Definition 4.8 (Concrete Semantics of Parametric Regulatory Networks). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network of dimension n and let $\xrightarrow{F_{\boldsymbol{P}}} \subseteq X \times X$ be a multivalued network semantics of an arbitrary but fixed type on state space X for all $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$.

Then the parametric regulatory network semantics of $G_{\boldsymbol{m}}$ is a relation $\xrightarrow{G_{\boldsymbol{m}}} \subseteq \left(X \times 2^{\mathbb{P}(G_{\boldsymbol{m}})}\right) \times \left(X \times 2^{\mathbb{P}(G_{\boldsymbol{m}})}\right)$ defined as follows:

$$
\begin{aligned}
&(x, \mathcal{P}) \times (y, \mathcal{P} \cap p((x,y))) \in \xrightarrow{G_{\boldsymbol{m}}} \overset{\Delta}{\iff} \\
&\mathcal{P} \cap p((x,y)) \neq \emptyset \iff \\
&\exists \boldsymbol{P} \in \mathcal{P},\ \boldsymbol{x} \xrightarrow{F_{\boldsymbol{P}}} \boldsymbol{y}
\end{aligned}
$$

While all subsets of parametrisations are considered for the definition, not all necessarily have to appear in the resulting transitions. We write $P\left(\xrightarrow{G_{\boldsymbol{m}}}\right)$ to denote the set of all parametrisation sets of $G_{\boldsymbol{m}}$ enabling some transition set possible under the semantics $\xrightarrow{G_{\boldsymbol{m}}}$, formally:

$$
P\left(\xrightarrow{G_{\boldsymbol{m}}}\right) \triangleq \left\{ p(T) \;\middle|\; T \subseteq \bigcup_{\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})} \xrightarrow{F_{\boldsymbol{P}}} \right\}
$$

The parametric regulatory network semantics as given in Definition 4.8 can be constructed for arbitrary choice of multivalued network semantics. The generality is certainly welcome in making the parametric regulatory networks more versatile. Due to the unknown nature of the multivalued network semantics used, however, no information can be obtained on the properties of the parametrisation sets enabling the individual transitions. With no structural information, one is forced to represent the parametrisation sets explicitly, despite the number of parametrisations being asymptotically double exponential in the number of variables.

To be able to exploit structural information of the parametrisation sets enabling transitions, we limit ourselves to the most widely used semantics of multivalued networks, the generalised asynchronous semantics and its subsets, such as the synchronous and asynchronous semantics. As we are working purely with multivalued networks, we consider the generalised asynchronous semantics as adjusted for multivalued networks in Definition 2.9, taking advantage of the value changes by steps of size 1.

Generalised asynchronous semantics subsumes all the types of semantics we consider, however, the very versatility of generalised asynchronous semantics may obscure some kinetic information. E.g. $\boldsymbol{x} \xrightarrow{F} \boldsymbol{x}\,[v \mapsto \boldsymbol{x}_v + 1]$ may refer to a variety of transitions. It may refer to a fully asynchronous transition updating the value of v only. However, it may as well refer to a transition that synchronously updates values of v and another variable, or several variables, u

such that $F_u(\boldsymbol{x}) = \boldsymbol{x}_u$. We therefore introduce $\mathcal{S}\colon \xrightarrow[gen]{G_{\boldsymbol{m}}} \to 2^{\{1,\dots,n\}}$ to annotate each transition $t \in \xrightarrow[gen]{G_{\boldsymbol{m}}}$ with a set of all variables updated synchronously, even if their value is not allowed to change.

Consider now the inconsistent behaviour, where a variable v first increases and then decreases value in a repeatedly visited state $\boldsymbol{x}$. With the generalised asynchronous semantics and its subsets, we can characterise this inconsistency directly on the level of parametrisations rather than transitions. When the value increases, a regulation function with increasing value in state $\boldsymbol{x}$ is required $f_v(\boldsymbol{x}) > \boldsymbol{x}_v$, or in terms of parametrised networks, $\boldsymbol{P}_{v,\omega_v(\boldsymbol{x})} > \boldsymbol{x}_v$. On the other hand, the decrease uses $f_v(\boldsymbol{x}) < \boldsymbol{x}_v$, or $\boldsymbol{P}_{v,\omega_v(\boldsymbol{x})} < \boldsymbol{x}_v$. Any such inconsistency can thus be characterised by a conflict on a single parameter value. Similar distinction can be made for inconsistencies when a variable v once changes value and once remains constant under the same transition t with $v \in \mathcal{S}(t)$.

A value changing (or in synchronous updates staying the same) during each transition can thus be interpreted as making a decision on the value of a particular parameter. Working only with parametrisations which comply with the previously made choices thus guarantees a consistent behaviour. We identify such compatible sets of parametrisations similarly to Definition 4.6 for transitions.

Definition 4.9 (Parametrisation Set Directing Variable Evolution). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $\xrightarrow{F_{\boldsymbol{P}}} \subseteq \xrightarrow[gen]{F_{\boldsymbol{P}}}$ be semantics with synchronicity relation $\mathcal{S}$ for all $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$.

Then, for any parameter $K_{v,\boldsymbol{\omega}}$ and any value $i \in \{1,\dots,\boldsymbol{m}_v\}$, the parametrisation sets $p(K_{v,\boldsymbol{\omega}} \geq i)$ and $p(K_{v,\boldsymbol{\omega}} \leq i)$ of all parametrisations that prevent v from decreasing, respectively increasing, value in $\boldsymbol{\omega}$ are defined as follows:

$$p(K_{v,\boldsymbol{\omega}} \geq i) \triangleq \{\, \boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}}) \mid \boldsymbol{P}_{v,\boldsymbol{\omega}} \geq i \,\}$$
$$p(K_{v,\boldsymbol{\omega}} \leq i) \triangleq \{\, \boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}}) \mid \boldsymbol{P}_{v,\boldsymbol{\omega}} \leq i \,\}$$

Although the inconsistent behaviour is fully preventable by following the decisions made on single parameter values, the decisions themselves are tied to transitions. Therefore, by grouping the parametrisation sets sharing parameter value, representing the choices of a individual parameter values, by the respective transition, we obtain a refinement of Definition 4.6:

Definition 4.10 (Parametrisation Set Enabling a Transition under Generalised Asynchronous or Derived Semantics). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $\xrightarrow{F_{\boldsymbol{P}}} \subseteq \xrightarrow[gen]{F_{\boldsymbol{P}}}$ be a multivalued network semantics of an arbitrary but fixed type for all $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$.

Then, for any transition $t = (\boldsymbol{x}, \boldsymbol{y}) \in \bigcup_{\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})} \xrightarrow{F_{\boldsymbol{P}}}$, the parametrisation

set $p(t)$ enabling t is defined as follows:

$$p((\boldsymbol{x},\boldsymbol{y})) \triangleq \left\{ \boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}}) \;\middle|\; (\boldsymbol{x},\boldsymbol{y}) \in \xrightarrow{F_{\boldsymbol{P}}} \right\} = \bigcap_{\mathcal{P} \in \mathcal{K}((\boldsymbol{x},\boldsymbol{y}))} \mathcal{P}$$

where $\mathcal{K}((\boldsymbol{x},\boldsymbol{y}))$ is the set of all the parametrisation sets restricting individual parameters:

$$\begin{aligned}\mathcal{K}((\boldsymbol{x},\boldsymbol{y})) \triangleq & \left\{ p\left(K_{v,\omega_v(\boldsymbol{x})} \geq \boldsymbol{y}_v\right) \;\middle|\; v \in D(\boldsymbol{x},\boldsymbol{y}) \wedge \boldsymbol{y}_v = \boldsymbol{x}_v + 1 \right\} \cup \\ & \left\{ p\left(K_{v,\omega_v(\boldsymbol{x})} \leq \boldsymbol{y}_v\right) \;\middle|\; v \in D(\boldsymbol{x},\boldsymbol{y}) \wedge \boldsymbol{y}_v = \boldsymbol{x}_v - 1 \right\} \cup \\ & \left\{ p\left(K_{v,\omega_v(\boldsymbol{x})} \geq \boldsymbol{x}_v\right), p\left(K_{v,\omega_v(\boldsymbol{x})} \leq \boldsymbol{x}_v\right) \;\middle|\; v \in \mathcal{S}(t) \setminus D(\boldsymbol{x},\boldsymbol{y}) \right\}\end{aligned}$$

Example 4.2. *Take the parametric regulatory network $G_{\boldsymbol{m}}$ from Example 4.1 and a generalised asynchronous transition $t = (201, 111)$ with $\mathcal{S}(t) = \{a, b, c\}$.*

The parametrisation set $p(t)$ enabling the transition t is the set of all parametrisations $\boldsymbol{P}' \in \mathbb{P}(G_{\boldsymbol{m}})$ with $t \in \xrightarrow{F_{\boldsymbol{P}'}}$.

As t is a transition of the generalised asynchronous semantics, in fact t is fully synchronous ($\mathcal{S}(t) = V$), we get:

$$p(t) = p(K_{a,2} \leq 1) \cap p(K_{b,0} \geq 1) \cap p(K_{c,20} \leq 1) \cap p(K_{c,20} \geq 1)$$

by Definition 4.10. The parametrisation set $p(t)$ is therefore the set of all parametrisations $\boldsymbol{P}' \in \mathbb{P}(G_{\boldsymbol{m}})$ with $\boldsymbol{P}'_{a,2} \leq 1$, $\boldsymbol{P}'_{b,0} \geq 1$ and $\boldsymbol{P}'_{c,20} = 1$.

$|p(t)| = 1152$ is still too many parametrisations to list. To give an example, the parametrisation $\boldsymbol{P} = (K_{a,0} = 1, K_{a,1} = 2, K_{a,2} = 0, K_{b,0} = 1, K_{b,1} = 0, K_{c,00} = 0, K_{c,01} = 0, K_{c,10} = 0, K_{c,11} = 1, K_{c,20} = 1, K_{c,21} = 1)$ from Example 4.1 belongs to $p(t)$, $\boldsymbol{P} \in p(t)$. This is also documented by the transition $t = (201, 111)$ appearing in the state space graph of $F_A = F_{\boldsymbol{P}}$ with the synchronous semantics in Figure 2.2.

By limiting ourselves to generalised asynchronous semantics and its subsets, we have obtained important link between transitions and the properties of the sets of parametrisations that enable them. This connection is heavily exploited in the following Section 4.3.

4.3 Abstract Semantics of Parametric Regulatory Networks

The parametric regulatory network semantics as per Definition 4.8 rely on annotating the states with parametrisation sets. As the number of parametrisations is exponential in the number of regulation states and thus in general double exponential in the number of variables, explicit representation of the parametrisation sets is practically infeasible. To combat the computational limitation, we introduce an abstract version of the parametric regulatory network semantics which allows us to represent the parametrisation sets without explicit enumeration of the parametrisations. To this end we rely on the structural

connection between transitions and parametrisations that enable them, highlighted in Definition 4.10, maintaining the restriction to semantics $\xrightarrow{G_{\boldsymbol{m}}} \subseteq \xrightarrow[gen]{G_{\boldsymbol{m}}}$.

To illustrate how the connection between transitions and parametrisations can be utilised to avoid explicit enumeration of parametrisations, consider the parametrisation set $\mathbb{P}(G_{\boldsymbol{m}})$ as a lattice $(\mathbb{P}(G_{\boldsymbol{m}}), \leq_{G_{\boldsymbol{m}}})$ with the parametrisation order from Definition 4.5. We first remark that there exists unique $\leq_{G_{\boldsymbol{m}}}$-maximal parametrisation $\mathbf{1}^{G_{\boldsymbol{m}}} = \|_{i=1}^{n} \{\boldsymbol{m}_i\}^{|\Omega_i|} \in \mathbb{P}(G_{\boldsymbol{m}})$, as well as a $\leq_{G_{\boldsymbol{m}}}$-minimal parametrisation $\mathbf{0}^{G_{\boldsymbol{m}}} = \{0\}^{|\Omega|} \in \mathbb{P}(G_{\boldsymbol{m}})$. As such, the lattice $(\mathbb{P}(G_{\boldsymbol{m}}), \leq_{G_{\boldsymbol{m}}})$ is bounded.

Consider now a parametrisation $\boldsymbol{P}$ defined as follows for each $(u, \boldsymbol{\omega}') \in \Omega$:

$$\boldsymbol{P}_{u,\boldsymbol{\omega}'} = \left\{ \begin{array}{ll} i & \text{if } (u,\boldsymbol{\omega}') = (v,\boldsymbol{\omega}) \\ \boldsymbol{m}_u & \text{otherwise} \end{array} \right\}$$

for arbitrary $(v, \boldsymbol{\omega}) \in \Omega$ and $i \in \{0, \ldots, \boldsymbol{m}_v\}$. For any such parametrisation, we can construct the principal ideal $\mathcal{P} = \left\{ \boldsymbol{P}' \in \mathbb{P}(G_{\boldsymbol{m}}) \;\middle|\; \boldsymbol{P}' \leq_{G_{\boldsymbol{m}}} \boldsymbol{P} \right\}$ of the lattice $(\mathbb{P}(G_{\boldsymbol{m}}), \leq_{G_{\boldsymbol{m}}})$ with $\boldsymbol{P}$ as the principal element. Since all parameters except $K_{v,\boldsymbol{\omega}}$ are at their maximum values in $\boldsymbol{P}$, any parametrisation $\boldsymbol{P}' \in \mathbb{P}(G_{\boldsymbol{m}})$ with $\boldsymbol{P}'_{v,\boldsymbol{\omega}} \leq i$ is necessarily smaller or equal to $\boldsymbol{P}$ according to the parametrisation order, $\boldsymbol{P}' \leq_{G_{\boldsymbol{m}}} \boldsymbol{P}$, as it assigns smaller or equal value to each variable. We can therefore describe the principal ideal of $\boldsymbol{P}$ simply as $\mathcal{P} = \left\{ \boldsymbol{P}' \in \mathbb{P}(G_{\boldsymbol{m}}) \;\middle|\; \boldsymbol{P}'_{v,\boldsymbol{\omega}} \leq i \right\} = p(K_{v,\boldsymbol{\omega}} \leq i)$, which is equivalent to the definition of the parametrisation set preventing variable value increase.

Symmetrically, consider the parametrisation $\boldsymbol{P}$ to be of the following form for each $(u, \boldsymbol{\omega}') \in \Omega$:

$$\boldsymbol{P}_{u,\boldsymbol{\omega}'} = \left\{ \begin{array}{ll} i & \text{if } (u,\boldsymbol{\omega}') = (v,\boldsymbol{\omega}) \\ 0 & \text{otherwise} \end{array} \right\}$$

for arbitrary $(v, \boldsymbol{\omega}) \in \Omega$ and $i \in \{0, \ldots, \boldsymbol{m}_v\}$. Then, the principal filter (the dual of a principal ideal) with $\boldsymbol{P}$ as the principal element is the set $\mathcal{P} = \left\{ \boldsymbol{P}' \in \mathbb{P}(G_{\boldsymbol{m}}) \;\middle|\; \boldsymbol{P}' \geq \boldsymbol{P} \right\} = \left\{ \boldsymbol{P}' \in \mathbb{P}(G_{\boldsymbol{m}}) \;\middle|\; \boldsymbol{P}'_{v,\boldsymbol{\omega}} \geq i \right\} = p(K_{v,\boldsymbol{\omega}} \geq i)$, which is equivalent to the parametrisation set preventing variable value decrease by symmetrical reasoning.

We know that any ideal or filter of a lattice is a convex sublattice. Additionally, a principal ideal or a principal filter of a bounded lattice is also bounded with the principal element as the maximum, respectively minimum. Any parametrisation set directing variable evolution is thus a bounded convex sublattice of the lattice $(\mathbb{P}(G_{\boldsymbol{m}}), \leq_{G_{\boldsymbol{m}}})$ of all parametrisations. A bounded convex sublattice is uniquely identified by its minimum and maximum elements. The set of parametrisations $\emptyset \neq \mathcal{P} \in P\left(\xrightarrow{G_{\boldsymbol{m}}}\right)$ can thus be solely represented by the minimum and maximum elements, $\mathbf{0}, \mathbf{1} \in \mathcal{P}$, respectively. We refer to bounded convex sublattices of the lattice of all parametrisations represented by their minimum and maximum as *parametrisation lattices*.

Definition 4.11 (Parametrisation Lattice). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $(\mathbb{P}(G_{\boldsymbol{m}}), \leq_{G_{\boldsymbol{m}}})$ be the lattice of all parametrisations of $G_{\boldsymbol{m}}$ with the parametrisation order.

Then, a parametrisation lattice generated by $\mathbf{0}, \mathbf{1} \in \mathbb{P}(G_{\boldsymbol{m}})$ is the lattice $[\mathbf{0}, \mathbf{1}] = (\mathcal{P}, \leq_{G_{\boldsymbol{m}}})$ where $\mathcal{P} = \{ \boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}}) \mid \mathbf{0} \leq_{G_{\boldsymbol{m}}} \boldsymbol{P} \leq_{G_{\boldsymbol{m}}} \mathbf{1} \}$.

We write $P^\sharp(G_{\boldsymbol{m}}) = \{ [\mathbf{0}, \mathbf{1}] \mid \mathbf{0}, \mathbf{1} \in \mathbb{P}(G_{\boldsymbol{m}}) \}$ to denote the set of all parametrisation lattices of the parametric regulatory network $G_{\boldsymbol{m}}$.

By abuse of notation we write $\boldsymbol{P} \in [\mathbf{0}, \mathbf{1}]$ for any $\boldsymbol{P} \in \mathcal{P}$ where $(\mathcal{P}, \leq_{G_{\boldsymbol{m}}}) = [\mathbf{0}, \mathbf{1}]$.

Observe that Definition 4.11 imposes no restriction on the minimum and maximum parametrisations $\mathbf{0}, \mathbf{1}$. Instead, any $[\mathbf{0}, \mathbf{1}]$ with $\mathbf{0} >_{G_{\boldsymbol{m}}} \mathbf{1}$ is interpreted as the empty lattice $\varnothing = (\emptyset, \leq_{G_{\boldsymbol{m}}})$ which is also a parametrisation lattice. In fact, $\varnothing$ is the only parametrisation lattice which is not bounded and convex. On the other hand, all bounded and convex sublattices of the lattice of all parametrisations $(\mathbb{P}(G_{\boldsymbol{m}}), \leq_{G_{\boldsymbol{m}}})$ are expressible as parametrisation lattices using their bounds. This corresponds to every bounded convex sublattice $[\mathbf{0}, \mathbf{1}]$ being uniquely given as an intersection of a principal ideal, whose principal element is the maximum $\mathbf{1}$, with a principal filter, whose principal element is the minimum $\mathbf{0}$.

The parametrised lattices, being essentially bounded convex sublattices of $(\mathbb{P}(G_{\boldsymbol{m}}), \leq_{G_{\boldsymbol{m}}})$ allow us to represent more than just parametrisation sets directing variable evolution. In particular, parametrisation sets enabling a transition are built from parametrisation sets directing variable evolution using only intersections. Similarly parametrisation sets enabling transition sets are built from parametrisation sets enabling transition using only intersections. As a non-empty intersection of bounded convex sublattices is a bounded convex sublattice, any non-empty parametrisation set enabling a transition set, $\emptyset \neq \mathcal{P} \in P\left(\xrightarrow{G_{\boldsymbol{m}}}\right)$, coupled with the parametrisation order is a bounded convex sublattice of the lattice $(\mathbb{P}(G_{\boldsymbol{m}}), \leq_{G_{\boldsymbol{m}}})$ of all parametrisations. Any parametrisation set enabling a transition set thus forms a parametrisation lattice when coupled with the parametrisation order.

Hence, the parametrisation order allows us to easily capture parametrisation sets of interest by parametrisation lattices, represented by only two elements. We utilise parametrisation lattices to define an abstraction of parametrisation sets. In formal terms, the abstraction can be captured using a Galois connection [22].

Definition 4.12 (Parametrisation Set Abstraction). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network.

The parametrisation set abstraction of $G_{\boldsymbol{m}}$ is defined by the following Galois connection $\underset{\gamma}{\overset{\alpha}{\rightleftharpoons}}$:

$$\begin{array}{lll} \alpha\colon 2^{\mathbb{P}(G_{\boldsymbol{m}})} \to P^\sharp(G_{\boldsymbol{m}}) & & \gamma\colon P^\sharp(G_{\boldsymbol{m}}) \to 2^{\mathbb{P}(G_{\boldsymbol{m}})} \\ \alpha\colon \emptyset \mapsto \varnothing & & \gamma\colon (\mathcal{P}, \leq_{G_{\boldsymbol{m}}}) \mapsto \mathcal{P} \\ \alpha\colon \mathcal{P} \mapsto [\bigwedge_{\mathcal{P}}, \bigvee_{\mathcal{P}}] & \text{for } \mathcal{P} \neq \emptyset & \end{array}$$

	$K_{a,0}$	$K_{a,1}$	$K_{a,2}$	$K_{b,0}$	$K_{b,1}$	
$\mathbf{0} = \bigwedge_{p(t)}$	0	0	0	1	0	
$\mathbf{1} = \bigvee_{p(t)}$	2	2	1	1	1	
	$K_{c,00}$	$K_{c,01}$	$K_{c,10}$	$K_{c,11}$	$K_{c,20}$	$K_{c,21}$
$\mathbf{0} = \bigwedge_{p(t)}$	0	0	0	0	1	0
$\mathbf{1} = \bigvee_{p(t)}$	1	1	1	1	1	1

Table 4.2: The infimum and supremum parametrisations of the parametrisation set $p(t)$.

where $\wedge$ and $\vee$ are the standard lattice operators of meet and join, in the lattice of all parametrisations $(\mathbb{P}(G_{\boldsymbol{m}}), \leq_{G_{\boldsymbol{m}}})$.

Example 4.3. *Let $p(t)$ be the parametrisation set enabling the transition $t = (201, 111)$ with $\mathcal{S}(t) = \{a, b, c\}$ from Example 4.2.*

The abstraction $\alpha(p(t)) = \left[\bigwedge_{p(t)}, \bigvee_{p(t)}\right]$ is characterised by the infimum and supremum of $p(t)$. From Example 4.2, $p(t) = p(K_{a,2} \leq 1) \cap p(K_{b,0} \geq 1) \cap p(K_{c,20} \geq 1)$.[1] The infimum $\mathbf{0} = \bigwedge_{p(t)}$ and supremum $\mathbf{1} = \bigvee_{p(t)}$ are given by the bounds on the parameters defining the parametrisation sets in $\mathcal{K}(t)$. We list the parametrsations $\mathbf{0}$ and $\mathbf{1}$ in Table 4.2.

The backwards concretisation $\gamma([\mathbf{0}, \mathbf{1}])$ is simply the parametrisation lattice $[\mathbf{0}, \mathbf{1}]$ stripped of the lattice structure (order). As such, $\boldsymbol{P}' \in \gamma([\mathbf{0}, \mathbf{1}]) \iff \forall (v, \boldsymbol{\omega}) \in \Omega,\ \mathbf{0}_{v,\boldsymbol{\omega}} \leq \boldsymbol{P}'_{v,\boldsymbol{\omega}} \leq \mathbf{1}_{v,\boldsymbol{\omega}}$. By foregoing all 0 valued parameters in $\mathbf{0}$ and maximum valued parameters in $\mathbf{1}$, we get $\boldsymbol{P}' \in \gamma([\mathbf{0}, \mathbf{1}]) \iff \boldsymbol{P}'_{a,2} \leq 1 \wedge \boldsymbol{P}'_{b,0} \geq 1 \wedge \boldsymbol{P}'_{c,20} \geq 1$, which is exactly when $\boldsymbol{P}' \in p(t)$. Thus $\gamma(\alpha(p(t))) = p(t)$, the abstraction $\alpha(p(t))$ is exact.

We use the parametrisation set abstraction to define the *abstract semantics of parametric regulatory networks*. Instead of annotating the states of the network by explicit parametrisation sets, abstract semantics utilise parametrisation lattices, which are fully specified by only two parametrisations. The state annotations thus end up being of linear rather than exponential size in the number of parameters.

Definition 4.13 (Abstract Semantics of Parametric Regulatory Networks). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network of dimension n and let $\xrightarrow{F_{\boldsymbol{P}}} \subseteq \underset{gen}{\xrightarrow{F_{\boldsymbol{P}}}}$ be a multivalued network semantics of fixed type for all $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$.

Then the abstract parametric regulatory network semantics of $G_{\boldsymbol{m}}$ is a

[1] We drop $p(K_{c,20} \leq 1)$ from the intersection as $1 = \boldsymbol{m}_c$ giving us $p(K_{c,20} \leq 1) = \mathbb{P}(G_{\boldsymbol{m}})$.

relation $\xrightarrow[abs]{G_{\boldsymbol{m}}} \subseteq \left(X_{\boldsymbol{m}} \times P^{\sharp}\left(G_{\boldsymbol{m}}\right)\right) \times \left(X_{\boldsymbol{m}} \times P^{\sharp}\left(G_{\boldsymbol{m}}\right)\right)$ defined as follows:

$$(x, [\mathbf{0}, \mathbf{1}]) \times (y, [\mathbf{0}, \mathbf{1}] \cap \alpha\left(p\left((x, y)\right)\right)) \in \xrightarrow[abs]{G_{\boldsymbol{m}}} \overset{\Delta}{\Longleftrightarrow}$$
$$\left(\exists \boldsymbol{P} \in \mathbb{P}\left(G_{\boldsymbol{m}}\right),\ x \xrightarrow{F_{\boldsymbol{P}}} y\right) \wedge \left([\mathbf{0}, \mathbf{1}] \cap \alpha\left(p\left((x, y)\right)\right) \neq \varnothing\right)$$

To ease notation, we write $p^{\sharp}\left(t\right) = \alpha\left(p\left(t\right)\right)$ to denote the abstract parametrisation set enabling a transition $t \in \bigcup_{\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})} \xrightarrow{F_{\boldsymbol{P}}}$ and $p^{\sharp}\left(T\right) = \alpha\left(p\left(T\right)\right)$ to denote the abstract parametrisation set enabling a transition set $T \subseteq \bigcup_{\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})} \xrightarrow{F_{\boldsymbol{P}}}$.

The Galois connection in Definition 4.12 captures arbitrary set of parametrisations $\mathcal{P} \subseteq \mathbb{P}\left(G_{\boldsymbol{m}}\right)$. In general, such parametrisation set $\mathcal{P}$ may not have a minimum, $\bigwedge \mathcal{P} \notin \mathcal{P}$ or maximum, $\bigvee \mathcal{P} \notin \mathcal{P}$ ($\mathcal{P}$ is not bounded) or it may not be convex, with respect to the parametrisation order $\leq_{G_{\boldsymbol{m}}}$. The abstraction thus constructs the smallest bounded convex cover of $\mathcal{P}$ in the form of parametrisation lattice $[\bigwedge \mathcal{P}, \bigvee \mathcal{P}]$. Therefore, in the general case, the abstraction is an over-approximation of the parametrisation set.

As illustrated in the beginning of the section, however, the parametrisation sets enabling transition sets, $\mathcal{P} \in P\left(\xrightarrow{G_{\boldsymbol{m}}}\right)$, are exactly the element sets of the parametrisation lattices. The abstraction restricted to $P\left(\xrightarrow{G_{\boldsymbol{m}}}\right)$ is hence exact.

Theorem 4.1 (Parametrisation Set Abstraction is Exact)**.** *Let $G_{\boldsymbol{m}}$ be a parametric regulatory network with semantics $\xrightarrow{G_{\boldsymbol{m}}} \subseteq \xrightarrow[gen]{G_{\boldsymbol{m}}}$.*

Then, for any $\mathcal{P} \in P\left(\xrightarrow{G_{\boldsymbol{m}}}\right)$ and for any $[\mathbf{0}, \mathbf{1}] \in P^{\sharp}\left(G_{\boldsymbol{m}}\right)$:

$$\gamma\left(\alpha\left(\mathcal{P}\right)\right) = \mathcal{P} \qquad\qquad \alpha\left(\gamma\left([\mathbf{0}, \mathbf{1}]\right)\right) = [\mathbf{0}, \mathbf{1}]$$

Proof. We first prove $\gamma\left(\alpha\left(\mathcal{P}\right)\right) = \mathcal{P}$.

Let us assume $\mathcal{P} \neq \emptyset$ first. By definition we have:

$$\gamma\left(\alpha\left(\mathcal{P}\right)\right) = \gamma\left(\left[\bigwedge_{\mathcal{P}}, \bigvee_{\mathcal{P}}\right]\right) = \left\{ \boldsymbol{P} \in \mathbb{P}\left(G_{\boldsymbol{m}}\right) \;\middle|\; \bigwedge_{\mathcal{P}} \leq_{G_{\boldsymbol{m}}} \boldsymbol{P} \leq_{G_{\boldsymbol{m}}} \bigvee_{\mathcal{P}} \right\}$$

The direction $\mathcal{P} \subseteq [\bigwedge \mathcal{P}, \bigvee \mathcal{P}]$ and thus also $\mathcal{P} \subseteq \gamma\left([\bigwedge \mathcal{P}, \bigvee \mathcal{P}]\right)$ is trivial.

Since $\left(\mathcal{P}, \leq_{G_{\boldsymbol{m}}}\right)$ is an intersection of bounded convex sublattices of $\left(\mathbb{P}\left(G_{\boldsymbol{m}}\right), \leq_{G_{\boldsymbol{m}}}\right)$, it is also a bounded convex sublattice. Therefore, by boundedness, $\mathcal{P}$ has a minimal and a maximal parametrisation, $\bigwedge \mathcal{P}, \bigvee \mathcal{P} \in \mathcal{P}$. Consequently, by convexity, we have $\forall \boldsymbol{P} \in \mathbb{P}\left(G_{\boldsymbol{m}}\right),\ \bigwedge \mathcal{P} \leq_{G_{\boldsymbol{m}}} \boldsymbol{P} \leq_{G_{\boldsymbol{m}}} \bigvee \mathcal{P} \Longrightarrow \boldsymbol{P} \in \mathcal{P}$. Thus, $\gamma\left([\bigwedge \mathcal{P}, \bigvee \mathcal{P}]\right) \subseteq \mathcal{P}$ and $\gamma\left(\alpha\left(\mathcal{P}\right)\right) = \mathcal{P}$.

The case for empty set follows directly from definition:

$$\gamma\left(\alpha\left(\emptyset\right)\right) = \gamma\left(\varnothing\right) = \emptyset$$

Similarly, $\alpha(\gamma([\mathbf{0},\mathbf{1}])) = [\mathbf{0},\mathbf{1}]$ also follows directly from definition:

$$\alpha(\gamma([\mathbf{0},\mathbf{1}])) = \alpha(\{\, \boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}}) \mid \mathbf{0} \leq_{G_{\boldsymbol{m}}} \boldsymbol{P} \leq_{G_{\boldsymbol{m}}} \mathbf{1} \,\}) = [\mathbf{0},\mathbf{1}]$$

□

Thanks to Theorem 4.1 we can guarantee that the abstract parametrisation sets represent the exactly same parametrisations as their concrete counterpart. As such, usage of the abstract semantics of parametric regulatory networks introduces no false positives (over-approximation), nor false negatives (under-approximation) when compared against the concrete semantics. In fact, the concrete and abstract semantics of parametric regulatory networks are equivalent, provided states are only annotated by the parametrisation sets enabling transition sets, $\boldsymbol{P} \in P\left(\xrightarrow{G_{\boldsymbol{m}}}\right)$.

Corollary 4.1.1 (Concrete and Abstract Semantics of Parametric Regulatory Networks are Equivalent)**.** *Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $\xrightarrow{G_{\boldsymbol{m}}} \subseteq \underset{gen}{\xrightarrow{G_{\boldsymbol{m}}}}$, $\underset{abs}{\xrightarrow{G_{\boldsymbol{m}}}} \subseteq \underset{abs\cdot gen}{\xrightarrow{G_{\boldsymbol{m}}}}$ be the concrete and abstract semantics of $G_{\boldsymbol{m}}$ of arbitrary but fixed type.*

Then, for an arbitrary set of transitions $T \in \bigcup_{\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})} \xrightarrow{F_{\boldsymbol{P}}}$ and arbitrary states $\boldsymbol{x},\boldsymbol{y} \in X_{\boldsymbol{m}}$:

$$(\boldsymbol{x}, p(T)) \xrightarrow{G_{\boldsymbol{m}}} (\boldsymbol{y}, p(T \cup \{(\boldsymbol{x},\boldsymbol{y})\})) \iff \left(\boldsymbol{x}, p^\sharp(T)\right) \underset{abs}{\xrightarrow{G_{\boldsymbol{m}}}} \left(\boldsymbol{y}, p^\sharp(T \cup \{(\boldsymbol{x},\boldsymbol{y})\})\right)$$

Theorem 4.1 also gives us stronger grasp on the abstract parametrisation sets enabling a transition themselves. In particular, we now have $\boldsymbol{P} \in \alpha(p(t)) \iff \boldsymbol{P} \in p(t)$. This allows us to simplify the Definition 4.13 of abstract parametric regulatory network semantics, similarly to the concrete semantics, Definition 4.8.

Corollary 4.1.2 (Equivalent Definition of Abstract Parametric Regulatory Network Semantics)**.** *Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $\underset{abs}{\xrightarrow{G_{\boldsymbol{m}}}} \subseteq \underset{abs\cdot gen}{\xrightarrow{G_{\boldsymbol{m}}}}$ be abstract semantics of $G_{\boldsymbol{m}}$.*

Then, for an arbitrary parametrisation lattice $[\mathbf{0},\mathbf{1}] \in P^\sharp(G_{\boldsymbol{m}})$ and arbitrary states $\boldsymbol{x},\boldsymbol{y} \in X_{\boldsymbol{m}}$:

$$(\boldsymbol{x}, [\mathbf{0},\mathbf{1}]) \underset{abs}{\xrightarrow{G_{\boldsymbol{m}}}} \left(\boldsymbol{y}, [\mathbf{0},\mathbf{1}] \cap p^\sharp((\boldsymbol{x},\boldsymbol{y}))\right) \iff \exists\, \boldsymbol{P} \in [\mathbf{0},\mathbf{1}],\ \boldsymbol{x} \xrightarrow{F_{\boldsymbol{P}}} \boldsymbol{y}$$

The abstract parametrisation sets as used within the abstract semantics of parametric regulatory networks, are therefore a perfect equivalent for their concrete counterparts, while avoiding any explicit enumeration of parametrisations. The practical applicability of the abstract semantics, however, relies on computing several operations on the abstract parametrisation sets without

resorting to enumeration of parametrisations. The principal operations required are the computation of $p^\sharp(T \cup \{t\})$ form $p^\sharp(T)$ and membership checking $\boldsymbol{P} \in [\mathbf{0}, \mathbf{1}]$. While, membership checking is realisable without enumeration of parametrisations directly by definition,

$$\boldsymbol{P} \in [\mathbf{0}, \mathbf{1}] \iff \mathbf{0} \leq_{G_{\boldsymbol{m}}} \boldsymbol{P} \leq_{G_{\boldsymbol{m}}} \mathbf{1} \iff \forall (v, \boldsymbol{\omega}) \in \Omega,\ \mathbf{0}_{v,\boldsymbol{\omega}} \leq \boldsymbol{P}_{v,\boldsymbol{\omega}} \leq \mathbf{1}_{v,\boldsymbol{\omega}}$$

the computation of $p^\sharp(T \cup \{t\})$ benefits again from Theorem 4.1.

Corollary 4.1.3 (Abstract Parametrisation Set Enabling a Union of Transition Sets is the Intersection)**.** *Let $G_{\boldsymbol{m}}$ be a parametric regulatory network with semantics $\xrightarrow[abs]{G_{\boldsymbol{m}}} \subseteq \xrightarrow[abs \cdot gen]{G_{\boldsymbol{m}}}$.*

Then, for arbitrary transition sets $T, T' \in \bigcup_{\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})} \xrightarrow{F_{\boldsymbol{P}}}$:

$$p^\sharp(T \cup T') = p^\sharp(T) \cap p^\sharp(T')$$

Computation of $p^\sharp(T \cup \{t\})$ from $p^\sharp(T)$ can thus be conducted by the operation of intersection. Intersections of bounded convex sublattices in general are easy to capture using the meet and join operators. In particular, the minimum of the intersection is the join of the minimums and the maximum of the intersection is the meet of the maximums. By including the empty lattice $\varnothing$, the set of parametrisation lattices merely become closed under intersection as opposed to general bounded convex sublattices. This is captured formally in Proposition 4.1.

Proposition 4.1 (Intersection of Parametrisation Lattices)**.** *Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $[\mathbf{0}, \mathbf{1}], [\mathbf{0}', \mathbf{1}'] \in P^\sharp(G_{\boldsymbol{m}})$ be two arbitrary parametrisation lattices of $G_{\boldsymbol{m}}$.*

Then, $[\mathbf{0}, \mathbf{1}] \cap [\mathbf{0}', \mathbf{1}'] = [\mathbf{0} \vee \mathbf{0}', \mathbf{1} \wedge \mathbf{1}']$.

Proof.

$$\begin{aligned}
&\boldsymbol{P} \in [\mathbf{0}, \mathbf{1}] \cap [\mathbf{0}', \mathbf{1}'] \iff \\
&(\mathbf{0} \leq_{G_{\boldsymbol{m}}} \boldsymbol{P} \leq_{G_{\boldsymbol{m}}} \mathbf{1}) \wedge (\mathbf{0}' \leq_{G_{\boldsymbol{m}}} \boldsymbol{P} \leq_{G_{\boldsymbol{m}}} \mathbf{1}') \iff \\
&\mathbf{0} \vee \mathbf{0}' \leq_{G_{\boldsymbol{m}}} \boldsymbol{P} \leq_{G_{\boldsymbol{m}}} \mathbf{1} \wedge \mathbf{1}' \iff \\
&\boldsymbol{P} \in [\mathbf{0} \vee \mathbf{0}', \mathbf{1} \wedge \mathbf{1}']
\end{aligned}$$

□

To fully capture the computation of $p^\sharp(T \cup \{t\})$ from $p^\sharp(T)$, we also need to be able to infer $p^\sharp(t)$. Recall, however, that $p(t)$ is already an intersection of the parametrisation sets in $\mathcal{K}(t)$. Thus, by Corollary 4.1.3, $p^\sharp(t)$ can be computed as the intersection of $\alpha(\mathcal{P})$ for each $\mathcal{P} \in \mathcal{K}(t)$. The maximums and

minimums of each $p\left(K_{v,\boldsymbol{\omega}} \geq i\right)$ and $p\left(K_{v,\boldsymbol{\omega}} \leq i\right)$ are obvious from definition:

$$\bigvee_{p(K_{v,\boldsymbol{\omega}} \geq i)} = \mathbf{1}^{G_{\boldsymbol{m}}} \quad \bigvee_{p(K_{v,\boldsymbol{\omega}} \leq i)} = \bigtilde{\prod_{(u,\boldsymbol{\omega}') \in \Omega}} \left\{ \begin{array}{ll} i & \text{if } (u,\boldsymbol{\omega}') = (v,\boldsymbol{\omega}) \\ \boldsymbol{m}_u & \text{otherwise} \end{array} \right\}$$

$$\bigwedge_{p(K_{v,\boldsymbol{\omega}} \geq i)} = \bigtilde{\prod_{(u,\boldsymbol{\omega}') \in \Omega}} \left\{ \begin{array}{ll} i & \text{if } (u,\boldsymbol{\omega}') = (v,\boldsymbol{\omega}) \\ 0 & \text{otherwise} \end{array} \right\} \quad \bigwedge_{p(K_{v,\boldsymbol{\omega}} \leq i)} = \mathbf{0}^{G_{\boldsymbol{m}}}$$

We formalise the computation of $p^\sharp\left(T \cup \{t\}\right)$ from $p^\sharp\left(T\right)$ by a *narrowing operator* σ_t[2].

Definition 4.14 (Narrowing Operator of Abstract Parametrisation Sets). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network with abstract semantics $\xrightarrow[abs]{G_{\boldsymbol{m}}} \subseteq \xrightarrow[abs \cdot gen]{G_{\boldsymbol{m}}}$ and let $t \in \bigcup_{\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})} \xrightarrow{F_{\boldsymbol{P}}}$ be arbitrary.

Then, the narrowing operator σ_t for transition t is a function defined as:

$$\sigma_t \colon P^\sharp\left(G_{\boldsymbol{m}}\right) \to P^\sharp\left(G_{\boldsymbol{m}}\right)$$

$$\sigma_t \colon p^\sharp\left(T\right) = [\mathbf{0},\mathbf{1}] \mapsto \left[\begin{array}{l} \bigtilde{\prod_{(v,\boldsymbol{\omega}) \in \Omega}} \max(\left(\{\, i \mid p\left(K_{v,\boldsymbol{\omega}} \geq i\right) \in \mathcal{K}\left(t\right) \,\} \cup \{\mathbf{0}_{v,\boldsymbol{\omega}}\}\right)), \\ \bigtilde{\prod_{(v,\boldsymbol{\omega}) \in \Omega}} \min(\left(\{\, i \mid p\left(K_{v,\boldsymbol{\omega}} \leq i\right) \in \mathcal{K}\left(t\right) \,\} \cup \{\mathbf{1}_{v,\boldsymbol{\omega}}\}\right)) \end{array} \right]$$

where $T \subseteq \bigcup_{\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})} \xrightarrow{F_{\boldsymbol{P}}}$ is arbitrary set of transitions.

It is easy to see that the time complexity of the narrowing σ_t for arbitrary transition t is linear in the number of parameters, $\mathcal{O}\left(\Omega\right)$.

[2] We refer to the σ_t operator as *narrowing*, for a lack of a better name. However, the σ_t operator is not be confused with the usual notion of narrowing in abstract interpretation [22].

Chapter 5

Influence Constraints as Global Constraints on Parametrisations

The abstract semantics of parametric regulatory networks introduced in Section 4.3 are proven to be exact by Theorem 4.1 and the resulting corollaries. Theorem 4.1 is, however, only applicable if all considered parametrisation sets belong to $P\left(\xrightarrow{G_{\boldsymbol{m}}}\right)$. The parametrisations of the form $p\left(T\right) \in P\left(\xrightarrow{G_{\boldsymbol{m}}}\right)$ for some set of transitions T only allow us to differentiate parametrisations solely on their ability to replicate past transitions. As discussed in Section 2.3, however, information on the monotonicity of isolated influences, Definitions 2.10, 2.11, is often available in the literature. In this chapter we formalise the influence monotonicity properties as global constraints on the admissible parametrisations and relax the claim in Theorem 4.1 to obtain similar results for parametrisation sets constrained by the presence of influence properties.

We introduced the influence monotonicity as properties of multivalued networks, Definitions 2.10, 2.11. In this section, however, we consider the influence monotonicity to be given as an input and use it as *monotonicity constraints* on parametrisations [8]. Intuitively, a parametrisation satisfies a monotonicity constraint, if the associated parametrised network has the corresponding monotonicity property. More precisely, a $\boldsymbol{P}$ satisfies a positive, respectively negative, monotonicity constraint on an influence (u, v), if an increase in the value of u cannot cause the decrease, respectively increase, in the value of v, and vice-versa in the parametrised network $F_{\boldsymbol{P}}$. The monotonicity properties of this form are expressible as inequalities on parameter values, we therefore define monotonicity constraints anew, without reliance on the parametrised networks.

We additionally include a constraint called *observability*, used to emphasise necessity of some influences. A parametrisation $\boldsymbol{P}$ satisfies an observability constraint on influence (u, v), if there exists a state such that the sole change in the value of u forces a change in the value of v in the parametrised network $F_{\boldsymbol{P}}$. An observability constraint therefore requires the associated influence to be part of the minimal influence graph. Similarly to the monotonicity constraints, an

observability constraint is expressible as inequalities on the parameter values, and can be defined on the parametrisations themselves.

Definition 5.1 (Global Constraints on Parametrisations). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network with influence graph $G = (V, I)$ and let $e = I$ be arbitrary influence.

Then, an influence constraint r is tuple $r = (e, \star) \in I \times \{+1, -1, \mathrm{o}\}$.

We call a constraint of the form $(e, +1)$ a positive monotonicity constraint on influence e. Similarly, a constraint of the form $(e, -1)$ a negative monotonicity constraint on influence e. Finally, a constraint of the form (e, o) is an observability constraint on influence e.

To reduce notation nesting, we write $(u, v, \star) = (e, \star)$ for any influence $e = (u, v)$.

Set of all influence constraints of an influence graph $G = (V, I)$ is denoted $R(G) \subseteq I \times \{+1, -1, \mathrm{o}\}$. An influence constraint set $R(G)$ is well-formed if for any influence $e \in I$, $\{(e, +1), (e, -1)\} \not\subseteq R(G)$.

The influence constraints can be considered labels on the edges of the influence graph. We then consider the set of all constraints present for the given influence graph to restrict the parametrisation space. Influence constraints are not exclusive, meaning each influence can have multiple constraints. This applies in particular to combinations of monotonicity and observability constraints, as having both monotonicity constraints on a single influence is, within our framework, equivalent to having no such influence. We thus consider only *well-formed* constraint sets further on, to avoid such pathological cases.

5.1 Concrete Constrained Semantics of Parametric Regulatory Networks

Definition 5.1 fixes the notation for the influence constraints, however, does not provide the semantics. The semantics of the monotonicity constraints follow the monotonicity properties in Definitions 2.10, 2.11. Without the use of parametrised networks, a parametrisation $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$ satisfies a positive monotonicity constraint $(u, v, +1)$, if:

$$\forall \boldsymbol{\omega} \in \Omega_v,\ \forall i \in \{1, \ldots, \boldsymbol{m}_u\},\ \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i]} \geq \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i-1]}$$

i.e., the sole increase of the activator u cannot cause a decrease of the regulated variable v.

Similarly, a parametrisation $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$ satisfies a negative monotonicity constraint $(u, v, -1)$, if:

$$\forall \boldsymbol{\omega} \in \Omega_v,\ \forall i \in \{1, \ldots, \boldsymbol{m}_u\},\ \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i]} \leq \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i-1]}$$

i.e., the sole increase of the inhibitor u cannot cause an increase of the regulated variable v.

Finally, a parametrisation $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$ satisfies an observability constraint (u, v, o), if:

$$\exists \boldsymbol{\omega} \in \Omega_v,\ \exists\, i \in \{1, \ldots, \boldsymbol{m}_u\},\ \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i]} \neq \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i-1]}$$

i.e., there exists a state where the sole change of the regulator u triggers a change of the regulated variable v.

As we are generally interested in all parametrisations that satisfy a particular constraint, we define sets of parametrisation restricted to influence constraints.

Definition 5.2 (Concrete Parametrisation Set Satisfying an Influence Constraint). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $r \in I \times \{+1, -1, \mathrm{o}\}$ be an arbitrary influence constraint.

Then, the set of parametrisations $\mathcal{P}_r$ satisfying the influence constraint r is defined as follows:

$$\mathcal{P}_{(u,v,+1)} \triangleq \left\{ \boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}}) \;\middle|\; \begin{array}{l} \forall \boldsymbol{\omega} \in \Omega_v,\ \forall i \in \{1, \ldots, \boldsymbol{m}_u\}, \\ \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i]} \geq \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i-1]} \end{array} \right\}$$

$$\mathcal{P}_{(u,v,-1)} \triangleq \left\{ \boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}}) \;\middle|\; \begin{array}{l} \forall \boldsymbol{\omega} \in \Omega_v,\ \forall i \in \{1, \ldots, \boldsymbol{m}_u\}, \\ \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i]} \leq \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i-1]} \end{array} \right\}$$

$$\mathcal{P}_{(u,v,\mathrm{o})} \triangleq \left\{ \boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}}) \;\middle|\; \begin{array}{l} \exists \boldsymbol{\omega} \in \Omega_v,\ \exists i \in \{1, \ldots, \boldsymbol{m}_u\}, \\ \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i]} \neq \boldsymbol{P}_{v,\boldsymbol{\omega}[u \mapsto i-1]} \end{array} \right\}$$

The definition naturally extends to sets of influence constraints by intersection.

Definition 5.3 (Concrete Parametrisation Set Satisfying Influence Constraints). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $R \subseteq I \times \{+1, -1, \mathrm{o}\}$ be a well-formed influence constraint set.

Then, the set of parametrisations satisfying the influence constraints in R is defined as follows:

$$\mathcal{P}_R \triangleq \bigcap_{r \in R} \mathcal{P}_r$$

The parametrisation restrictions imposed by influence constraints are no longer constraints on a single parameter, but rather inequality constraints on parameter values. The intersection based construction, however, aligns with the representation of parametrisation sets enabling a transition set. Indeed, intersections are sufficient to express the combination in the form of parametrisation sets enabling a transition set while satisfying a constraint set as well.

Definition 5.4 (Concrete Parametrisation Set Enabling a Transition Set and Satisfying Influence Constraint Set). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network with semantics $\xrightarrow{G_{\boldsymbol{m}}} \subseteq \xrightarrow[gen]{G_{\boldsymbol{m}}}$ and let $R \subseteq I \times \{+1, -1, \mathrm{o}\}$ be a well-formed influence constraint set.

Then the set of parametrisations $p_R(T)$ enabling a transition set $T \subseteq \bigcup_{\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})} \xrightarrow{F_{\boldsymbol{P}}}$ and satisfying the constraint set R is defined as follows:

$$p_R(T) \triangleq p(T) \cap \mathcal{P}_R$$

The seamless inclusion of influence constraints in the parametrisation sets enabling transition sets facilitates the definition of influence constraint aware semantics for parametric regulatory networks. The influence constraints are, moreover, defined for influence graph and are therefore global for the whole parametric regulatory network. The same applies to the resulting parametrisation set satisfying the constraint set. Thanks to this, instead of using parametrisation sets as per Definition 5.4 explicitly, the *constrained semantics* of parametric regulatory network can be defined to the same effect simply by restricting the concrete semantics to subsets of the parametrisation set satisfying the influence constraints.

Definition 5.5 (Constrained Semantics of Parametric Regulatory Networks). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network, let $\xrightarrow{G_{\boldsymbol{m}}} \subseteq \xrightarrow[gen]{G_{\boldsymbol{m}}}$ be a type of parametric regulatory network semantics and let $R \subseteq I \times \{+1, -1, \mathrm{o}\}$ be a well formed influence constraints of the influence graph G.

Then, the constrained semantics of the parametric regulatory network $G_{\boldsymbol{m}}$ is the relation $\xrightarrow[R]{G_{\boldsymbol{m}}} \subseteq \left(X_{\boldsymbol{m}} \times 2^{\mathcal{P}_R}\right) \times \left(X_{\boldsymbol{m}} \times 2^{\mathcal{P}_R}\right)$ defined as:

$$(x, \mathcal{P} \cap \mathcal{P}_R) \xrightarrow[R]{G_{\boldsymbol{m}}} (y, \mathcal{P}' \cap \mathcal{P}_R) \overset{\Delta}{\Longleftrightarrow} (x, \mathcal{P}) \xrightarrow{G_{\boldsymbol{m}}} (y, \mathcal{P}') \wedge \mathcal{P}' \cap \mathcal{P}_R \neq \emptyset$$

As illustrated by Definition 5.5, influence constraints can only limit the semantics of parametric regulatory networks and therefore cannot introduce new behaviour.

5.2 Abstract Constrained Semantics of Parametric Regulatory Networks

Introduction of influence constraints can, often significantly, reduce the number of parametrisations that need to be considered. However, the induced reduction of parametrisation space is not asymptotic. The parametrisation sets thus not only remain exponentially large in the general case, the structure of the parametrisation sets imposed by the influence constraints is highly nontrivial. Considering only the monotonicity constraints, $\mathcal{P}_{R(G)}$ can be used to enumerate all monotonic Boolean functions of a given dimension. Even counting monotonic Boolean functions, however, is known to be a hard problem [66].

To avoid explicit representation of parametrisation sets, we rely once again on the abstract semantics of parametric regulatory networks (Definition 4.13). We implement the influence constraints $r \in R(G)$ by the means of a narrowing operator σ_r on the parametrisation lattices. While the narrowing operator

produces over-approximation of the concrete parametrisation sets, we show the over-approximation to be optimal in terms of bounded convex sublattices.

Due to fundamental differences in the nature of the monotonicity and observability constraints (universal versus existential quantification), we treat the definition of σ_r separately depending on r being monotonicity of observability constraint. First, let us consider monotonicity constraints.

Definition 5.6 (Monotonicity Constraint Narrowing of Abstract Parametrisation Sets). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $r = (u, v, s) \in R(G)$ where $s \in \{+1, -1\}$ be an arbitrary monotonicity constraint.

Then, the narrowing operator σ_r is defined as:

$$\sigma_r \colon P^\sharp(G_{\boldsymbol{m}}) \to P^\sharp(G_{\boldsymbol{m}})$$

$$\sigma_r \colon [\mathbf{0}, \mathbf{1}] \mapsto [{f_0}^*(\mathbf{0}), {f_1}^*(\mathbf{1})]$$

where the functions $f_0, f_1 \colon \mathbb{P}(G_{\boldsymbol{m}}) \to \mathbb{P}(G_{\boldsymbol{m}})$ are defined as follows:

$$f_0 \colon \boldsymbol{P} \mapsto \boldsymbol{P} \vee \bigvee_{\boldsymbol{\omega} \in \Omega_v} \boldsymbol{P}\left[(v, \boldsymbol{\omega}) \mapsto \boldsymbol{P}_{v, \boldsymbol{\omega}[u \mapsto \boldsymbol{\omega}_u - s]}\right]$$

$$f_1 \colon \boldsymbol{P} \mapsto \boldsymbol{P} \wedge \bigwedge_{\boldsymbol{\omega} \in \Omega_v} \boldsymbol{P}\left[(v, \boldsymbol{\omega}) \mapsto \boldsymbol{P}_{v, \boldsymbol{\omega}[u \mapsto \boldsymbol{\omega}_u + s]}\right]$$

Since both of the functions f_0, f_1 are monotonic in $\leq_{G_{\boldsymbol{m}}}$, the fixed points ${f_0}^*, {f_1}^*$ are guaranteed to exist for any input. Moreover, the restriction of parameter values by f_0, f_1 happens progressively in the direction of the monotonicity constraint on the influence (u, v). E.g. assuming (u, v) to be an activation, $s = +1$, increasing the lower bound of $\boldsymbol{\omega}$ of v by f_0 leads to increase of the lower bound also for the $\boldsymbol{\omega}[u \mapsto \boldsymbol{\omega}_u + 1]$ in the subsequent iteration of f_0, if necessary, etc. We formalise this concept by the means of a partial order on regulator states of individual variables, called *monotonicity order*.

Definition 5.7 (Monotonicity Order). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network, R a well-formed influence constraint set and $v \in V$ and arbitrary variable of $G_{\boldsymbol{m}}$.

Then, the monotonicity order on Ω_v is the partial order $\preceq_{v,R} \subseteq \Omega_v \times \Omega_v$ defined as:

$$\boldsymbol{\omega} \preceq_{v,R} \boldsymbol{\omega}' \overset{\Delta}{\Longleftrightarrow} \forall (u, v, s) \in R,\ sign({\boldsymbol{\omega}'}_u - \boldsymbol{\omega}_u) \in \{0, s\}$$

We write $\boldsymbol{\omega} \parallel_{v,R} \boldsymbol{\omega}'$ if and only if $\boldsymbol{\omega}$ and $\boldsymbol{\omega}'$ are not comparable according to $\preceq_{v,R}$. This is the case notably when $\boldsymbol{\omega}_u \neq {\boldsymbol{\omega}'}_u$ for some $u \in R(v)$ such that the influence (u, v) is not monotonic in R, $\{(u, v, +1), (u, v, -1)\} \cap R = \emptyset$.

To ease notation, we write simply $\preceq_v$ instead of $\preceq_{v,R}$ when the entire influence constraint set $R = R(G)$ is considered.

In terms of the monotonicity order, the narrowing operator simply adjusts the parameter values in $\mathbf{0}$ to maximum value of the parameters associated with $\preceq_{v,\{(u,v,+1)\}}$-smaller regulator states and the parameter values in $\mathbf{1}$ to minimum value of $\preceq_{v,\{(u,v,+1)\}}$-larger regulator states. An analogical operation with the maximums and minimum reversed is done for negative monotonic influence constraints. More formally, the monotonicity order allows for an alternative definition of the monotonicity influence constraint narrowing:

$$\sigma_{(u,v,s)}[\mathbf{0},\mathbf{1}] \triangleq \left[\begin{array}{l} \blacktriangleleft\!\!\prod\limits_{(w,\boldsymbol{\omega})\in\Omega} \left\{ \begin{array}{ll} \max\limits_{\{\boldsymbol{\omega}'\in\Omega_v \mid \boldsymbol{\omega}'\preceq_{v,\{(u,v,s)\}}\boldsymbol{\omega}\}} (\mathbf{0}_{v,\boldsymbol{\omega}'}) & \text{if } w = v \\ \mathbf{0}_{w,\boldsymbol{\omega}} & \text{otherwise} \end{array} \right\}, \\ \blacktriangleleft\!\!\prod\limits_{(w,\boldsymbol{\omega})\in\Omega} \left\{ \begin{array}{ll} \min\limits_{\{\boldsymbol{\omega}'\in\Omega_v \mid \boldsymbol{\omega}'\succeq_{v,\{(u,v,s)\}}\boldsymbol{\omega}\}} (\mathbf{1}_{v,\boldsymbol{\omega}'}) & \text{if } w = v \\ \mathbf{1}_{w,\boldsymbol{\omega}} & \text{otherwise} \end{array} \right\} \end{array} \right]$$

Note that by iterating over $\boldsymbol{\omega} \in \Omega_v$ in increasing, respectively decreasing, direction of $\preceq_{v,\{(u,v,s)\}}$, the maximum, respectively minimum, can be computed on the run, rather than explicitly for each parameter. The computation of the whole narrowing σ_r is thus linear in the number of parameters of the variable v ($\mathcal{O}(|\Omega_v|)$).

Since the narrowing operator only computes minimums and maximums, it is easily composable with narrowing operators for other monotonicity influence constraints on the same variable. It is enough to take the smaller of the minimums, respectively larger of the maximums, obtained for two different monotonicity constraints and the same parameter. This comparison is moreover automatically included once monotonicity order over both of the monotonicity constraints is considered. Indeed, replacing the order $\preceq_{v,\{(u,v,s)\}}$ in the above definition by $\preceq_{v,R}$ for arbitrary $R \subseteq R(G)$ is enough to compute the narrowing operator of all monotonicity constraints in R at the same time, while keeping the linear complexity. Of particular interest is then the narrowing operator σ_v using the full monotonicity order $\preceq_v$ which allows us the compute the monotonicity narrowing for all monotonicity influence constraints on the variable v at the same time.

Unlike for the monotonicity influence constraints, which introduce universal inequality constraints on the parameter values for individual regulator states, the observability influence constraint is existential, yielding no global inequality constraints on parameter values. Indeed, to ensure observability, the parametrisations which do not satisfy the influence constraint have to be removed on individual basis. Due to the nature of the abstract parametrisation lattices, namely the convexity, it is impossible to individually treat parametrisations unless they happen to be the lower or upper bounds, $\mathbf{0}$ and $\mathbf{1}$ respectively. As such, any parametrisations $\mathbf{0} <_{G_{\boldsymbol{m}}} \boldsymbol{P} <_{G_{\boldsymbol{m}}} \mathbf{1}$ that do not satisfy the observability influence constraints are ignored, at the cost of over-approximation.

The narrowing operator $\sigma_{(u,v,\mathrm{o})}$ therefore translates into checking whether the observability influence constraint (u, v, o) is satisfied under both of the extreme cases of $\mathbf{0}$ and $\mathbf{1}$. By negation of the observability influence constraint condition as given in Definition 5.2, (u, v, o) is not satisfied under $\mathbf{0}$, respectively

1, if for each $\boldsymbol{\omega} \in \Omega_v$ and each $i \in \{0, \ldots, \boldsymbol{m}_u\}$:

$$\mathbf{0}_{v,\boldsymbol{\omega}} = \mathbf{0}_{v,\boldsymbol{\omega}[u \mapsto i]} \quad \text{respectively:} \quad \mathbf{1}_{v,\boldsymbol{\omega}} = \mathbf{1}_{v,\boldsymbol{\omega}[u \mapsto i]}$$

Once the influence (u, v) is determined to be unobservable under $\mathbf{0}$, respectively $\mathbf{1}$, increasing $\mathbf{0}_{v,\boldsymbol{\omega}}$, respectively decreasing $\mathbf{1}_{v,\boldsymbol{\omega}}$, for any $\boldsymbol{\omega} \in \Omega_v$ ensures the observability of (u, v) under the new parametrisation. In fact, once a value is changed for one regulator state, all influences are guaranteed to be observable under the new parametrisation unless values for each other regulator state differing only in the value of the corresponding regulator are adjusted to match the new value. This introduces a measure of *distance* between unobservable parametrisations, formally captured in Lemma 5.1.

Lemma 5.1 (Unobservable Parametrisation Distance)**.** *Let $G_{\boldsymbol{m}}$ be a parametric regulatory network, $r = (u, v, o) \in R(G)$ an arbitrary observability influence constraint and $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$ be such that the influence (u, v) is not observable under $\boldsymbol{P}$.*

Then, for any $\boldsymbol{\omega} \in \Omega_v$, $i \in \{0, \ldots, \boldsymbol{P}_{v,\boldsymbol{\omega}} - 1, \boldsymbol{P}_{v,\boldsymbol{\omega}} + 1, \ldots, \boldsymbol{m}_v\}$ and all $u' \in R(v)$, the influence (u', v) is observable under $\boldsymbol{P}' = \boldsymbol{P}[(v, \boldsymbol{\omega}) \mapsto i]$.

Proof. Let $\boldsymbol{\omega} \in \Omega_v, i \in \{0, \ldots, \boldsymbol{m}_v\}$ be arbitrary such that $\boldsymbol{P}_{v,\boldsymbol{\omega}} \neq i$.

Let $\boldsymbol{P}' = \boldsymbol{P}[v, \boldsymbol{\omega} \mapsto i]$ denote the modified parametrisation and $\hat{\boldsymbol{\omega}}$ denote a regulator state of variable v differing from $\boldsymbol{\omega}$ in the value of u,

$$\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}[u \mapsto \boldsymbol{\omega}_u + j] \text{ where } j = \begin{cases} 1 & \text{if } \boldsymbol{\omega}_u = 0 \\ -1 & \text{otherwise} \end{cases}$$

For the influence constraint r we get $\boldsymbol{P}'_{v,\hat{\boldsymbol{\omega}}} = \boldsymbol{P}_{v,\hat{\boldsymbol{\omega}}} = \boldsymbol{P}_{v,\boldsymbol{\omega}} \neq \boldsymbol{P}'_{v,\boldsymbol{\omega}}$ and thus $\boldsymbol{P}' \in \mathcal{P}_r$. Now let us assume v has at least two (observable) influences and let $r' = (w, v, \mathrm{o}) \in R(G)$ be arbitrary such that $w \neq u$.

First, we introduce two additional regulator states. The regulator state $\boldsymbol{\omega}'$ identical to $\boldsymbol{\omega}$ up to the value w and the regulator state $\hat{\boldsymbol{\omega}}'$ identical to $\hat{\boldsymbol{\omega}}$ up to the value of w,

$$\begin{aligned} \boldsymbol{\omega}' &= \boldsymbol{\omega}[w \mapsto \boldsymbol{\omega}_w + k] \\ \hat{\boldsymbol{\omega}}' &= \hat{\boldsymbol{\omega}}[w \mapsto \hat{\boldsymbol{\omega}}_w + k] \end{aligned} \quad \text{where } k = \begin{cases} 1 & \text{if } \boldsymbol{\omega}_w = 0 \\ -1 & \text{otherwise} \end{cases}$$

We use the four regulator states $\boldsymbol{\omega}, \hat{\boldsymbol{\omega}}, \boldsymbol{\omega}'$ and $\hat{\boldsymbol{\omega}}'$ to prove the influence (w, v) is indeed observable under $\boldsymbol{P}'$. To do this, we need to show either $\boldsymbol{P}'_{v,\boldsymbol{\omega}} \neq \boldsymbol{P}'_{v,\boldsymbol{\omega}'}$ or $\boldsymbol{P}'_{v,\hat{\boldsymbol{\omega}}} \neq \boldsymbol{P}'_{v,\hat{\boldsymbol{\omega}}'}$ as both $\boldsymbol{\omega}, \boldsymbol{\omega}'$ and $\hat{\boldsymbol{\omega}}, \hat{\boldsymbol{\omega}}'$ differ only in the value of w. The proof also relies on the analogous proximity of $\boldsymbol{\omega}, \hat{\boldsymbol{\omega}}$ and $\boldsymbol{\omega}', \hat{\boldsymbol{\omega}}'$, which differ only in the value of u.

We know $\boldsymbol{P}'_{v,\boldsymbol{\omega}} = i$ and $\boldsymbol{P}'_{v,\boldsymbol{\omega}'} = \boldsymbol{P}_{v,\boldsymbol{\omega}'}$. Thus, if $i \neq \boldsymbol{P}_{v,\boldsymbol{\omega}'}$, the result is trivial.

Let us therefore assume $\boldsymbol{P}_{v,\boldsymbol{\omega}'} = i = \boldsymbol{P}'_{v,\boldsymbol{\omega}}$. Since (u, v) is not observable under $\boldsymbol{P}$, $\boldsymbol{P} \notin \mathcal{P}_r$, we have $\boldsymbol{P}_{v,\boldsymbol{\omega}} = \boldsymbol{P}_{v,\hat{\boldsymbol{\omega}}}$ and $\boldsymbol{P}_{v,\boldsymbol{\omega}'} = \boldsymbol{P}_{v,\hat{\boldsymbol{\omega}}'}$. $\boldsymbol{P}'$ only differs from

$\boldsymbol{P}$ in the value of $\boldsymbol{\omega}$. We can thus expand the prior observation to obtain $\boldsymbol{P}'_{v,\boldsymbol{\omega}} \neq \boldsymbol{P}_{v,\boldsymbol{\omega}} = \boldsymbol{P}_{v,\hat{\boldsymbol{\omega}}} = \boldsymbol{P}'_{v,\hat{\boldsymbol{\omega}}}$ and $\boldsymbol{P}'_{v,\boldsymbol{\omega}'} = \boldsymbol{P}_{v,\boldsymbol{\omega}'} = \boldsymbol{P}_{v,\hat{\boldsymbol{\omega}}'} = \boldsymbol{P}'_{v,\hat{\boldsymbol{\omega}}'}$. By our assumption, $\boldsymbol{P}_{v,\boldsymbol{\omega}'} = \boldsymbol{P}'_{v,\boldsymbol{\omega}}$, we obtain the coveted $\boldsymbol{P}'_{v,\hat{\boldsymbol{\omega}}'} = \boldsymbol{P}_{v,\boldsymbol{\omega}'} = \boldsymbol{P}'_{v,\boldsymbol{\omega}} \neq \boldsymbol{P}'_{v,\hat{\boldsymbol{\omega}}}$. □

While Lemma 5.1 guarantees that a single value change is sufficient to ensure observability under $\mathbf{0}$, respectively $\mathbf{1}$, for all influences of v, the value change may not be desirable for every $\boldsymbol{\omega} \in \Omega_v$. We therefore identify regulator states that are *open for value change* (or simply *open*) as regulator states $\boldsymbol{\omega} \in \Omega_v$ such that increasing $\mathbf{0}_{v,\boldsymbol{\omega}}$, respectively decreasing $\mathbf{1}_{v,\boldsymbol{\omega}}$, does not break any monotonicity constraints in $R(G)$ and does not result in an empty parametrisation lattice.

Definition 5.8 (Open Regulator State For Observability Enforcement). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network of dimension n and let $R(G)$ be a well-formed set of influence constraints. Let further $[\mathbf{0},\mathbf{1}] \in P^\sharp(G_{\boldsymbol{m}})$ and $v \in \{1,\ldots,n\}$ be arbitrary.

Then a regulator state $\boldsymbol{\omega} \in \Omega_v$ is open to value increase in $[\mathbf{0},\mathbf{1}]$ if $\mathbf{0}_{v,\boldsymbol{\omega}} < \mathbf{1}_{v,\boldsymbol{\omega}}$ and for all $\boldsymbol{\omega}' \in \Omega_v$ such that $\boldsymbol{\omega}' \succ_v \boldsymbol{\omega}$, $\mathbf{0}_{v,\boldsymbol{\omega}'} > \mathbf{0}_{v,\boldsymbol{\omega}}$.

Similarly, a regulator state $\boldsymbol{\omega} \in \Omega_v$ is open to value decrease in $[\mathbf{0},\mathbf{1}]$ if $\mathbf{0}_{v,\boldsymbol{\omega}} < \mathbf{1}_{v,\boldsymbol{\omega}}$ and for all $\boldsymbol{\omega}' \in \Omega_v$ such that $\boldsymbol{\omega}' \prec_v \boldsymbol{\omega}$, $\mathbf{1}_{v,\boldsymbol{\omega}'} < \mathbf{1}_{v,\boldsymbol{\omega}}$.

We write $O_v^+([\mathbf{0},\mathbf{1}]) = \{\,\boldsymbol{\omega} \in \Omega_v \mid \mathbf{0}_{v,\boldsymbol{\omega}} < \mathbf{1}_{v,\boldsymbol{\omega}} \wedge \forall \boldsymbol{\omega}' \succ_v \boldsymbol{\omega},\ \mathbf{0}_{v,\boldsymbol{\omega}'} > \mathbf{0}_{v,\boldsymbol{\omega}}\,\}$ to denote the set of all regulator states of v open to value increase in $[\mathbf{0},\mathbf{1}]$. Analogically, $O_v^-([\mathbf{0},\mathbf{1}]) = \{\,\boldsymbol{\omega} \in \Omega_v \mid \mathbf{0}_{v,\boldsymbol{\omega}} < \mathbf{1}_{v,\boldsymbol{\omega}} \wedge \forall \boldsymbol{\omega}' \prec_v \boldsymbol{\omega},\ \mathbf{1}_{v,\boldsymbol{\omega}'} < \mathbf{1}_{v,\boldsymbol{\omega}}\,\}$ is used to denote the set of all regulator states of v open to value decrease in $[\mathbf{0},\mathbf{1}]$.

The action taken by the narrowing operator depends on the regulator states open in the parametrisation lattice $[\mathbf{0},\mathbf{1}]$. If no regulator states are open, an empty state is returned to reflect that the influence u, v is not observable under any parametrisation in $[\mathbf{0},\mathbf{1}]$. If, on the other hand, more than one regulator state is open to value increase, respectively value decrease, no values are increased, respectively decreased, to preserve all possibilities at the cost of overapproximation. The value is only restricted if a unique regulator state is open to value increase, respectively decrease.

Definition 5.9 (Observability Constraint Narrowing of Abstract Parametrisation Sets). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $r = (u, v, \mathrm{o}) \in R(G)$ be an arbitrary observability influence constraint.

Then, the narrowing operator σ_r is defined as:

$$\sigma_r \colon P^\sharp(G_{\boldsymbol{m}}) \to P^\sharp(G_{\boldsymbol{m}})$$

$$\sigma_r \colon [\mathbf{0},\mathbf{1}] \mapsto \begin{cases} \varnothing & \text{if } (\mathbf{0} \notin \mathcal{P}_r \wedge O_v^+([\mathbf{0},\mathbf{1}]) = \emptyset) \vee (\mathbf{1} \notin \mathcal{P}_r \wedge O_v^-([\mathbf{0},\mathbf{1}]) = \emptyset) \\ [\mathbf{0}',\mathbf{1}'] & \text{otherwise} \end{cases}$$

where

$$\mathbf{0}' = \begin{cases} \mathbf{0}\,[(v,\boldsymbol{\omega}) \mapsto \mathbf{0}_{v,\boldsymbol{\omega}} + 1] & \text{if } \mathbf{0} \notin \mathcal{P}_r \wedge O_v^+([\mathbf{0},\mathbf{1}]) = \{\boldsymbol{\omega}\} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\mathbf{1}' = \begin{cases} \mathbf{1}\left[(v,\boldsymbol{\omega}) \mapsto \mathbf{1}_{v,\boldsymbol{\omega}} - 1\right] & \text{if } \mathbf{1} \notin \mathcal{P}_r \wedge O_v^-\left([\mathbf{0},\mathbf{1}]\right) = \{\boldsymbol{\omega}\} \\ \mathbf{1} & \text{otherwise} \end{cases}$$

Determining the observability of the influence u, v under $\mathbf{0}$ and $\mathbf{1}$ has linear complexity with respect to the number of parameters of variable v ($\mathcal{O}\left(|\Omega_v|\right)$). By iterating over regulator states in Ω_v in decreasing order of $\preceq_v$, respectively increasing order of $\preceq_v$, the open regulator state set $O_v^+\left([\mathbf{0},\mathbf{1}]\right)$, respectively $O_v^-\left([\mathbf{0},\mathbf{1}]\right)$, can be computed with the same linear complexity. Computing the narrowing operator $\sigma_{(u,v,\star)}$ thus has complexity in $\mathcal{O}\left(|\Omega_v|\right)$ for both monotonicity and observability influence constraints.

By aggregating the narrowing operators for all the individual influence constraints we obtain a global narrowing operator $\sigma_{R(G)}$. As observability narrowing respects the monotonicity constraints thanks to the use of the monotonicity order, the global narrowing is definable simply as a function composition.

Definition 5.10 (Influence Constraint Set Narrowing). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network of dimension n and $R(G)$ a well-formed set of influence constraints.

Then the global influence constraint narrowing operator $\sigma_{R(G)} \colon P^\sharp\left(G_{\boldsymbol{m}}\right) \to P^\sharp\left(G_{\boldsymbol{m}}\right)$ is defined as a function composition:

$$\sigma_{R(G)} = \bigcirc_{(u,v,\mathrm{o}) \in R(G)} \sigma_{\{(u,v,\mathrm{o})\}} \circ \bigcirc_{v \in \{1,\dots,n\}} \sigma_v$$

The constrained abstract semantics are defined using a combination of the influence constraint narrowing and the transition narrowing, Definition 4.14.

Definition 5.11 (Constrained Abstract Parametric Regulatory Network Semantics). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network of dimension n and let $R(G)$ be a well-formed set of influence constraints.

Then the constrained abstract parametric regulatory network semantics of $G_{\boldsymbol{m}}$ is a relation $\xrightarrow[R(G)\cdot abs]{G_{\boldsymbol{m}}} G_{\boldsymbol{m}} \subseteq \left(X_{\boldsymbol{m}} \times P^\sharp\left(G_{\boldsymbol{m}}\right)\right) \times \left(X_{\boldsymbol{m}} \times P^\sharp\left(G_{\boldsymbol{m}}\right)\right)$ defined as follows:

$$\begin{aligned} &(x, [\mathbf{0},\mathbf{1}]) \times \left(y, \left[\mathbf{0}', \mathbf{1}'\right]\right) \in \xrightarrow[R(G)\cdot abs]{G_{\boldsymbol{m}}} G_{\boldsymbol{m}} \overset{\Delta}{\Longleftrightarrow} \\ &\exists\, t \in \xrightarrow[abs]{G_{\boldsymbol{m}}}, \quad t = (x, [\mathbf{0},\mathbf{1}]) \xrightarrow[abs]{F_{\boldsymbol{P}}} \left(y, \left[\mathbf{0}'', \mathbf{1}''\right] \supseteq \left[\mathbf{0}', \mathbf{1}'\right]\right) \wedge \\ &\qquad\qquad \left[\mathbf{0}', \mathbf{1}'\right] = \sigma_{R(G)} \circ \sigma_t\left([\mathbf{0},\mathbf{1}]\right) \end{aligned}$$

To ease notation, we use $p_R^\sharp\left(T\right)$ to denote the over-approximation of the set of all parametrisations enabling all transitions in T while satisfying the influence constraints in R,

$$p_R^\sharp\left(T\right) = \bigcirc_{t \in T} \left(\sigma_R \circ \sigma_t\right)\left(\left[\mathbb{P}\left(G_{\boldsymbol{m}}\right)\right]\right) = \sigma_R \circ \bigcirc_{t \in T} \sigma_t\left(\left[\mathbb{P}\left(G_{\boldsymbol{m}}\right)\right]\right)$$

The constrained abstract semantics of parametric regulatory networks result in an over-approximation of the parametrisation sets, introducing false positives into reachable state space. We can show, however, that the false positives cannot introduce spurious behaviour, i.e. no transition is included unless it is supported by at least one true positive parametrisation. This is a natural consequence of the tightness of the abstraction.

Theorem 5.1 (Abstraction Computed by Influence Constraint Set Narrowing is Tight)**.** *Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let $R = R(G)$ be a well-formed set of influence constraints.*

Then for arbitrary set of transitions $T \subseteq \xrightarrow[R \cdot abs]{G_{\boldsymbol{m}}}$, $p_R^\sharp(T)$ is the smallest convex cover of $p_R(T)$, $p_R^\sharp(T) = [p_R(T)]$.

Proof. Note that marking an influence of a variable $v \in \{1, \ldots, n\}$ as either monotonic or observable does not influence other variables $u \neq v$. We therefore conduct the proof for single variable only, allowing us to limit our attention to regulator states $\boldsymbol{\omega} \in \Omega_v$ while maintaining universal applicability.

We conduct the proof of $p_R^\sharp(T) = [p_R(T)]$ by mathematical induction on the transition set T. This corresponds to the actual application of the narrowing operators, as transitions are generally explored one at a time.

Base step ($T = \emptyset$):

By definition, we have $p_R^\sharp(\emptyset) = \sigma_R([\mathbb{P}(G_{\boldsymbol{m}})])$. σ_r for either monotonicity or observability constraints only restricts the parametrisation lattice in case r is not satisfied under $\mathbf{0}$ or $\mathbf{1}$. In the beginning, $[\mathbf{0}, \mathbf{1}] = [\mathbb{P}(G_{\boldsymbol{m}})]$ with $\mathbf{0}_{v,\boldsymbol{\omega}} = 0$ and $\mathbf{1}_{v,\boldsymbol{\omega}} = \boldsymbol{m}_v$ for every $\boldsymbol{\omega} \in \Omega_v$. Thus, any monotonicity constraint on influence of v is necessarily satisfied under both $\mathbf{0}$ and $\mathbf{1}$.

Let us now consider there exists at least one observability constraint $r = (u, v, \mathrm{o}) \in R$ on an influence of variable v. The influence (u, v) is not observable under $\mathbf{0}$ and $\mathbf{1}$ and the parametrisation set may thus get restricted by σ_r.

The result of σ_r depends on the sets of regulator states open for observability enforcement. All regulator states are assigned the value 0 in $\mathbf{0}$, $O_v^+([\mathbf{0}, \mathbf{1}])$ thus contains exactly the $\preceq_v$-maximal regulator states. Surely, at least one such regulator state must exist, giving us $\sigma_r([\mathbf{0}, \mathbf{1}]) \neq \varnothing$. Similarly, the $O_v^-([\mathbf{0}, \mathbf{1}])$ contains exactly the $\preceq_v$-minimal regulator states. By definition, $\sigma_r([\mathbf{0}, \mathbf{1}]) = [\mathbf{0}, \mathbf{1}]$, if there are at least two $\preceq_v$-maximal and at least two $\preceq_v$-minimal regulator states. Since the $\preceq_v$ is always isomorphic to its dual, the reverse $\preceq_v$, the number of $\preceq_v$-minimal and $\preceq_v$-maximal regulator states is always the same. Moreover, regulator states are only incomparable with each other in the $\preceq_v$, if they differ on a value of a non-monotonic regulator. Both The number of $\preceq_v$-maximal and $\preceq_v$-minimal regulator states is therefore exactly 2 to the power of the number of non-monotonic influences of v. As such, $\sigma_r([\mathbf{0}, \mathbf{1}]) \neq [\mathbf{0}, \mathbf{1}]$ if and only if all the influences of v are monotonic.

Assuming thus, there exists a non-monotonic influence $(w, v) \in I$, $p^\sharp(\emptyset) = [\mathbb{P}(G_{\boldsymbol{m}})] = [\mathbf{0}, \mathbf{1}]$. Having two distinct $\preceq_v$-maximal elements $\overline{\boldsymbol{\omega}}, \overline{\boldsymbol{\omega}}' \in \Omega_v$ gives us two parametrisations $\overline{\boldsymbol{P}} = \mathbf{0}[\overline{\boldsymbol{\omega}} \mapsto 1]$ and $\overline{\boldsymbol{P}}' = \mathbf{0}[\overline{\boldsymbol{\omega}}' \mapsto 1]$. Both $\overline{\boldsymbol{P}}, \overline{\boldsymbol{P}}' \in$

$p_R(\emptyset)$ as the satisfaction of monotonicity constraints comes from $\overline{\boldsymbol{\omega}},\overline{\boldsymbol{\omega}}'$ being $\preceq_v$-maximal and the satisfaction of observability constraints is thanks to Lemma 5.1. The construction for $\preceq_v$-minimal regulator states and $\mathbf{0}$ is symmetrical, thus $[p_R(\emptyset)] = [\mathbf{0},\mathbf{1}] = p^\sharp(\emptyset)$.

Let us now consider the situation where the $\preceq_v$-maximal element $\overline{\boldsymbol{\omega}} \in \Omega_v$ is unique. Then, $\overline{\boldsymbol{\omega}}$ is also the only regulator state open for value increase for the purpose of the observability constraint narrowing, $O_v^+([\mathbf{0},\mathbf{1}]) = \{\overline{\boldsymbol{\omega}}\}$. Symmetrically, the unique $\preceq_v$-minimal element $\underline{\boldsymbol{\omega}}$ is also the unique regulator state open for value decrease, $O_v^-([\mathbf{0},\mathbf{1}]) = \{\underline{\boldsymbol{\omega}}\}$. The influence constraint narrowing therefore restricts both the minimal and maximal parametrisations, $\sigma_R([\mathbf{0},\mathbf{1}]) = [\mathbf{0}[v,\overline{\boldsymbol{\omega}} \mapsto 1], \mathbf{1}[v,\underline{\boldsymbol{\omega}} \mapsto \boldsymbol{m}_v - 1]]$.

For the concrete set, we know $\mathbf{0},\mathbf{1} \notin p_R(\emptyset)$ due to the observability influence constraint. Let now $\boldsymbol{P} \in p_R(\emptyset)$ be any parametrisation such that $\boldsymbol{P}_{v,\boldsymbol{\omega}} > 0$ for some $\boldsymbol{\omega} \in \Omega_v$. We know $\overline{\boldsymbol{\omega}} \succeq_v \boldsymbol{\omega}$ thanks to all influences being monotonic giving us $\boldsymbol{P}_{v,\overline{\boldsymbol{\omega}}} \geq \boldsymbol{P}_{v,\boldsymbol{\omega}} > 0$. A symmetrical argument can be made for $\underline{\boldsymbol{\omega}}$ always being smaller than the maximum $\boldsymbol{m}_v$. Thus, $[p_R(\emptyset)] = [\mathbf{0}[v,\overline{\boldsymbol{\omega}} \mapsto 1], \mathbf{1}[v,\underline{\boldsymbol{\omega}} \mapsto \boldsymbol{m}_v - 1]] = p_R^\sharp(\emptyset)$.

Induction hypothesis: $p_R^\sharp(T) = [p_R(T)]$ for any set of transitions T such that $|T| \leq k$ where $k \in \mathbb{N}_0$.

We now show $p_R^\sharp(T \cup \{t\}) = [p_R(T \cup \{t\})]$ for arbitrary transition $t \notin T$. We prove the lattice equality as the two lattices being mutual sublattices of each other. Moreover, as both of the lattices use the same order and elements from the same superset, the sublattice relation is equivalent to subset relation, we thus liberally treat the lattices as sets throughout the proof.

First, we show $[p_R(T \cup \{t\})] \subseteq p_R^\sharp(T \cup \{t\})$ (soundness of the abstraction).

If $p_R(T \cup \{t\}) = \emptyset$, the resulting convex cover is also empty, $[p_R(T \cup \{t\})] = \varnothing \subseteq p_R^\sharp(T \cup \{t\})$, which is in turn surely a sublattice of the abstract parametrisation set. We now assume $p_R(T \cup \{t\}) \neq \emptyset$.

By Definition 5.10 and Definition 5.11, the computation of $p_R^\sharp(T \cup \{t\}) = \bigcirc_{(u,v,o)\in R} \sigma_{\{(u,v,o)\}} \circ \bigcirc_{v\in\{1,\dots,n\}} \sigma_v \circ \sigma_t\left(p_R^\sharp(T)\right)$ is divided into three iterations of narrowing. Starting with the transition t and followed by monotonicity influence constraints and finally observability influence constraints. We follow this separation in the soundness proof.

We first show $[p_R(T \cup \{t\})] \subseteq \sigma_t\left(p_R^\sharp(T)\right)$. From Theorem 4.1 we have $[p(T \cup \{t\})] = p^\sharp(T \cup \{t\}) = \sigma_t\left(p^\sharp(T)\right)$. We start by intersecting both sides of the equation by $p_R^\sharp(T)$, thus obtaining $[p(T \cup \{t\})] \cap p_R^\sharp(T) = \sigma_t\left(p^\sharp(T)\right) \cap p_R^\sharp(T)$. Since both $[p_R(T \cup \{t\})] \subseteq [p(T \cup \{t\})]$ and by induction hypothesis $[p_R(T \cup \{t\})] \subseteq [p_R(T)] = p_R^\sharp(T)$, we replace the left hand side by $[p_R(T \cup \{t\})]$ changing the equality relation to a subset one, $[p_R(T \cup \{t\})] \subseteq \sigma_t\left(p^\sharp(T)\right) \cap p_R^\sharp(T)$. The restriction imposed by the narrowing operator σ_t does not depend on the actual parametrisation set. $\sigma_t\left(p^\sharp(T)\right)$ can thus be rewritten as $p^\sharp(T) \cap p(t)$. As $p^\sharp(T) \subseteq p_R^\sharp(T)$, the right hand side becomes $p_R^\sharp(T) \cap p(t) =$

$\sigma_t\left(p_R^\sharp(T)\right)$, giving us the coveted $[p_R(T\cup\{t\})]\subseteq\sigma_t\left(p_R^\sharp(T)\right)$.

What remains to be proven is that any restriction by σ_R on $\sigma_t\left(p_R^\sharp(T)\right)$ is reflected in $[p_R(T\cup\{t\})]$. We continue with the monotonicity influence constraint narrowing to first prove $[p_R(T\cup\{t\})]\subseteq\bigcirc\limits_{v\in\{1,\ldots,n\}}\sigma_v\circ\sigma_t\left(p_R^\sharp(T)\right)$.

Monotonicity constraint narrowing only imposes restrictions if the constraint is not satisfied by either of the limit parametrisations generating the abstract parametrisation set. Since all parametrisations in $p_R^\sharp(T)$ satisfy the monotonicity constraints, only prior restriction of a $v=v(t)$ parameter $K_{v,\boldsymbol{\omega}}$ for some $\boldsymbol{\omega}\in\Omega_v$ by σ_t may cause the necessary condition. It is therefore enough to consider σ_v.

Let now $[\mathbf{0},\mathbf{1}]=\sigma_t\left(p_R^\sharp(T)\right)$ and $\left[\mathbf{0}',\mathbf{1}'\right]=\sigma_v([\mathbf{0},\mathbf{1}])=\bigcirc\limits_{v\in\{1,\ldots,n\}}\sigma_v\circ\sigma_t\left(p_R^\sharp(T)\right)$ be the relevant parametrisation lattices and $\boldsymbol{P}\in p_R(T\cup\{t\})$ arbitrary. We now prove $\mathbf{0}'_{w,\boldsymbol{\omega}'}\leq\boldsymbol{P}_{w,\boldsymbol{\omega}'}\leq\mathbf{1}'_{w,\boldsymbol{\omega}'}$ for each $(w,\boldsymbol{\omega}')\in\Omega$.

Since $p_R(T\cup\{t\})\subseteq[\mathbf{0},\mathbf{1}]$, we have $\forall(w,\boldsymbol{\omega}')\in\Omega,\ \mathbf{0}_{w,\boldsymbol{\omega}'}\leq\boldsymbol{P}_{w,\boldsymbol{\omega}'}\leq\mathbf{1}_{w,\boldsymbol{\omega}'}$. By definition, $\mathbf{0}_{w,\boldsymbol{\omega}'}=\mathbf{0}'_{w,\boldsymbol{\omega}'}$ and $\mathbf{1}_{w,\boldsymbol{\omega}'}=\mathbf{1}'_{w,\boldsymbol{\omega}'}$ for any $(w,\boldsymbol{\omega}')$ with $w\neq v$.

Let thus $\boldsymbol{\omega}'\in\Omega_v$ be such that $\mathbf{0}_{v,\boldsymbol{\omega}'}<\mathbf{0}'_{v,\boldsymbol{\omega}'}$. Such restriction may only be due to $\mathbf{0}_{v,\boldsymbol{\omega}}$, giving us $\boldsymbol{\omega}'\succ_v\boldsymbol{\omega}$ and $\mathbf{0}'_{v,\boldsymbol{\omega}'}=\mathbf{0}_{v,\boldsymbol{\omega}}$. We know all influence constraints are satisfied under $\boldsymbol{P}$. $\boldsymbol{\omega}'\succ_v\boldsymbol{\omega}$ thus mandates $\boldsymbol{P}_{v,\boldsymbol{\omega}'}\geq\boldsymbol{P}_{v,\boldsymbol{\omega}}\geq\mathbf{0}_{v,\boldsymbol{\omega}}$.

Let us now consider $\boldsymbol{\omega}'\in\Omega_v$ to be such that $\mathbf{1}_{v,\boldsymbol{\omega}'}>\mathbf{1}'_{v,\boldsymbol{\omega}'}$. Again, the restriction is due to $\mathbf{1}_{v,\boldsymbol{\omega}}$, giving us $\boldsymbol{\omega}'\prec_v\boldsymbol{\omega}$ and $\mathbf{1}'_{v,\boldsymbol{\omega}'}=\mathbf{1}_{v,\boldsymbol{\omega}}$. And for $\boldsymbol{P}$ to satisfy the influence constraints, $\boldsymbol{P}_{v,\boldsymbol{\omega}'}\leq\boldsymbol{P}_{v,\boldsymbol{\omega}}\leq\mathbf{1}_{v,\boldsymbol{\omega}}$.

All parametrisations in the concrete set $p_R(T\cup\{t\})$ therefore fit within the confines of $\left[\mathbf{0}',\mathbf{1}'\right]$. The construction of the convex cover preserves the minimal and maximal values of the individual parameters, giving us the coveted $[p_R(T\cup\{t\})]\subseteq\left[\mathbf{0}',\mathbf{1}'\right]=\bigcirc\limits_{v\in\{1,\ldots,n\}}\sigma_v\circ\sigma_t\left(p_R^\sharp(T)\right)$.

We now conclude the soundness proof by showing that the observability constraint based restrictions are also reflected in the concrete parametrisation set, $[p_R(T\cup\{t\})]\subseteq\bigcirc\limits_{(u,v,\mathrm{o})\in R}\sigma_{\{(u,v,\mathrm{o})\}}\circ\bigcirc\limits_{v\in\{1,\ldots,n\}}\sigma_v\circ\sigma_t\left(p_R^\sharp(T)\right)=p_R^\sharp(T)$. Keeping to the $\left[\mathbf{0}',\mathbf{1}'\right]$ notation, we use $\left[\mathbf{0}'',\mathbf{1}''\right]=\bigcirc\limits_{(u,v,\mathrm{o})\in R}\sigma_{\{(u,v,\mathrm{o})\}}\left(\left[\mathbf{0}',\mathbf{1}'\right]\right)$ to denote the parametrisation lattice after applying observability constraint restrictions. Similarly to monotonicity, observability narrowing also imposes restriction only if the constraint is not satisfied under one, or both, of the limit parametrisations. Moreover, thanks to Lemma 5.1, observability narrowing only changes the value of at most one parameter per limit parametrisation.

Let thus $\overline{\boldsymbol{\omega}}\in\Omega_v$ be the unique regulator state whose associated parameter value gets changed in the minimum parametrisation, $\mathbf{0}''_{v,\overline{\boldsymbol{\omega}}}=\mathbf{0}'_{v,\overline{\boldsymbol{\omega}}}+1$. By definition, $\overline{\boldsymbol{\omega}}$ is the only regulator state open to value increase, $O_v^+([\mathbf{0},\mathbf{1}])=\{\overline{\boldsymbol{\omega}}\}$. $O_v^+([\mathbf{0},\mathbf{1}])$ being a singleton guarantees the equality $\mathbf{0}'_{v,\boldsymbol{\omega}'}=\mathbf{1}'_{v,\boldsymbol{\omega}'}$ for a number of regulator states $\boldsymbol{\omega}'\in\Omega_v$. Namely, any $\boldsymbol{\omega}'\parallel_v\overline{\boldsymbol{\omega}}$ or $\boldsymbol{\omega}'\succ_v\overline{\boldsymbol{\omega}}$, as well as any $\boldsymbol{\omega}'\prec_v\overline{\boldsymbol{\omega}}$ such that $\mathbf{0}'_{v,\boldsymbol{\omega}'}<\mathbf{0}'_{v,\overline{\boldsymbol{\omega}}}$.

Let now $\boldsymbol{P} \in p_R(T \cup \{t\})$ be arbitrary. We prove $\mathbf{0}''_{v,\overline{\boldsymbol{\omega}}} \leq \boldsymbol{P}_{v,\overline{\boldsymbol{\omega}}}$ and thus $\boldsymbol{P} \in \left[\mathbf{0}'', \mathbf{1}''\right]$. Unlike under $\mathbf{0}'$, all observability constraints are satisfied under $\boldsymbol{P}$, thus namely there must exist at least one $\boldsymbol{\omega}' \in \Omega_v$ such that $\boldsymbol{P}_{v,\boldsymbol{\omega}'} > \mathbf{0}'_{v,\boldsymbol{\omega}'}$. Following from the previous part of the safety proof, $\boldsymbol{P} \in p_R(T \cup \{t\})$ gives us $\boldsymbol{P}_{v,\boldsymbol{\omega}'} \leq \mathbf{1}'_{v,\boldsymbol{\omega}'}$ and thus $\boldsymbol{\omega}' \preceq_v \overline{\boldsymbol{\omega}}$ and $\mathbf{0}'_{v,\boldsymbol{\omega}'} = \mathbf{0}'_{v,\overline{\boldsymbol{\omega}}}$. Following from all monotonicity contraints being satisfied under $\boldsymbol{P}$, we get $\boldsymbol{P}_{v,\overline{\boldsymbol{\omega}}} \geq \boldsymbol{P}_{v,\boldsymbol{\omega}'} > \mathbf{0}'_{v,\boldsymbol{\omega}'} = \mathbf{0}'_{v,\overline{\boldsymbol{\omega}}}$ and therefore the coveted $\boldsymbol{P}_{v,\overline{\boldsymbol{\omega}}} \geq \mathbf{0}''_{v,\overline{\boldsymbol{\omega}}}$.

The proof for the upper limit restriction of $\underline{\boldsymbol{\omega}} \in \Omega_v$, $\mathbf{1}''_{v,\underline{\boldsymbol{\omega}}} = \mathbf{1}'_{v,\underline{\boldsymbol{\omega}}} - 1$, is analogous. $O_v^-([\mathbf{0},\mathbf{1}]) = \{\underline{\boldsymbol{\omega}}\}$ gives us $\mathbf{0}'_{v,\boldsymbol{\omega}'} = \mathbf{1}'_{v,\boldsymbol{\omega}'}$ for $\boldsymbol{\omega}' \parallel_v \underline{\boldsymbol{\omega}}$, $\boldsymbol{\omega}' \prec_v \underline{\boldsymbol{\omega}}$ and any $\boldsymbol{\omega}' \succ_v \underline{\boldsymbol{\omega}}$ such that $\mathbf{1}'_{v,\boldsymbol{\omega}'} > \mathbf{1}'_{v,\underline{\boldsymbol{\omega}}}$.

Then, for any $\boldsymbol{P} \in p_R(T \cup \{t\})$, there must exist $\boldsymbol{\omega}' \in \Omega_v$ such that $\boldsymbol{P}_{v,\boldsymbol{\omega}'} < \mathbf{1}'_{v,\boldsymbol{\omega}'}$. $\boldsymbol{P} \in p_R(T \cup \{t\})$ gives us $\boldsymbol{P}_{v,\boldsymbol{\omega}'} \geq \mathbf{0}'_{v,\boldsymbol{\omega}'}$ and thus $\boldsymbol{\omega}' \succeq_v \underline{\boldsymbol{\omega}}$ and $\mathbf{1}'_{v,\boldsymbol{\omega}'} = \mathbf{1}'_{v,\underline{\boldsymbol{\omega}}}$. Following again from monotonicity constraint satisfaction, $\boldsymbol{P}_{v,\underline{\boldsymbol{\omega}}} \leq \boldsymbol{P}_{v,\boldsymbol{\omega}'} < \mathbf{1}'_{v,\boldsymbol{\omega}'} = \mathbf{1}'_{v,\underline{\boldsymbol{\omega}}}$ and therefore the coveted $\boldsymbol{P}_{v,\underline{\boldsymbol{\omega}}} \leq \mathbf{1}''_{v,\underline{\boldsymbol{\omega}}}$.

Same as for the monotonocity case, the concrete parametrisation set therefore fits within the confines of $\left[\mathbf{0}'', \mathbf{1}''\right]$. The minimum and maximum values of individual parameters being preserved by the construction of the convex cover, we are done proving the safety of the abstraction, $[p_R(T \cup \{t\})] \subseteq \left[\mathbf{0}'', \mathbf{1}''\right] = p_R^\sharp(T \cup \{t\})$.

We now proceed with the proof of the minimality of the over-approximation $p_R^\sharp(T \cup \{t\}) \subseteq [p_R(T \cup \{t\})]$.

Adopting the generating lattice notation $\left[\mathbf{0}^\sharp, \mathbf{1}^\sharp\right] = p_R^\sharp(T \cup \{t\})$ and $\left[\mathbf{0}', \mathbf{1}'\right] = [p_R(T \cup \{t\})]$, allows us to express the sublattice relation in terms of the limit parametrisations, $\mathbf{0}^\sharp \geq_{G_{\boldsymbol{m}}} \mathbf{0}'$ and $\mathbf{1}^\sharp \leq_{G_{\boldsymbol{m}}} \mathbf{1}'$. We prove the inequalities by showing that inequalities on individual parameter values hold in the same direction.

Let us first establish a common starting point $[\mathbf{0},\mathbf{1}] = [p_R(T)] \cap p(t)$. Surely, $[p_R(T \cup \{t\})] \subseteq [p_R(T)] \cap p(t)$ and by the induction hypothesis also $p_R^\sharp(T \cup \{t\}) \subseteq \sigma_t\left(p_R^\sharp(T)\right) = [p_R(T)] \cap p(t)$. Let further $p_R^\sharp(T \cup \{t\}) \neq \varnothing$ as the sublattice relation is trivial for the empty lattice.

Let now $\boldsymbol{\omega} \in \Omega_v$ be such that $\mathbf{0}'_{v,\boldsymbol{\omega}} > \mathbf{0}_{v,\boldsymbol{\omega}}$, respectively, $\mathbf{1}'_{v,\boldsymbol{\omega}} < \mathbf{1}_{v,\boldsymbol{\omega}}$. Any such restriction on the value limits of the parameter $K_{v,\boldsymbol{\omega}}$ has to be justified by one or more influence constraints. The monotonicity influence constraints are the simple case, where $\boldsymbol{\omega} \succ_v \omega(t)$ and $\mathbf{0}'_{v,\boldsymbol{\omega}} = \mathbf{0}_{v,\omega(t)}$, respectively, $\boldsymbol{\omega} \prec_v \omega(t)$ and $\mathbf{1}'_{v,\boldsymbol{\omega}} = \mathbf{1}_{v,\omega(t)}$.

With $[\mathbf{0},\mathbf{1}]$ as the input, the minimum, respectively maximum, value of the parameter $K_{v,\boldsymbol{\omega}}$ gets restricted by σ_v, giving us the coveted: $\mathbf{0}^\sharp_{v,\boldsymbol{\omega}} \geq \mathbf{0}_{v,\omega(t)} = \mathbf{0}'_{v,\boldsymbol{\omega}}$, respectively, $\mathbf{1}^\sharp_{v,\boldsymbol{\omega}} \leq \mathbf{1}_{v,\omega(t)} = \mathbf{1}'_{v,\boldsymbol{\omega}}$.

Let us therefore assume $\boldsymbol{\omega}$ to be such that $\mathbf{0}'_{v,\boldsymbol{\omega}} > \mathbf{0}''_{v,\boldsymbol{\omega}}$, respectively $\mathbf{1}'_{v,\boldsymbol{\omega}} < \mathbf{1}''_{v,\boldsymbol{\omega}}$, where $\left[\mathbf{0}'', \mathbf{1}''\right] = \sigma_v([\mathbf{0},\mathbf{1}])$. It is important to note that $\left[\mathbf{0}'', \mathbf{1}''\right]$ cannot be limited much further as both $\mathbf{0}'', \mathbf{1}'' \in p(T \cup \{t\}) \cap \bigcap_{(u,v,s)\in R} \mathcal{P}_{(u,v,s)}$ by construction. As such any further restriction to $K_{v,\boldsymbol{\omega}}$ values results from observability. Thus by Lemma 5.1, such a $\boldsymbol{\omega}$ is unique for $\mathbf{0}$, respectively $\mathbf{1}$.

We therefore know there exists an observability influence constraint $r \in R$ which is not satisfied under $\mathbf{0}''$, respectively $\mathbf{1}''$, giving us the first pre-

requisite for the action of the observability constraint narrowing. Assuming, $[p_R(T \cup \{t\})] \neq \varnothing$, we need to prove $O_v^+([\mathbf{0},\mathbf{1}]) = \{\boldsymbol{\omega}\}$, respectively $O_v^-([\mathbf{0},\mathbf{1}]) = \{\boldsymbol{\omega}\}$.

Let us further assume the observability constraint r is not satisfied under $\mathbf{0}''$. For any parametrisation $\boldsymbol{P} \in [\mathbf{0}'', \mathbf{1}'']$ with $\boldsymbol{P}_{v,\boldsymbol{\omega}} = \mathbf{0}''_{v,\boldsymbol{\omega}}$ we know $\boldsymbol{P} \notin p_R(T \cup \{t\})$. In other words, by Lemma 5.1, at least one monotonicity constraint is not satisfied under any parametrisation $\boldsymbol{P}' \in [\mathbf{0}'', \mathbf{1}'']$ differing from $\boldsymbol{P}$ in the value of exactly one parameter other than $K_{v,\boldsymbol{\omega}}$. This can be translated to all $\preceq_v$-maximal regulator states $\boldsymbol{\omega}' \neq \boldsymbol{\omega}$ having $\mathbf{0}''_{v,\boldsymbol{\omega}'} = \mathbf{1}''_{v,\boldsymbol{\omega}'}$. Moreover, this applies separately for all levels of $\mathbf{0}''$ values. We thus get $\mathbf{0}''_{v,\boldsymbol{\omega}'} = \mathbf{1}''_{v,\boldsymbol{\omega}'}$ for each $\preceq_v$-maximal $\boldsymbol{\omega}' \neq \boldsymbol{\omega}$ in $\{\boldsymbol{\omega}'' \in \Omega_v \mid \mathbf{0}''_{v,\boldsymbol{\omega}''} = k\}$ for each $k \in \mathbb{N}$.

The $\preceq_v$-maximal $\boldsymbol{\omega}' \neq \boldsymbol{\omega}$ being value locked, $\mathbf{0}''_{v,\boldsymbol{\omega}'} = \mathbf{1}''_{v,\boldsymbol{\omega}'}$, by definition guarantees that any $\boldsymbol{\omega}'' \prec_v \boldsymbol{\omega}'$ such that $\mathbf{0}''_{v,\boldsymbol{\omega}''} = \mathbf{0}''_{v,\boldsymbol{\omega}'}$ is also value locked. This thus, in particular, holds for any $\boldsymbol{\omega}'' \succ_v \boldsymbol{\omega}$, $\boldsymbol{\omega}'' \parallel_v \boldsymbol{\omega}$ and $\boldsymbol{\omega}'' \prec_v \boldsymbol{\omega}$ such that $\mathbf{0}''_{v,\boldsymbol{\omega}''} < \mathbf{0}''_{v,\boldsymbol{\omega}}$, giving us the coveted $O_v^+([\mathbf{0},\mathbf{1}]) = \{\boldsymbol{\omega}\}$.

The proof for the observability constraint r not being satisfied under $\mathbf{1}''$ is symmetrical. For any parametrisation $\boldsymbol{P} \in [\mathbf{0}'', \mathbf{1}'']$ with $\boldsymbol{P}_{v,\boldsymbol{\omega}} = \mathbf{1}''_{v,\boldsymbol{\omega}}$ we know $\boldsymbol{P} \notin p_R(T \cup \{t\})$. Again by Lemma 5.1, all $\preceq_v$-minimal regulator states $\boldsymbol{\omega}' \neq \boldsymbol{\omega}$ must have $\mathbf{0}''_{v,\boldsymbol{\omega}'} = \mathbf{1}''_{v,\boldsymbol{\omega}'}$. This ultimately guarantees, the value is locked for any $\boldsymbol{\omega}'' \prec_v \boldsymbol{\omega}$, $\boldsymbol{\omega}'' \parallel_v \boldsymbol{\omega}$ and $\boldsymbol{\omega}'' \succ_v \boldsymbol{\omega}$ such that $\mathbf{1}''_{v,\boldsymbol{\omega}''} > \mathbf{1}''_{v,\boldsymbol{\omega}}$, giving us the coveted $O_v^-([\mathbf{0},\mathbf{1}]) = \{\boldsymbol{\omega}\}$.

Finally, let us consider $[p_R(T \cup \{t\})] = \varnothing$. Following the same line of reasoning with the use of Lemma 5.1, we know the value is locked, $\mathbf{0}''_{v,\boldsymbol{\omega}'} = \mathbf{1}''_{v,\boldsymbol{\omega}'}$, for each $\preceq_v$-maximal $\boldsymbol{\omega}' \in \{\boldsymbol{\omega}'' \in \Omega_v \mid \mathbf{0}''_{v,\boldsymbol{\omega}''} = k\}$, respectively $\preceq_v$-minimal $\boldsymbol{\omega}' \in \{\boldsymbol{\omega}'' \in \Omega_v \mid \mathbf{1}''_{v,\boldsymbol{\omega}''} = k\}$, for each $k \in \mathbb{N}$. Again, this by definition guarantees that any other regulator state is also value locked, giving us the coveted $O_v^+([\mathbf{0},\mathbf{1}]) = \emptyset$, respectively $O_v^-([\mathbf{0},\mathbf{1}]) = \emptyset$.

This concludes the proof of $p_R^\sharp(T \cup \{t\}) \subseteq [p_R(T \cup \{t\})]$. Combined with the soundness proof, we obtain the coveted $p_R^\sharp(T \cup \{t\}) = [p_R(T \cup \{t\})]$. □

5.3 Examples

In this section, we give an example of a parametric regulatory network with influence constraints, Example 5.1, to showcase how influence constraints restrict the set of possible parametrisations.

We further give several examples of how the narrowing operators for influence constraints restrict the parametrisation lattices.

Example 5.1. *Let $G' = G(F_B)$ be the minimal influence graph of the Boolean network F_B from Example 2.2. Let further $R(G')$ be a well-formed set of influence constraints containing $(b,b,-1)$, $(d,d,-1)$, $(d,a,-1)$ and positive monotonicity constraints on all other influences, as well as observability constraint on all influences except (b,a). The influence graph G' with influences annotated by their constraints from $R(G')$ is depicted in Figure 5.1.*

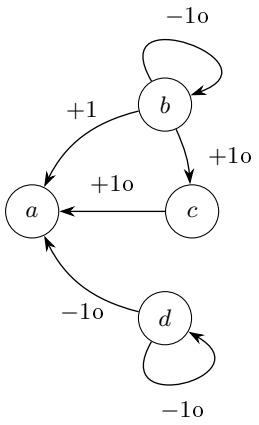


Figure 5.1: Influence graph G' of the Boolean network F_B. The influences are labelled with $+1$, -1 and o for positive monotonicity, negative monotonicity or observability constraints, respectively, that exists for the influence in $R(G')$.

Finally, let ${G'}_{\{1\}^4}$ *be a parametric regulatory network defined on the influence graph* G' *with Boolean variable domains. Unconstrained,* $2^8 \times 2^2 \times 2^2 \times 2^2 = 16384 = \left|\mathbb{P}\left({G'}_{\{1\}^4}\right)\right|$ *parametrisations are possible for the parametric regulatory network* ${G'}_{\{1\}^4}$*. By introducing* $R(G')$*, the number of compatible parametrisations decreases significantly. In some instances, such as the monotonicity minimal and maximal regulator states of a Boolean variable with all incoming influences monotonic and at least one also observable, the values of some parameters are fixed by the influence constraint set* $R(G')$*. In our case, this applies to the all four variables* a*,* b*,* c*,* d*. Indeed, for any parametrisation* $\boldsymbol{P} \in \mathcal{P}_{R(G')}$*,* $\boldsymbol{P}_{a,001} = 0$*,* $\boldsymbol{P}_{a,110} = 1$*,* $\boldsymbol{P}_{b,0} = 1$*,* $\boldsymbol{P}_{b,1} = 0$*,* $\boldsymbol{P}_{c,0} = 0$*,* $\boldsymbol{P}_{c,1} = 1$*,* $\boldsymbol{P}_{d,0} = 1$ *and* $\boldsymbol{P}_{d,1} = 0$*. Example 5.4 describes in detail how these constraints can be obtained by the means of the narrowing operator.*

Observe that since the variables b*,* c *and* d *only have one regulator,* $|R(b)| = |R(c)| = |R(d)| = 1$*, they only have two parameters each. Thus, the values of all the parameters of variables* b*,* c *and* d *are fixed by the influence constraint set* $R(G')$*. The remaining parametrisations are then all the non-constant monotonic Boolean functions on three variables which constitute the parameters of variable* a*. While enumerating all monotonic Boolean functions is generally not trivial [66], with only three variables we arrive at* 18 *different non-constant monotonic Boolean functions and thus, parametrisations* $\left|\mathcal{P}_{R(G')}\right| = 18$*.*

The different parametrisations in $\mathcal{P}_{R(G')}$ *can also be represented as cuts of the Hasse diagram [11] of the monotonicity order, or more precisely of the lattice of all regulator states of a variable with the monotonicity order, into two (in the Boolean case) or more convex sublattices. We give the Hasse diagram of the lattice* $(\Omega_a, \preceq_a)$ *in Figure 5.2. For simplicity, we use the value concatena-*

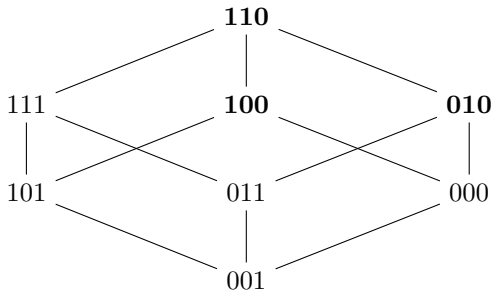


Figure 5.2: The Hasse diagram of the lattice $(\Omega_a, \preceq_a)$. The regulator states are labelled by the concatenation of the regulator values. The diagram specifies the values of the parameters of variable a in the parametrisation $\boldsymbol{P}$ by the means of non-bold regulator states having value 0 and bold regulator states having value 1.

	$K_{c,00}$	$K_{c,01}$	$K_{c,10}$	$K_{c,11}$	$K_{c,20}$	$K_{c,21}$
0	0	1	0	0	0	0
1	1	1	1	1	1	1

Table 5.1: The values of variable c parameters in the minimum and maximum parametrisations $\mathbf{0}$, $\mathbf{1}$.

tion shorthand for the regulator states in the Hasse diagram. To illustrate how a parametrisation corresponds to a cut in the Hasse diagram, we mark some of the regulator states in Figure 5.2 in bold. The cut is then the set of all edges from a bold to non-bold vertex and the corresponding parametrisation assigns the value 0 *to all non-bold regulator states and the value* 1 *to all bold regulator states. The parametrisation* $\boldsymbol{P}$ *in Figure 5.2 is thus such that the parametric Boolean network* $G'_{\{1\}^4}$ *parametrised by* $\boldsymbol{P}$ *is exactly the Boolean network* F_B *from Example 2.2.*

Of note is also the structure of the Hasse diagram in Figure 5.2. The Hasse diagram of the monotonicity order of Boolean network variable always takes the form of a hypercube, or several disjoint hypercubes in case not all incoming influences are monotonic, of dimension given by the number of monotonic regulations.

Example 5.2. *Consider the parametric regulatory network* $G_{\boldsymbol{m}}$ *introduced in Example 4.1 and a set of influence constraints* $R(G) = \{(a, c, +1)\}$. *Let* $[\mathbf{0}, \mathbf{1}]$ *be a parametrisation lattice such that the variable* c *parameter values in minimum and maximum parametrisations correspond to the Table 5.1.*

As the maximum parametrisation $\mathbf{1}$ *assign the same value* 1 *to each regulator state of variable* c, *the sole monotonicity constraint* $(a, c, +1)$ *is trivially satisfied under* $\mathbf{1}$. *On the other hand, the monotonicity constraint* $(a, c, +1)$ *is not satisfied under the minimum parametrisation* $\mathbf{0}$, *or any other paramet-*

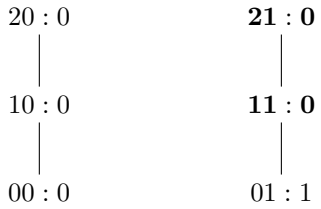


Figure 5.3: The Hasse diagram of the monotonicity order on the regulator states of variable c. The regulator states are annotated by concatenation of individual regulator values as well as their respective parameter value in the minimum parametrisation $\mathbf{0}$. The regulator states whose values are inconsistent with the influence constraint $(a, b, +1)$ are labelled in bold.

	$K_{c,00}$	$K_{c,01}$	$K_{c,10}$	$K_{c,11}$	$K_{c,20}$	$K_{c,21}$
$\mathbf{0}$	0	0	1	1	1	1
$\mathbf{1}$	1	0	1	1	1	1

Table 5.2: The values of variable c parameters in the minimum and maximum parametrisations $\mathbf{0}$, $\mathbf{1}$.

risation $\boldsymbol{P}' \in [\mathbf{0}, \mathbf{1}]$ *such that* $\boldsymbol{P}'_{c,11} = 0$ *or* $\boldsymbol{P}'_{c,21} = 0$, *due to* $(a = 0, b = 1) \prec_c (a = 1, b = 1) \prec_c (a = 2, b = 1)$.

The narrowing operator $\sigma_{(a,c,+1)}$ *assigns each parameter in the minimum parametrisation the maximum value across all monotonicity smaller regulator states. To illustrate we give the monotonicity order* $\preceq_c$ *in Figure 5.3 as the Hasse diagram of the lattice* $(\Omega_c, \preceq_c)$. *The regulator states in Figure 5.3 are annotated by their value in the minimum parametrisation* $\mathbf{0}$. *In our case, the value* $\mathbf{0}_{c,01} = 1$ *is the maximum for all three regulator states with* $b = 1$. *The two regulator states* $(a = 1, b = 1)$ *and* $(a = 2, b = 1)$, *highlighted in bold, thus have their minimum value increased by the narrowing operator. As such,* $\sigma_{(a,c,+1)}([\mathbf{0}, \mathbf{1}]) = [\mathbf{0}[(c, 11), (c, 21) \mapsto 1], \mathbf{1}]$

Example 5.3. *Similarly to Example 5.2, consider the parametric regulatory network* $G_{\boldsymbol{m}}$ *from Example 4.1 and a set of influence constraints* $R(G) = \{(b, c, o)\}$. *Let further* $[\mathbf{0}, \mathbf{1}]$ *be a parametrisation lattice such that the variable c parameter values in minimum and maximum parametrisations correspond to the Table 5.2.*

For any parametrisation $\boldsymbol{P}' \in [\mathbf{0}, \mathbf{1}]$ *such that* $\boldsymbol{P}'_{c,00} = 0$ *we have* $\boldsymbol{P}' \notin \mathcal{P}_{(b,c,o)}$ *as the value of variable b has not direct impact on the target value of variable c under any such* $\boldsymbol{P}'$. *In particular, the minimum parametrisation* $\mathbf{0}$ *is one such parametrisation. We show in detail how* $\sigma_{(b,c,o)}$ *restricts the parametrisation lattice, to eliminate the false positives in this case.*

As there are no monotonicity constraints, all regulator states are $\preceq_c$*-minimal and* $\preceq_c$*-maximal at the same time. The regulator states open for increase or decrease are thus exactly the regulator states with different value in* $\mathbf{0}$ *and* $\mathbf{1}$.

	$K_{c,00}$	$K_{c,01}$	$K_{c,10}$	$K_{c,11}$	$K_{c,20}$	$K_{c,21}$
$\mathbf{0}$	0	0	0	0	0	0
$\mathbf{1}$	1	1	1	1	1	1

Table 5.3: The values of variable c parameters in the minimum and maximum parametrisations $\mathbf{0}$, $\mathbf{1}$.

In our case $O_v^+([\mathbf{0},\mathbf{1}]) = O_v^-([\mathbf{0},\mathbf{1}]) = \{(a=0,b=0)\}$. *The existence of a single open regulator state is therefore satisfied for both* $\mathbf{0}$ *and* $\mathbf{1}$. *The observability constraint* (b,c,o) *is, however, satisfied under the maximum parametrisation* $\mathbf{1} \in \mathcal{P}_{(b,c,o)}$ *and* $\mathbf{1}$ *is not restricted. On the other hand, the minimum parametrisation* $\mathbf{0} \notin \mathcal{P}_{(b,c,o)}$ *is restricted, obtaining a new parametrisation* $\mathbf{0}' = \mathbf{0}[c, 00 \mapsto 1]$ *and* $\sigma_{(b,c,o)}([\mathbf{0},\mathbf{1}]) = [\mathbf{0}',\mathbf{1}]$.

Example 5.4. *Following from Example 5.2 and Example 5.3, consider again the parametric regulatory network* $G_{\boldsymbol{m}}$ *from Example 4.1 and a different set of influence constraints* $R(G) = \{(a,c,+1),(b,c,+1),(b,c,o)\}$. *Let* $[\mathbf{0},\mathbf{1}]$ *be again a parametrisation lattice whose variable c parameter values correspond to the Table 5.3.*

$\mathbf{0}_{c,\boldsymbol{\omega}} < \mathbf{1}_{c,\boldsymbol{\omega}}$ *for every regulator state* $\boldsymbol{\omega} \in \Omega_c$ *and every regulator state of variable c is thus a candidate for being open for increase and decrease. However, due to the monotonicity constraints* $(a,c,+1)$ *and* $(b,c,+1)$, *there exists a unique* $\preceq_c$*-minimal element* $(a=0,b=0)$ *and a unique* $\preceq_c$*-maximal element* $(a=2,b=1)$. *We thus get the following open regulator states* $O_v^-([\mathbf{0},\mathbf{1}]) = \{(a=0,b=0)\}$ *and* $O_v^+([\mathbf{0},\mathbf{1}]) = \{(a=2,b=1)\}$.

The existence of unique regulator states open for increase and decrease, respectively is thus satisfied. Moreover the observability constraint (b,c,o) *is not satisfied under either* $\mathbf{0}$ *and* $\mathbf{1}$. *Both minimum and maximum parametrisations therefore get restricted by* $\sigma_{(b,c,o)}([\mathbf{0},\mathbf{1}]) = [\mathbf{0}[c,21 \mapsto 1], \mathbf{1}[c,00 \mapsto 0]]$.

Chapter 6

Unfolding Semantics of Parametric Regulatory Networks

In this chapter we introduce partial order semantics for parametric regulatory networks. We achieve this by lifting the Petri net unfoldings introduced in Chapter 3 to parametric regulatory networks. As the dynamics of parametric regulatory networks are in essence transition systems, it follows that one can represent parametric regulatory networks by the means of Petri nets. Rather than unfolding a Petri net obtained by conversion of a parametric regulatory network, however, we introduce a modified version of the unfolding procedure which acts on the parametric regulatory networks directly in spite of the output staying a Petri net. Unfolding the parametric regulatory networks allows us to keep the model concise and thus easier to process thanks to the ability to represent the transitions symbolically.

Being a partial order semantics, the true benefit of unfoldings shows especially in highly concurrent systems. While the relative sparsity of influence graphs of most gene regulatory network models promises high degree of concurrency, this is only true for asynchronous semantics. Allowing synchronous transitions essentially introduces new interdependencies into the influence graph, prohibiting concurrent execution of transitions. To harness the full benefit of the partial order reduction, we assume all parametric regulatory network use strictly asynchronous semantics throughout the chapter.

6.1 Parametric Regulatory Network Unfolding

In this section we discuss the modifications necessary to the Petri net unfolding procedure to be applicable to parametric regulatory networks.

Before the definition of parametric regulatory network unfolding procedure itself, using Petri nets to represent the unfoldings of parametric regulatory networks requires a slight extension of the model. Namely, the unfolding is dependant on the initial marking, however, parametric regulatory networks provide no equivalent notion. We therefore annotate the parametric regulat-

ory network model with *initial condition* consisting of initial state and initial parametrisation set.[1]

Definition 6.1 (Initialised Parametric Regulatory Network). An initialised parametric regulatory network $\left(G_{\boldsymbol{m}}, \left(\mathring{\boldsymbol{x}}^{G_{\boldsymbol{m}}}, \mathring{\mathcal{P}}^{G_{\boldsymbol{m}}}\right)\right)$ of dimension n is a parametric regulatory network $G_{\boldsymbol{m}}$ of the same dimension n coupled with a state $\left(\mathring{\boldsymbol{x}}^{G_{\boldsymbol{m}}}, \mathring{\mathcal{P}}^{G_{\boldsymbol{m}}}\right) \in X_{\boldsymbol{m}} \times 2^{\mathbb{P}(G_{\boldsymbol{m}})}$.

When referring to unfolding semantics of parametric regulatory network $G_{\boldsymbol{m}}$ we automatically assume $G_{\boldsymbol{m}}$ is initialised by an initial condition $\left(\mathring{\boldsymbol{x}}^{G_{\boldsymbol{m}}}, \mathring{\mathcal{P}}^{G_{\boldsymbol{m}}}\right)$. We then say the unfolding is conducted from the initial condition $\left(\mathring{\boldsymbol{x}}^{G_{\boldsymbol{m}}}, \mathring{\mathcal{P}}^{G_{\boldsymbol{m}}}\right)$.

To simplify notation, we write $\left(\mathring{\boldsymbol{x}}, \mathring{\mathcal{P}}\right)$ instead of $\left(\mathring{\boldsymbol{x}}^{G_{\boldsymbol{m}}}, \mathring{\mathcal{P}}^{G_{\boldsymbol{m}}}\right)$ where $G_{\boldsymbol{m}}$ is obvious from context.

We proceed with the (initialised) parametric regulatory network unfolding definition by reiterating the unfolding definition for petri nets in Chapter 3.

Just like for Petri nets, the parametric regulatory network unfolding consists of the same core construction in the form of an occurrence net. Thanks to the occurrence net definition, Definition 3.5, being simplified by omitting the initial marking, no modifications are necessary for application to parametric regulatory networks. Labelling of the occurrence nets by the β function is, however, strongly dependant on the structure of the original model. We thus redefine the branching process for parametric regulatory networks using a new labelling function $\beta^\sharp$ which relates to parametric regulatory networks rather than Petri nets.

Definition 6.2 (Branching Process of a Parametric Regulatory Network). A branching process of a parametric regulatory network $G_{\boldsymbol{m}}$ of dimension n, is an occurrence net O labelled with function $\beta^\sharp \colon B \cup E \to \bigcup_{v\in\{1,\dots,n\}} (\{v\} \times \{0,\dots,\boldsymbol{m}_v\}) \cup \bigcup_{\boldsymbol{P}\in\mathring{\mathcal{P}}}$ such that:

1. $\beta^\sharp(B) \subseteq \bigcup_{v\in\{1,\dots,n\}} (\{v\} \times \{0,\dots,\boldsymbol{m}_v\})$ and $\beta^\sharp(E) \subseteq \bigcup_{\boldsymbol{P}\in\mathring{\mathcal{P}}} \xrightarrow[async]{F_{\boldsymbol{P}}}$ ($\beta^\sharp$ preserves the nature of nodes).

2. Given an arbitrary event $e \in E$ with $\beta^\sharp(e) = t = (\boldsymbol{x}, \boldsymbol{y})$. $\forall b \in {}^\bullet e$ there exists a unique $v \in \{v(t)\} \cup R(v(t))$ such that $\beta^\sharp(b) = (v, \boldsymbol{x}_v)$ and vice versa, $\forall v \in \{v(t)\} \cup R(v(t))$ there exists a unique $b = {\beta^\sharp}^{-1}((v, \boldsymbol{x}_v))$ and $b \in {}^\bullet e$. ($\beta^\sharp$ restricted to ${}^\bullet e$ is a bijection.)

[1] If unfoldings from several initial states are desired, one may benefit from including an "extra selection layer" in the unfolding. This layer would serve to allow the initial state to be selected on the run during the unfolding procedure while avoiding duplicate exploration of common behaviours. While a relatively standard practice for variable values [14, 16], if parametrisation sets are also to be picked freely, the scope may quickly explode.

Similarly, $\forall b \in e^\bullet$ there exists a unique $v \in \{v(t)\} \cup R(v(t))$ such that $\beta^\sharp(b) = (v, \boldsymbol{y}_v)$ and vice versa, $\forall v \in \{v(t)\} \cup R(v(t))$ there exists a unique $b = \beta^{\sharp -1}((v, \boldsymbol{y}_v))$ and $b \in e^\bullet$. ($\beta^\sharp$ restricted to $e^\bullet$ is a bijection.)

3. $\forall b \in \min((O))$ there exists a unique $v \in \{1, \ldots, n\}$ such that $\beta^\sharp(b) = (v, \mathring{\boldsymbol{x}}_v)$. and $\forall v \in \{1, \ldots, n\}$ there exists a unique $b = \beta^{\sharp -1}((v, \mathring{\boldsymbol{x}}_v))$ and $b \in \min((O))$. ($\beta^\sharp$ restricted to causality-minimal conditions is a bijection with the initial state.)

4. $\forall e_0, e_1 \in E,\ {}^\bullet e_0 = {}^\bullet e_1 \wedge \beta^\sharp(e_0) = \beta^\sharp(e_1) \Longrightarrow e_0 = e_1$ (No duplicate transitions).

The $\beta^\sharp$ is constructed with conditions in the unfolding of parametric regulatory networks corresponding to a combination of variable and one of its possible values, rather than the variable alone. This construction follows from the restriction to safe Petri nets, which allows at most one token in any given place. While this representation of variable values results in Definition 6.2 being considerably technical, the requirements imposed on $\beta^\sharp$ correspond exactly to the requirements on β in Definition 3.6.

The labelling function $\beta^\sharp$ allows us to use the same occurrence nets for parametric regulatory networks as for Petri nets. Thanks to the same primary building structure being preserved, no additional changes are necessary to adapt the unfolding definition to parametric regulatory networks. In the following, we reiterate Definition 3.7 and Definition 3.8 with respect to the new labelling function $\beta^\sharp$.

Definition 6.3 (Parametric Regulatory Network Branching Process Prefix). Let $(O, \beta^\sharp)$ and $\left(O', \beta^{\sharp\prime}\right)$ be two branching processes of the same parametric regulatory network $G_{\boldsymbol{m}}$.

Then, $(O, \beta^\sharp)$ is a prefix of $\left(O', \beta^{\sharp\prime}\right)$ if the following conditions are satisfied:

1. $B \subseteq B'$, $E \subseteq E'$ and $F \subseteq F'$ (O is a subnet of O');

2. $\min((O')) \subseteq B$ (The natural initial marking is the same for both branching processes);

3. For each condition $b \in B$ and the single event $e \in E'$ such that $e \in {}^\bullet b$ (if it exists), $e \in E$;

4. Similarly, for each event $e \in E$ and each condition $b \in B'$ such that $b \in {}^\bullet e \cup e^\bullet$, $b \in B$;

5. For each $x \in B \cup E$, $\beta^\sharp(x) = \beta^{\sharp\prime}(x)$ ($\beta^\sharp$ is the restriction of $\beta^{\sharp\prime}$ to O).

Definition 6.4 (Parametric Regulatory Network Unfolding). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network.

The unfolding of $G_{\boldsymbol{m}}$ is a branching process $(O, \beta^\sharp)$ of $G_{\boldsymbol{m}}$ such that any other branching process $\left(O', \beta^{\sharp\prime}\right)$ of $G_{\boldsymbol{m}}$ is a prefix of (O, β).

The unfolding of a $G_{\boldsymbol{m}}$ is unique up to isomorphism.

Unfoldings of parametric regulatory networks are in the general case infinite, as opposed to the parametric regulatory networks themselves. The infiniteness of the model results from the same patterns as we discussed for the Petri net unfoldings, namely due to the cyclic behaviour being unfolded into an acyclic structure.

While parametric regulatory network unfolding inherits the infinite structure of the Petri net unfolding, it also retains the safety conditions. Instead of reachable markings, the unfolding of parametric regulatory networks is guaranteed to contain all reachable states of the original network. Thus, for any state $\boldsymbol{x}$ reachable in the original model, there exists at least one reachable marking M of the unfolding such that $\bigcup_{v\in\{1,\dots,n\}}(v,\boldsymbol{x}_v) = \beta^\sharp(M)$. To ease notation, we write simply $\beta^\sharp(M) = \boldsymbol{x}$ by abuse of notation.

Similarly the traces of the original parametric regulatory network are captured within the unfolding. As such, given a marking M reachable in the unfolding, another marking M' and a transition $t \in \bigcup_{\boldsymbol{P}\in\mathring{\mathcal{P}}} \xrightarrow[async]{F_{\boldsymbol{P}}}$, an event $e \in E$ such that $M \xrightarrow{e} M'$, $\beta^\sharp(e) = t$ exists in the unfolding if and only if $t = \left(\beta^\sharp(M), \beta^\sharp(M')\right)$. Note that the traces as captured within the parametric regulatory network unfolding are not explicitly parametrisation sensitive. The parametrisation sets are handled on the level of configurations, as is illustrated also in Definition 6.5.

Thanks to the same underlying structure of the Petri net and parametric regulatory network unfoldings, we can readily adopt the same tools, configurations (Definition 3.9) and cuts (Definition 3.10), for expressing the behavioural equivalence between parametric regulatory network unfolding and the original network as well as between individual branches of the unfolding (up to isomorphism). While configuration and cut only depend on the occurrence net structure, we give a fresh definition for the *complete branching process of parametric regulatory networks* to properly reflect the labelling function $\beta^\sharp$ while mirroring Definition 3.11 of complete branching process of Petri nets.

Definition 6.5 (Complete Branching Process of Parametric Regulatory Networks). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network.

We say that a branching process $(O, \beta^\sharp)$ of $G_{\boldsymbol{m}}$ is complete, if for every parametrisation $\boldsymbol{P} \in \mathring{\mathcal{P}}$ and for each state $\boldsymbol{x}$ reachable in $G_{\boldsymbol{m}}$ under $\boldsymbol{P}$, there exists a configuration C in $(O, \beta^\sharp)$, satisfying:

1. $\beta^\sharp(Cut(C)) = \boldsymbol{x}$ and $\boldsymbol{P} \in p_{R(G)}\left(\beta^\sharp(C)\right) \cap \mathring{\mathcal{P}}$ ($\boldsymbol{x}$ is represented in the branching process with the parametrisation $\boldsymbol{P}$);

2. For every transition $t = (\boldsymbol{x}, \boldsymbol{y}) \in \xrightarrow[async]{G_{\boldsymbol{m}}}$ such that $\boldsymbol{P} \in p_{R(G)}(t)$, there exists an event $e \in E \setminus C$ such that $\beta^\sharp(e) = t$ and $C \cup \{e\}$ is a configuration of (O, β) (all transitions enabled under $\boldsymbol{P}$ can be reproduced in the branching process).

To ease notation, we write $p_R(C)$ instead of $p_{R(G)}\left(\beta^\sharp(C)\right) \cap \mathring{\mathcal{P}}$ to denote the set of parametrisations allowing a configuration C of the unfolding.

A branching process of parametric regulatory network is complete under the same criteria as the branching process of a Petri net. All reachable states and enabled transitions have to be represented. In the case of parametric regulatory networks, however, we additionally require the states to be represented for any parametrisation witnessing the reachability in the original network. Similarly, the transitions have to be represented for any parametrisation that enables them. These conditions ensure that no parametrisations are lost in the complete branching processes. Thus, namely, complete branching process fully encompasses the behaviour of the parametric regulatory network parametrised by any $\boldsymbol{P} \in \mathring{\mathcal{P}}$.

6.2 Complete Finite Prefix of Parametric Unfolding

Parametric regulatory networks and their state transition graphs, which represent the behaviour of the networks, are finite by definition unlike the generally infinite unfoldings. Addressing the discrepancy, this section presents elevation of the construction of complete finite prefixes of Petri net unfoldings to parametric regulatory network unfoldings.

Similarly to the unfolding itself, complete finite prefixes of parametric regulatory network unfoldings are based on the same principles as complete finite prefixes of Petri nets. In particular, we capitalise on the uniqueness of the unfoldings up to isomorphism and the 'self inclusion' of unfoldings from different initial condition. More precisely, given an unfolding (B, E, F, M_0) of a parametric regulatory network $G_{\boldsymbol{m}}$ with a configuration $C \subseteq E$, a branching process $\left(O', \beta^{\sharp\prime}\right)$ such that $B' \cup E' = \{\, x \in (B \cup E) \setminus ({}^\bullet C \cup C) \mid \forall\, e \in C,\ \neg\,(e \,\#\, x) \,\}$ is the unfolding of $G_{\boldsymbol{m}}$ with initial condition $\left(\beta^\sharp(\mathit{Cut}(C)), p_{R(G)}(C)\right)$. Then, thanks to unfoldings being unique up to isomorphism, we know the branching process constructed in the same fashion for any other configuration $C' \subseteq E$ such that $\beta^\sharp(\mathit{Cut}(C')) = \beta^\sharp(\mathit{Cut}(C))$ and $p_{R(G)}(C') = p_{R(G)}(C)$ is isomorphic to $\left(O', \beta^{\sharp\prime}\right)$.

While the above highlighted isomorphism of unfolding branches is fundamentally sufficient, we can benefit from the structure of initialised parametric regulatory networks to obtain a more general inclusion relation for branches of parametric regulatory network unfoldings. In particular, the initial state $\left(\mathring{\boldsymbol{x}}, \mathring{\mathcal{P}}\right)$ corresponds to the initial marking of Petri nets, the initialised parametric regulatory networks, however, also specify the initial parametrisation set $\mathring{\mathcal{P}}$. The parametric regulatory network unfolding are essentially a union over unfoldings of the original network parametrised by any parametrisation $\boldsymbol{P} \in \mathring{\mathcal{P}}$. It follows that the unfolding from initial condition $(\boldsymbol{x}, \mathcal{P})$ should be fully included (up to isomorphism) in the unfolding from initial condition $(\boldsymbol{x}, \mathcal{P}')$ where $\mathcal{P} \subseteq \mathcal{P}'$.

Lemma 6.1 (Inclusion of Parametric Regulatory Network Unfoldings)**.** *Let $G_{\boldsymbol{m}}$ be a parametric regulatory network and let (B, E, F, M_0) and (B', E', F', M_0') be unfoldings of $G_{\boldsymbol{m}}$ from initial configurations $(\boldsymbol{x}, \mathcal{P})$ and $(\boldsymbol{x}, \mathcal{P}')$ respectively, such that $\mathcal{P} \subseteq \mathcal{P}'$. Let further $\boldsymbol{P} \in \mathcal{P}$ be an arbitrary parametrisation and $C \subseteq E$ a configuration such that $\boldsymbol{P} \in p_R(C)$.*

Then there exists a configuration $C' \subseteq E'$ satisfying $\beta^{\sharp'}(C') = \beta^{\sharp}(C)$, $\beta^{\sharp'}(Cut(C')) = \beta^{\sharp}(Cut(C))$ and $\boldsymbol{P} \in p_R(C')$.

Proof. We conduct the proof as mathematical induction on the size of the configurations of the unfolding (B, E, F, M_0).

For the base step, $C = \emptyset = C'$. $Cut(C) = M_0$ and $Cut(C') = M_0'$. By definition $\beta^{\sharp}(M_0) = \boldsymbol{x} = \beta^{\sharp'}(M_0')$. $\beta^{\sharp}(C) = \emptyset = \beta^{\sharp'}(C')$ and $\boldsymbol{P} \in p_R(C') = \mathcal{P}'$ follow trivially.

Let now $C \in E$ and $C' \in E'$ be such that $\beta^{\sharp}(C) = \beta^{\sharp'}(C')$, $\beta^{\sharp}(Cut(C)) = \beta^{\sharp'}(Cut(C'))$ and $\boldsymbol{P} \in p_R(C) \cap p_R(C')$. Let further $e \in E$ be arbitrary such that $C \cup \{e\}$ is a configuration and $\boldsymbol{P} \in p_R\left(\beta^{\sharp}(e)\right)$.

Consider now an event e' such that $\beta^{\sharp'}(e') = \beta^{\sharp}(e)$ and ${}^\bullet e' \subseteq Cut(C')$. Such an event surely exists thanks to $\beta^{\sharp}(Cut(C)) = \beta^{\sharp'}(Cut(C'))$. Furthermore $\boldsymbol{P} \in p_R\left(\beta^{\sharp}(e')\right) = p_R\left(\beta^{\sharp}(e)\right)$. Since the unfolding is the largest possible branching process, we know $e' \in E'$ (unless an isomorphic event exists in E', in which case we consider the isomorphic event to be e'). Finally, $C' \cup \{e'\}$ is a configuration of (B', E', F', M_0') by definition. □

As is the case for Petri net unfoldings, the complete finite prefix of parametric regulatory network unfoldings is constructed using cut-off events. To determine which events are safe to be marked as cut-off events, we once again use local configurations Definition 3.12 to associate each event in the unfolding with a unique configuration. While the definitions of local configuration and possible extension (Definition 3.13) carry over from Petri net unfoldings, we introduce a new *parametric cut-off event* for unfoldings of parametric regulatory networks to be able to utilise the asymmetric inclusion from Lemma 6.1.

Definition 6.6 (Parametric Cut-Off Event). Let $\left(O, \beta^{\sharp}\right)$ be a finite branching process of a parametric regulatory network $G_{\boldsymbol{m}}$ and let (O', β') be another branching process of the same parametric regulatory network such that $B' = B \cup e^\bullet$ and $E' = E \cup \{e\}$, where $e \in PE\left(\left(O, \beta^{\sharp}\right)\right)$ is a possible extension of the branching process (O, β).

Then the event e is a cut-off in (O', β'), $e \in cutoffs\left((O', \beta')\right)$, if and only if $p_R([e]) = \emptyset$ or there exists an event $e' \in E$ such that $\beta^{\sharp}(Cut([e])) = \beta^{\sharp}(Cut([e']))$ and $p_R([e]) \subseteq p_R([e'])$.

The set of parametric cut-off events $cutoffs\left(\left(O, \beta^{\sharp}\right)\right)$ is similarly to the cut-off event set of Petri net unfoldings, Definition 3.14, dependant on the order in which the possible extensions are explored. To guarantee the obtained finite prefix is complete, the use of adequate order as per Definition 3.15 is once again necessary. Additionally, we utilise the cut-off events for the purpose of pruning

the branches that are not allowed by any parametrisation in $\mathring{\mathcal{P}}$. This allows us to keep the definition of the branching process itself parametrisation free, and only annotate configurations with parametrisation sets.

Due to the added dependency on the parametrisation sets, however, use of total adequate order does not guarantee the obtained unfolding to be smaller than the state transition graph. Instead, parametric cut-off events being dependant on the parametrisation set inclusion introduces another optimality criterion. In particular, it is desirable to explore the possible extensions in the reverse subset order of the parametrisation sets. Unfortunately, the subset order is orthogonal with the total adequate order of Esparza et al. [31]. As such, using the Esparza et al. [31] total adequate order, it is possible for an event e to be explored after e' while $\beta^\sharp\left(Cut\left([e]\right)\right) = \beta^\sharp\left(Cut\left([e']\right)\right)$ and $p_{R(G)}\left(e'\right) \subseteq p_{R(G)}\left(e\right)$. Such cases lead to what we refer to as 'backwards cut-off' or 'backwards merge' resulting in the chronologically older event e' being declared cut-off and any further events $e'' > e'$ being discarded or re-evaluated. Nevertheless, the obtained complete finite prefixes of parametric regulatory network unfoldings while using the Esparza et al. [31] total adequate order tend to be considerably small in comparison to other methods, as highlighted by results presented in the Chapter 9.

Finally, throughout this chapter, we use the concrete parametrisation set notation p with the influence constraints $R\left(G\right)$. However, it should be noted that any parametrisation set computation $p\colon 2^{\bigcup_{\boldsymbol{P}\in\mathring{\mathcal{P}}} \xrightarrow[async]{F_{\boldsymbol{P}}}} \to 2^{\mathbb{P}(G_{\boldsymbol{m}})}$ is applicable for the complete finite prefix construction, as long as, the following implication is preserved: $T \subseteq T' \Longrightarrow p\left(T'\right) \subseteq p\left(T\right)$. Thus, in particular, the abstract parametrisation sets used in the abstract semantics of parametric regulatory networks, Definition 5.11, are compatible. It is therefore possible to construct unfoldings and complete finite prefixes of parametric regulatory networks using the abstract parametrisation sets. Such a combination results in heuristic reduction of both the exponential size parametrisation space, thanks to the abstract parametrisation sets, and the exponential size state space, thanks to the partial order semantics.

6.3 Examples

In this section we give two examples of a unfolding prefix for the parametric regulatory network from Example 4.1. The first prefix, in Example 6.1, is an incomplete prefix showcasing detection of a cut-off event. The second prefix, in Example 6.2, is then a complete finite prefix of the unfolding.

Example 6.1. *In this example we show the detection of a cut-off event in a prefix of the unfolding of the parametric regulatory network $G_{\boldsymbol{m}}$ from Example 4.1 with the monotonicity constraints from influence constraints set $R\left(G\right) = \{(a,a,-1),(b,b,-1),(a,c,+1),(b,c,+1)\}$. The prefix shown in Figure 6.1 captures the status of the unfolding procedure after processing the fifth event e_5.*

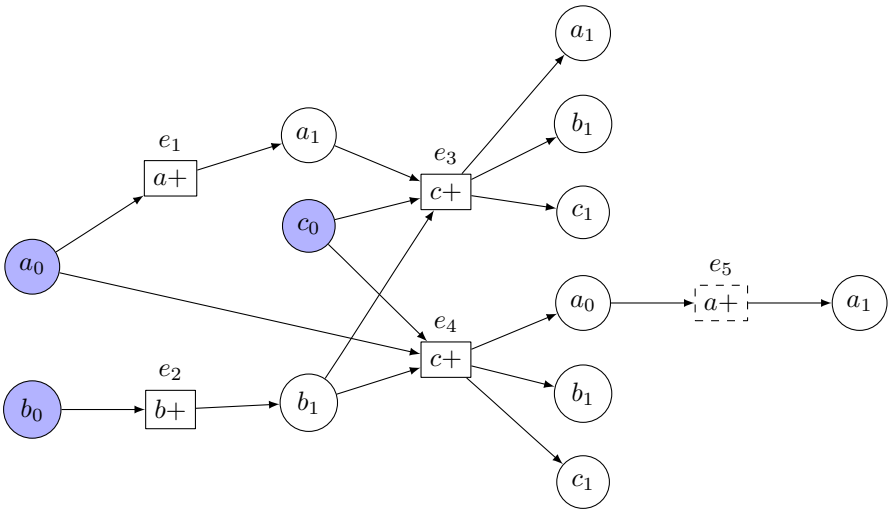


Figure 6.1: A prefix of the unfolding of the parametric regulatory network $G_{\boldsymbol{m}}$ with the influence constraint set $R(G)$. The usual Petri net notation of spherical and rectangular nodes is used to distinguish conditions and events, respectively. The conditions are labelled by the variable and its value given by the labelling function $\beta^\sharp$. Similarly, the events are labelled by the unique variable changing value and the nature of the value change for transition given by the labelling function $\beta^\sharp$. The events are additionally numbered in the order of exploration by the unfolding process. The conditions belonging to the initial marking are highlighted in blue. The dashed event is declared as cut-off during the complete finite prefix construction.

Before we discuss the cut-off event e_5, observe that events e_1 and e_2 are concurrent, irrespective of the firing order they lead to the state $(a = 1, b = 1, c = 0)$ via the labelling function $\beta^\sharp$. In a classical state space graph computation, transitions $\beta^\sharp(e_1)$ and $\beta^\sharp(e_2)$ would appear twice. $\beta^\sharp(e_1)$ followed by $\beta^\sharp(e_2)$ in the intermediate state $(a = 1, b = 0, c = 0)$ and vice versa, $\beta^\sharp(e_2)$ followed by $\beta^\sharp(e_1)$ in the intermediate state $(a = 0, b = 1, c = 0)$. The unfolding allows the state $(a = 1, b = 1, c = 0)$ to be used by including both posets of e_1 and e_2 in the preset, as is done by e_3, while also maintaining both intermediate states $(a = 1, b = 0, c = 0)$ and $(a = 0, b = 1, c = 0)$ (used by e_4).

Let us now focus on the events e_3 and e_5. As $\beta^\sharp(Cut([e_3])) = \beta^\sharp(Cut([e_5]))$, e_5 would be declared cut-off in a Petri net unfolding. To be declared cut-off in parametric regulatory network unfolding, however, $p^\sharp_{R(G)}([e_5]) \subseteq p^\sharp_{R(G)}([e_3])$ must also hold in order to capture all possible behaviours.

In the case of e_3, $[e_3] = \{e_1, e_2, e_3\}$. The value of the minimum parametrisation in $p^\sharp_{R(G)}([e_3]) = [\mathbf{0}, \mathbf{1}]$ thus differ from the minimal possible values in

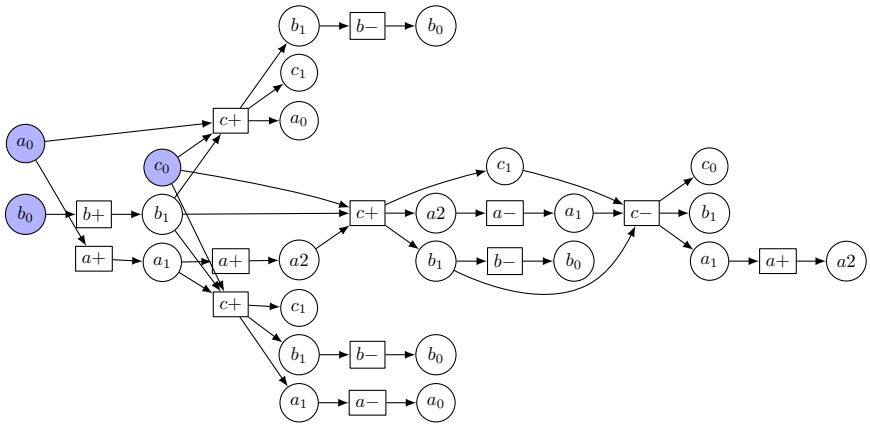


Figure 6.2: The complete finite prefix of the unfolding of the parametric regulatory network $G_{\boldsymbol{m}}$ with the influence constraint set $R(G)$. The cut-off events are not represented.

the following:

$$\mathbf{0}_{a,0} = 1 \text{ due to } e_1$$
$$\mathbf{0}_{b,0} = 1 \text{ due to } e_2$$
$$\mathbf{0}_{c,11} = 1 \text{ due to } e_3$$
$$\mathbf{0}_{c,21} = 1 \text{ by } \sigma_{(a,c,+1)}$$

For e_5, $[e_5] = \{e_2, e_4, e_5\}$. *Following constraints are thus placed on the minimum parametrisation in* $p^\sharp_{R(G)}([e_5]) = [\mathbf{0}', \mathbf{1}]$*:*

$$\mathbf{0}'_{b,0} = 1 \text{ due to } e_2$$
$$\mathbf{0}'_{c,01} = 1 \text{ due to } e_4$$
$$\mathbf{0}'_{a,0} = 1 \text{ due to } e_5$$
$$\mathbf{0}'_{c,11} = \mathbf{0}_{c,21} = 1 \text{ by } \sigma_{(a,c,+1)}$$

The parametrisation $\mathbf{0}'$ *has one extra restriction on the parameter* $K_{c,01}$ *compared to* $\mathbf{0}$*, giving us* $\mathbf{0}' >_{G_{\boldsymbol{m}}} \mathbf{0}$*. Thus,* $p^\sharp_{R(G)}([e_5]) \subseteq p^\sharp_{R(G)}([e_3])$ *and* e_5 *is indeed a cut-off.*

Example 6.2. *In this example we illustrate the complete finite prefix constructed for the unfolding of the parametric regulatory network* $G_{\boldsymbol{m}}$ *from Example 4.1. The parametric regulatory network* F *is considered with influence constraint set* $R(G) = \{(a,a,-1),(b,b,-1),(a,c,+1),(b,c,+1),(a,a,o),(b,b,o),(a,c,o),(b,c,o)\}$*. The resulting complete finite prefix is given in Figure 6.2.*

The conditions and events are labelled by the $\beta^\sharp$ *in the same fashion as in Figure 6.1. The conditions representing the initial state* $(a = 0, b = 0, c = 0)$

are also coloured blue. The cut-off events are omitted to improve readability of the figure.

Chapter 7

Goal-Driven Unfolding

Parametric regulatory networks encompass the behaviours of the parametrised networks for any admissible parametrisation. As such, the sheer number of different behavioural patterns parametric regulatory networks exhibit leads to very heavy branching in the parametric regulatory network unfoldings. Rather than the entire reachable state space, however, the interest of many reachability questions is whether a particular reachability property can be satisfied. By focusing only on a single reachability property, it may become possible to prune some branches of the unfolding early, knowing they may never lead to the desired outcome. For this reason we extend the unfolding semantics of parametric regulatory networks with a goal-driven application.

In this chapter, we adapt the goal-driven unfolding method for Petri nets [16] to the unfoldings of parametric regulatory networks. The method is based on static analysis method which reduces the input model by pruning transitions that never lead to the given target property. We briefly recall the model reduction method for automata networks as introduced in [62] and adapt it to parametric regulatory networks in Section 7.1. Parametric regulatory networks need to be processed for the reduction method to be applicable. To achieve this, the information on transitions enabled by the parametrisation sets is compiled into suitable structures (Definition 7.9). Section 7.2 thus presents an example algorithm for conducting the necessary conversion.

The application within the unfolding itself mirrors the application to Petri net unfoldings [16]. In principle, the model reduction method is executed after inclusion of an event e during the unfolding procedure. The possible extensions e' such that $e < e'$ are then considered based on the reduced model. Thus, by adapting the model reduction method of [62] to the parametric regulatory networks whose semantics rely on refinement of parametrisation sets, we obtain a combination of orthogonal on-the-run reduction and refinement of the model.

7.1 Goal-Driven Reduction

In this section we introduce the automata network procedure of [62] adapted to parametric regulatory networks.[1] The original automata network reduction procedure of [62] is based on causality analysis of transitions of individual automata within the network. The method identifies and prunes transitions which are guaranteed not to lead to the target property while preserving all minimal (in terms of included loops) traces which reach the goal. As such, if the automata network allows several different ways of reaching the goal, the reduction preserves all of them.

As the concept of trace is heavily exploited in the defining principles of the reduction method, we first formalise the trace of parametric regulatory networks. A trace is generally understood as a sequence of compatible transitions. Instead of using the global transitions as understood in the semantics, we define the trace using local transitions, or more literally *value updates*. The local definition not only mirrors the use of local causality in the original reduction procedure for automata networks [62], but corresponds to the usual notion of Mazurkiewicz traces used in both trace and concurrency theory [56].

Definition 7.1 (Variable Value Update). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network of dimension n and let $v \in \{1, \ldots, n\}$ be an arbitrary variable of $G_{\boldsymbol{m}}$.

Then, a value update $\mu = v_x \to v_y$ is couple of values $(x, y) \in \{0, \ldots, \boldsymbol{m}_v\}^2$ of variable v such that $|x - y| = 1$.

To ease notation, we write simply $x \to y$ instead of $v_x \to v_y$ when the variable v is obvious from context.

Given a state $\boldsymbol{x} \in X_{\boldsymbol{m}}$ such that $\boldsymbol{x}_v = x$, we use $\mu(\boldsymbol{x}) = \boldsymbol{x}[v \mapsto y]$ to denote the state reached by executing the value update μ in state $\boldsymbol{x}$. The notation naturally extends to sequences of value changes:[2]

$$
\begin{aligned}
()(\boldsymbol{x}) &= \boldsymbol{x} \\
\mu \cdot \pi(\boldsymbol{x}) &= \pi(\mu(\boldsymbol{x}))
\end{aligned}
$$

Furthermore, given a parametrisation $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$, we say that the value update μ is enabled in $\boldsymbol{x}$ under $\boldsymbol{P}$ if $(\boldsymbol{x}, \boldsymbol{x}[v \mapsto y]) \in \xrightarrow[async]{F_{\boldsymbol{P}}}$.

Finally, given a transition $t = (\boldsymbol{x}, \boldsymbol{y}) \in \xrightarrow[async]{F_{\boldsymbol{P}}}$ for some $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$, we say $\mu(t) = \boldsymbol{x}_v \to \boldsymbol{y}_v$ where $v = v(t)$ is the value update of the transition t.

The motivation being application in conjunction with unfolding, we again limit ourselves to traces of asynchronous semantics, either concrete of abstract,

[1] Note that while the original model reduction method is defined for automata networks, as mentioned in Section 2.4, automata networks and discrete regulatory networks are expressivity equivalent. As such, the adaptation is essentially lifting from the parametrised to the parametric regulatory networks.

[2] Note that this operation corresponds to the monoidal category action [53] of the trace monoid on states of the network.

of the parametric regulatory networks. As such, we use value updates directly in the trace definition. An extension to generalised asynchronous semantics can be achieved by considering *steps* composed of value updates of several variables akin to the construction in [62].

Definition 7.2 (Parametric Regulatory Network Trace). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network of dimension n and let $\boldsymbol{x} \in X_{\boldsymbol{m}}$ be an arbitrary state of $G_{\boldsymbol{m}}$.

Then a sequence of value updates π is a (local or Mazurkiewicz) trace of $G_{\boldsymbol{m}}$ starting in $\boldsymbol{x}$ if there exists a parametrisation $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$ such that for every $i \in \{2, \ldots, |\pi|\}$, $((\pi_1, \ldots, \pi_{i-1})(\boldsymbol{x}), (\pi_1, \ldots, \pi_i)(\boldsymbol{x})) \in \xrightarrow[async]{F_{\boldsymbol{P}}}$.

To ease notation, we use $\pi^{:i} = (\pi_1, \ldots, \pi_i)$, $\pi^{i:} = (\pi_i, \ldots)$ and $\pi^{i:j} = (\pi_i, \ldots, \pi_j)$ to denote prefix, suffix and infix sub-traces of trace π respectively, and $\pi \cdot \pi'$ to denote concatenation of two traces.

Finally, we write $p(\pi, \boldsymbol{x}) = \left\{ \boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}}) \;\middle|\; \forall i \in \{2, \ldots, |\pi|\}, \; (\pi^{:i-1}(\boldsymbol{x}), \pi^{:i}(\boldsymbol{x})) \in \xrightarrow[async]{F_{\boldsymbol{P}}} \right\}$ to denote the set of all parametrisations that enable the trace π from initial state $\boldsymbol{x}$.

We introduce the reduction method for goal properties specified as a target value $\top$ for a particular variable g of the parametric regulatory network.

Definition 7.3 (Goal Reachability). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network of dimension n, $\boldsymbol{x} \in X_{\boldsymbol{m}}$ a state of $G_{\boldsymbol{m}}$, $g \in \{1, \ldots, n\}$ a variable of $G_{\boldsymbol{m}}$ and $\top \in \{0, \ldots, \boldsymbol{m}_g\}$ a value of the variable g.

We say that value $\top$ of variable g is reachable in $G_{\boldsymbol{m}}$ from state $\boldsymbol{x}$ if, either $\boldsymbol{x}_g = \top$, or there exists a trace π from the initial state $\boldsymbol{x}$ such that $\pi(\boldsymbol{x})_g = \top$.

To simplify to notation we refer to $g_\top$ as goal.

While the goal is defined on a single variable, the extension to values for several variables or even value sets as opposed to single value per variable is straightforward.

A goal $g_\top$ is commonly reachable by traces with infix loops which, by themselves, do not affect the reachability of the goal. To maximise the efficiency of the reduction procedure, only *minimal* traces, devoid of such loops, are considered. Adapted from [62], a trace is minimal for reachability of a goal $g_\top$ if there exists no other realisable trace reaching $g_\top$ with a subsequence of value updates through the same regulator states.

Definition 7.4 (Minimal Trace). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network, $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$ an arbitrary parametrisation, $\boldsymbol{x} \in X_{\boldsymbol{m}}$ a state of $G_{\boldsymbol{m}}$ and $g_\top$ a goal.

Then a trace π from state $\boldsymbol{x}$ of the parametrised network $F_{\boldsymbol{P}}$ such that $\pi(\boldsymbol{x})_g = \top$ is minimal for the reachability of $g_\top$ from $\boldsymbol{x}$ if and only if there exists no other trace ρ from state $\boldsymbol{x}$ of $F_{\boldsymbol{P}}$ satisfying all of the following:

1. $\rho(\boldsymbol{x})_g = \top$;

2. $|\rho| < |\pi|$;

3. there exists a monotonic injection $\Phi\colon \{0,\dots,|\pi|\} \to \{0,\dots,|\rho|\}$ such that $\Phi\colon 0 \mapsto 0$ and $\forall i,j \in \{0,\dots,|\pi|\},\ i \leq j \implies \Phi(i) \leq \Phi(j) \wedge \pi_i = v_x \to v_y = \rho_{\Phi(i)} \wedge \omega_v\left(\pi^{:i}(\boldsymbol{x})\right) = \omega_v\left(\rho^{:\Phi(i)}(\boldsymbol{x})\right)$.

Unlike transitions of automata networks in [62], value updates themselves do not contain any information on the actual traversed states of the network. An extra condition on regulator state equality, $\omega_v\left(\pi^{:i}(\boldsymbol{x})\right) = \omega_v\left(\rho^{:\Phi(i)}(\boldsymbol{x})\right)$, is therefore necessary to retain the dynamic information allowing us to distinguish different traces which are subsequences of value updates (See Example 7.1). Another important property of minimal traces safeguarded by the regulator state equality is their independence on the exact parametrisation. More precisely, if a trace is minimal for at least one parametrisation, then it is minimal for any other parametrisation under which it is enabled.

Proposition 7.1. *Let $G_{\boldsymbol{m}}$ be a parametric regulatory network of dimension n and π a trace from $\boldsymbol{x} \in X_{\boldsymbol{m}}$ minimal for reachability of goal $g_\top$ in $F_{\boldsymbol{P}}$ for some $\boldsymbol{P} \in \mathbb{P}(G_{\boldsymbol{m}})$. Then, π is minimal in the network parametrised by any other parametrisation $F_{\boldsymbol{P}'}$ where $\boldsymbol{P}' \in p_{R(G)}(\pi)$.*

Proof. $\boldsymbol{P}' \in p_{R(G)}(\pi)$ guarantees π is a proper trace of $F_{\boldsymbol{P}'}$. We conduct the rest of the proof by contradiction. Let thus ρ be a trace of $F_{\boldsymbol{P}'}$ satisfying the conditions in Definition 7.4. From the existence of the injection Φ we get $\widetilde{\rho} \subseteq \widetilde{\pi}$. By definition of parametrisation sets $T \subseteq T' \implies p_{R(G)}(T') \subseteq p_{R(G)}(T)$. Thus, thanks to $\omega_v\left(\pi^{:i}(\boldsymbol{x})\right) = \omega_v\left(\rho^{:\Phi(i)}(\boldsymbol{x})\right)$ for all $i \in \{0,\dots,|\pi|\}$, $p_{R(G)}(\widetilde{\pi}) \subseteq p_{R(G)}(\widetilde{\rho})$. ρ is therefore a trace of $F_{\boldsymbol{P}}$ contradicting the minimality of π. □

Thanks to Proposition 7.1, it is sufficient to say that a trace of a parametric regulatory network is minimal without the need to explicitly list the parametrisation bearing witness to the minimality.

The goal-driven reduction of automata networks is facilitated by pruning transitions which are not part of any minimal trace reaching the goal [62]. The individual transitions are, however, not explicitly represented in parametric regulatory networks. While it is not a challenge to represent the removed transitions explicitly, the reduction method is proposed for the general automata networks, which allow arbitrary transitions within the automata. On the other hand, we have limited ourselves to multivalued networks that are only allowed to change value of a variable by steps of size 1. As such, if a transition increasing the value of a variable v to $x \in \{0,\dots,\boldsymbol{m}_v\}$ is to be pruned, all transitions increasing the value of v to a value beyond, to $y \geq x$, can surely be pruned as well. A symmetrical reasoning applies to decreasing transitions.

Thus, instead of pruning individual transitions of parametric regulatory networks and representing the removed/preserved transitions explicitly, we adapt the method to disable increasing, respectively decreasing, value of a variable in a given regulator state beyond a certain value (or entirely). This is achieved

in a similar pattern to the parametrisation lattices. We extend the parametric regulatory networks with a record of the activation (increase) and inhibition (decrease) limits for each variable in vectors $\boldsymbol{l}^A$ and $\boldsymbol{l}^I$ respectively.

Definition 7.5 (Directed Parametric Regulatory Network). A directed parametric regulatory network $\mathcal{G} = \left(G_{\boldsymbol{m}}, \boldsymbol{l}^A, \boldsymbol{l}^I\right)$ is a parametric regulatory network $G_{\boldsymbol{m}}$ coupled with a vector $\boldsymbol{l}^A \in (\mathbb{N} \cup \{-\infty\})^{|\Omega|}$ of activation limits for each regulator state $\boldsymbol{\omega} \in \Omega$ and a vector $\boldsymbol{l}^I \in (\mathbb{N} \cup \{0, \infty\})^{|\Omega|}$ of inhibition limits for each regulator state $\boldsymbol{\omega} \in \Omega$.

The semantics of the directed parametric regulatory network $\mathcal{G}$ are the semantics of the parametric regulatory network $G_{\boldsymbol{m}}$ restricted to the activation and inhibition limits, $\boldsymbol{l}^A$ and $\boldsymbol{l}^I$ respectively. For any $t = (\boldsymbol{x}, \mathcal{P}) \xrightarrow[async]{G_{\boldsymbol{m}}} (\boldsymbol{y}, \mathcal{P}')$ of the parametric regulatory network $G_{\boldsymbol{m}}$:

$$t \in \xrightarrow[async]{\mathcal{G}} \overset{\Delta}{\Longleftrightarrow} \begin{array}{c} sign\,(t) = +1 \Longrightarrow \boldsymbol{x}_{v(t)} < {\boldsymbol{l}^A}_{\omega_{v(t)}(\boldsymbol{x})} \\ \wedge \\ sign\,(t) = -1 \Longrightarrow \boldsymbol{x}_{v(t)} > {\boldsymbol{l}^I}_{\omega_{v(t)}(\boldsymbol{x})} \end{array}$$

One may remark that the aforementioned parametrisation lattices used in abstract parametric regulatory networks already allow restriction of the activation or inhibition of variables in individual regulator states. Indeed, $\left[\boldsymbol{l}^I, \boldsymbol{l}^A\right]$ forms a parametrisation lattice itself as long as it contains no infinity values. The role of the parametrisation lattices as employed in the abstract parametric network semantics and of the limit vectors $\boldsymbol{l}^A$ and $\boldsymbol{l}^I$, however, differs on a fundamental level.

The parametrisation lattices serve to keep track of parametrisations capable of reproducing certain behaviour(s), and thus restrict the set of enabled transitions based on their causal history. On the other hand, the $\boldsymbol{l}^A$ and $\boldsymbol{l}^I$ of directed parametric regulatory networks mark components whose activation or inhibition (beyond a certain value) is not necessary to reach a given goal by the means of a minimal trace. A parametrisation that allows changing a component value beyond the limit, thus allowing behaviour which does not lead to the established goal may still allow a different sequence of transitions leading to the goal. We want to retain such parametrisations, thus the 'useless' behaviour which does not lead to the goal cannot be restricted by the means of the parametrisation lattice of the abstract parametric regulatory network semantics. While the independence from parametrisation lattices in abstract parametric regulatory network semantics requires us to keep track of the extra limit vectors $\boldsymbol{l}^A$ and $\boldsymbol{l}^I$, it also guarantees that the goal-driven unfolding is applicable alongside both the concrete and the abstract semantics of parametric regulatory networks.

With the minimal traces and the directed parametric regulatory networks we have the necessary groundwork to introduce the reduction procedure itself. As we have already mentioned, the reduction relies on causality analysis. In

particular, the reduction procedure identifies sub-goals based on local causality for individual components, called (local) *objectives*. The objectives represent a change of variable value that necessarily has to occur before a certain transition can achieve later objective or the goal itself. The objectives as defined for the automata networks [62] keep the desired value evolution abstract and thus do not specify how, i.e. by means of which transitions, the value change occurs. While we do not explicitly include this information, the restriction we have imposed on the multivalued networks allowing the variables to only change value by steps of size 1, distinctly predetermines how the desired value change may be achieved. Temporary value swings may still be necessary, however, to facilitate regulation of a necessary variable.

Definition 7.6 (Objective). Let $\mathcal{G} = \left(G_{\boldsymbol{m}}, \boldsymbol{l}^A, \boldsymbol{l}^I\right)$ be a directed parametric regulatory network of dimension n.

Then an objective $O = v_x \rightsquigarrow v_y$ is a pair of values $x, y \in \{0, \ldots, \boldsymbol{m}_v\}$ of a variable $v \in \{1, \ldots, n\}$.

We say an objective O is *valid* in an initial state $\boldsymbol{x} \in X_{\boldsymbol{m}}$ if $x = y$ or $\mathcal{G}$ has a trace π from $\boldsymbol{x}$ such that $\pi(\boldsymbol{x})_v = y$ and there exists an index $i \in \{0, \ldots, |\pi|\}$ such that $\pi^{:i}(\boldsymbol{x})_v = x$.

We use $x \rightsquigarrow y$ to denote $v_x \rightsquigarrow v_y$ where the variable $v \in \{1, \ldots, n\}$ is obvious from the context.

Each objective $v_x \rightsquigarrow v_y$ represents either increase or decrease of the value of the component. We use $sign(v_x \rightsquigarrow v_y) = sign(y - x)$ to denote the direction of the prescribed value evolution.

We demand the traces of the parametric regulatory networks to be realisable within at least one of the associated parametrised networks. The condition on existence of a trace thus also guarantees that there exists a parametrisation that enables said trace. A valid objective is therefore surely fully realisable under at least one parametrisation.

The objective represents a change of value of only one variable $v \in \{1, \ldots, n\}$. The trace bearing witness to the validity of such an objective may, however, require other variables to also change value, namely the regulators of v. The automata network reduction procedure associates each objective with a set of transitions which may be used to fulfil the objective [62]. Such a transition set may then be used to obtain the objectives for regulators of v. In the parametric regulatory networks, however, transitions are not explicitly represented. A particular transition, or more generally, the value update of a given variable is possible if there exists a parametrisation enabling it in the associated parametrisation set. The transitions thus have to be drawn from parametrisation set, which specifies whether a variable value can increase or decrease within each regulator state.

Definition 7.7 (Value Update Enabled in a Regulator State). Let $\mathcal{G}$ be a directed parametric regulatory network of dimension n, $\mathcal{P} \subseteq \mathbb{P}(G_{\boldsymbol{m}})$ a parametrisation set, and $\boldsymbol{\omega} \in \Omega_v$ be an arbitrary regulator state of some variable some variable $v \in \{1, \ldots, n\}$ of $\mathcal{G}$.

Then a value update $\mu = v_x \to v_y$, such that $x = \boldsymbol{\omega}_v$ in case $v \in R(v)$, is enabled in a regulator state $\boldsymbol{\omega} \in \Omega_v$ under $\mathcal{P}$ if there exists a parametrisation $\boldsymbol{P} \in \mathcal{P}$ such that $sign\,(x - \boldsymbol{P}_{v,\boldsymbol{\omega}}) = sign\,(x - y)$.

Given a value update μ, we refer to regulator states $\boldsymbol{\omega}$ which do not enable μ as *bad regulator states*.

Seeing as the reduction procedure constructs the objectives based on the regulators of the variable in question, the transitions of parametric regulatory networks as given by the semantics which depend on the entire states of the network, are highly unsuitable. Instead, to maximise the efficiency of the reduction procedure, a minimalistic, with respect to the number of regulators, representation of the value evolutions is desirable. One such obvious reduction is the projection from the entire state to the regulators themselves in the form of the regulator states, which we already employ within the parametrisations. A particular change in the value of a given variable is possible if there exists a parametrisation enabling it in the associated parametrisation set. The criterion being existential with respect to parametrisations, it is a common occurrence for a variable to be enabled to update value in a given direction within numerous regulator states. Enumerating the transitions for each of the regulator states individually could thus still lead to substantial redundancy. In particular, if the value update is enabled within all regulator states that differ only in the value of a particular regulator $u \in R(v)$, i.e. if the value update is possible regardless of the value of u, the regulator u can be omitted as no trace that changes the value of u for the express purpose of changing value of v is minimal. To minimise the amount of regulators to be analysed within the model reduction procedure, we introduce a *partial regulator state* as a union over several regulator states characterised by only a subset of regulators being evaluated.

Definition 7.8 (Partial Regulator State). Let $G_{\boldsymbol{m}}$ be a parametric regulatory network of dimension n and let $v \in \{1, \ldots, n\}$ be a variable of $G_{\boldsymbol{m}}$.

Then a partial regulator state of v is a vector $\boldsymbol{\aleph} \in \prod^{\blacktriangleleft}_{u \in R(v)} \{0, \ldots, \boldsymbol{m}_u\} \cup \{\star\}$ assigning a value or a wildcard $\star$ to each regulator u of v.

By abuse of notation, a partial regulator state is also a set of regulator states, $\boldsymbol{\aleph} = \{\, \boldsymbol{\omega} \in \Omega_v \mid \forall\, u \in R(v),\ \boldsymbol{\omega}_u = \boldsymbol{\aleph}_u \vee \boldsymbol{\aleph}_u = \star \,\}$.

$\mathcal{A}_v = \prod^{\blacktriangleleft}_{u \in R(v)} \{0, \ldots, \boldsymbol{m}_u\} \cup \{\star\}$ denotes the set of all partial regulator states of a variable $v \in \{1, \ldots, n\}$.

Partial regulator states can be utilised to abstract the parametric regulatory network dynamics while minimising the number of repeated values for each regulator. We capture these abstractions by the means of sets of partial regulator states, called *regulation cover sets*, representing the enabling condition of a given value update. A regulation cover set of a value update $v_x \to v_y$ is subject to two conditions. First, the set has to cover all regulator states $\boldsymbol{\omega} \in \Omega_v$ such that $v_x \to v_y$ is enabled in $\boldsymbol{\omega}$. I.e., for each such regulator state there must exist one or more partial regulator states which specify the value

of each regulator in $\boldsymbol{\omega}$. Second, no bad regulator state $\boldsymbol{\omega}$, in which the $v_x \to v_y$ is not enabled, is subsumed by any of the partial regulator states in the cover set. The two conditions not only guarantee that the abstract dynamics enable exactly the same value changes as the concrete dynamics, but also preserve the regulator information, i.e. each value of each regulator that appears in the enabling conditions. The regulator information is necessary to accurately determine which regulator values are necessary to complete an objective.

Definition 7.9 (Regulation Cover Set). Let $\mathcal{G}$ be a directed parametric regulatory network of dimension n, $\mathcal{P} \subseteq \mathbb{P}(G_{\boldsymbol{m}})$ a parametrisation set and let $\mu = v_x \to v_y$ a value update of variable $v \in \{1, \ldots, n\}$ from x to y.

A set of partial regulator states $\mathcal{A}_\mu \subseteq \mathcal{A}_v$ is a cover set of μ if both of the following conditions are satisfied:

- For all regulator states $\boldsymbol{\omega} \in \Omega_v$ such that μ is enabled in $\boldsymbol{\omega}$ under $\mathcal{P}$, and for all regulators $u \in R(v)$, there exists a partial regulator state $\boldsymbol{\aleph} \in \mathcal{A}_\mu$, such that $\boldsymbol{\omega} \in \boldsymbol{\aleph} \wedge \boldsymbol{\omega}_u = \boldsymbol{\aleph}_u$.

- For all bad regulator states $\boldsymbol{\omega} \in \Omega_v$ such that μ in $\boldsymbol{\omega}$ is not enabled under $\mathcal{P}$, $\boldsymbol{\omega} \notin \bigcup_{\boldsymbol{\aleph} \in \mathcal{A}_\mu} \boldsymbol{\aleph}$.

Any regulation cover set, including the concrete regulation cover set containing only fully specified regulation states $\{\, \boldsymbol{\omega} \in \Omega_v \mid \mu \text{ is enabled in } \boldsymbol{\omega} \,\}$, may be used for the express purposes of the reduction procedure. The aim of the regulation cover set is to minimise the number of individual regulator values which appear across all of the partial regulator states. In Section 7.2, we give an example of an algorithm for computation of regulation cover sets with no more regulator value specifications than the concrete regulation cover set.

Since parametric regulatory networks allow only unitary value changes, the realisation of an objective $v_x \rightsquigarrow v_y$ involves a monotonic evolution of value of variable v from x to y, where each update of value depends on specific (partial) regulator state. This coupling of a value change with a corresponding partial regulator state is referred to as a *partial transition*.

The reduction of directed parametric regulatory networks relies on associating to objectives the set of partial transitions which are necessary to realise the objective. Starting from the final (goal) objective, the procedure then recursively collects objectives related to the identified partial transitions.

Definition 7.10 (Objective Transition Set). Let $\mathcal{G}$ be an directed parametric regulatory network of dimension n, $\mathcal{P} \subseteq \mathbb{P}(G_{\boldsymbol{m}})$ a set of parametrisations and let $O = v_x \rightsquigarrow v_y$ be an objective for a variable $v \in \{1, \ldots, n\}$.

Let first $\mu(O)$ be the set of all value updates *covered* by the objective O defined as follows:

$$\mu(O) \triangleq \left\{ v_z \to v_a \;\middle|\; \begin{array}{l} \mathit{sign}(v_z \to v_a) = \mathit{sign}(O) \wedge \\ \max(\{z, a\}) \leq \max(\{x, y\}) \wedge \\ \min(\{z, a\}) \geq \min(\{x, y\}) \end{array} \right\}$$

Then the objective transition set $\tau(O)$ is a set of partial transitions composed of a covered value update and a covered partial regulator state:

$$\tau(O) \triangleq \{ (\mu, \aleph) \mid \mu \in \mu(O) \wedge \aleph \in \mathcal{A}_\mu \}$$

Given an initial state $\boldsymbol{x} \in X_{\boldsymbol{m}}$, the *valid objective transition set* of an objective O in state $\boldsymbol{x}$ is the subset of the objective transition set $\tau_{\boldsymbol{x}}(O) \subseteq \tau(O)$ such that $(\mu, \aleph) \in \tau_{\boldsymbol{x}}(O) \overset{\Delta}{\iff} \forall u \in R(v),\ \aleph_u \neq \star \implies \boldsymbol{x}_u \rightsquigarrow \aleph_u$ is valid in the initial state $\boldsymbol{x}$.

The (valid) objective transition sets extend to sets of objectives in the natural manner, $\tau(\mathcal{O}) = \bigcup_{O \in \mathcal{O}} \tau(O)$.

Remark that the definition of a valid objective transition set benefits from the use of partial regulator states. Indeed, instead of having to check validity of an objective for each regulator, only the minimal necessary subset of regulators is considered. Checking objective validity consists of finding a trace witness, which translates to finding all possible extensions (enabled value updates) of a trace. The parametrisation lattices used in the abstract semantics of parametric regulatory networks allow searching for enabled value updates without explicitly enumerating the parametrisations. The objective validity computation is thus compatible with the abstraction of parametrisation sets.

The goal-oriented reduction of directed parametric regulatory networks can then be defined by recursively collecting objectives from partial transitions into an *reduced objective set* $\mathcal{B}$, and refining the component activation and inhibition limits accordingly.

Definition 7.11 (Directed Parametric Regulatory Network Reduction Procedure). Let $\mathcal{G} = \left(G_{\boldsymbol{m}}, \boldsymbol{l}^A, \boldsymbol{l}^I\right)$ be a directed parametric regulatory network of dimension n, $\boldsymbol{x} \in X_{\boldsymbol{m}}$ an arbitrary initial state and $g_\top$ a goal.

Then, the goal-driven reduction of $\mathcal{G}$ is again a directed network $\mathcal{G}' = \left(G_{\boldsymbol{m}}, \boldsymbol{l}^{A'}, \boldsymbol{l}^{I'}\right)$ of the same dimension n where the underlying parametric regulatory network is unchanged and the limit vectors $\boldsymbol{l}^{A'}$ and $\boldsymbol{l}^{I'}$ are defined as follows for each regulator state $(v, \boldsymbol{\omega}) \in \Omega$:

$$\boldsymbol{l}^{A'}_{\boldsymbol{\omega}} = \max(\{ x \in \{0, \ldots, \boldsymbol{m}_v\} \mid \exists (v_{x-1} \to v_x, \aleph) \in \tau_{\boldsymbol{x}}(\mathcal{B}),\ \boldsymbol{\omega} \in \aleph \} \cup \{-\infty\})$$

$$\boldsymbol{l}^{I'}_{\boldsymbol{\omega}} = \min(\{ x \in \{0, \ldots, \boldsymbol{m}_v\} \mid \exists (v_{x+1} \to v_x, \aleph) \in \tau_{\boldsymbol{x}}(\mathcal{B}),\ \boldsymbol{\omega} \in \aleph \} \cup \{\infty\})$$

where $\mathcal{B}$ is the smallest set of objectives satisfying all of the following conditions:

1. $\boldsymbol{x}_g \rightsquigarrow g_\top \in \mathcal{B}$;
2. For each $O \in \mathcal{B}$, each $(w_x \to w_y, \aleph) \in \tau_{\boldsymbol{x}}(O)$, and each $u \in R(w) \setminus \{w\}$, $\aleph_u \neq \star \implies \boldsymbol{x}_u \rightsquigarrow \aleph_u \in \mathcal{B}$;
3. For each $O \in \mathcal{B}$, each $(w_x \to w_y, \aleph) \in \tau_{\boldsymbol{x}}(O)$, and each $w_z \rightsquigarrow w_a \neq O \in \mathcal{B}$, $w_y \rightsquigarrow w_a \in \mathcal{B}$.

Following the interpretation of the reduction procedure a transition $t = (\boldsymbol{y}, \boldsymbol{z})$ is preserved in the reduced network $\mathcal{G}'$ if there exists a partial transition $(\mu(t), \aleph) \in \tau_{\boldsymbol{x}}(\mathcal{B})$ where $\omega_{v(t)}(\boldsymbol{y}) \in \aleph$. In particular, such partial transition must exist in the valid objective transition set of some objective $O \in \mathcal{B}$, $(\mu(t), \aleph) \in \tau_{\boldsymbol{x}}(O) \subseteq \tau_{\boldsymbol{x}}(\mathcal{B})$. As the objectives are realised in monotonic fashion in multivalued networks, we know such an O covers $\mu(t)$. The claim is formalised in Lemma 7.1.

Lemma 7.1 (Transitions Covered by an Objective in the Reduced Objective Set are Represented in the Valid Objective Transition Set)**.** *Let $\mathcal{G}$ be a directed parametric regulatory network of dimension n and let π be a trace of $\mathcal{G}$ reaching a goal $g_\top$ from an initial state $\boldsymbol{x} \in X_{\boldsymbol{m}}$. Let further $\mathcal{B}$ be the objective set constructed by the reduction procedure according to Definition 7.11. Finally, let $O = v_x \rightsquigarrow v_y \in \mathcal{B}$ be an arbitrary objective in the reduced objective set.*

Then, for any $i \in \{1, \ldots, |\pi|\}$ such that $\pi_i \in \mu(O)$, there exists $(\pi_i, \aleph) \in \tau_{\boldsymbol{x}}(O)$ where $\omega_v\left(\pi^{:i-1}(\boldsymbol{x})\right) \in \aleph$.

Proof. Since π is a trace of $\mathcal{G}$, we know the value update π_i is enabled in the regulator state $\boldsymbol{\omega} = \omega_v\left(\pi^{:i-1}(\boldsymbol{x})\right)$. Thus, by Definition 7.9 of the regulation cover sets, we know there must exist at least one partial regulator state $\aleph \in \mathcal{A}_{\pi_i}$ such that $\boldsymbol{\omega} \in \aleph$.

Then, by Definition 7.10 of objective transition sets, the corresponding partial transition $(\pi_i, \aleph) \in \tau(O)$. Finally, since π itself is a witness of the validity of objectives for all regulators required by $(\pi_i, \aleph)$, $(\pi_i, \aleph) \in \tau_{\boldsymbol{x}}(O)$. □

This leads us to formulate the soundness theorem of the reduction procedure, guaranteeing that all transitions witnessing all of the minimal traces are preserved and thus, in turn, all minimal traces are preserved. The proof of the theorem relies on Lemma 7.1 to show that any value update whose associated transition is not preserved is part of a cycle on any trace leading to the goal, and as a consequence does not belong to any minimal trace.

Theorem 7.1. *Let $\mathcal{G}$ be a directed parametric regulatory network of dimension n and let π be a trace of $\mathcal{G}$ from the initial state $\boldsymbol{x} \in X_{\boldsymbol{m}}$ minimal for a goal $g_\top$. Let further $\mathcal{B}$ be set of objectives constructed for reachability of $g_\top$ from $\boldsymbol{x}$ according to Definition 7.11.*

Then, for any $i \in \{1, \ldots, |\pi|\}$ with $\pi_i = v_x \to v_y$, there exists at least one partial transition $(\pi_i, \aleph) \in \tau(\mathcal{B})$ such that $\omega_v\left(\pi^{:i-1}(\boldsymbol{x})\right) \in \aleph$.

Proof. We conduct the proof by contradiction, showing that if a value change $v_x \to v_y = \pi_i$ for some $i \in \{0, \ldots, |\pi|\}$, is not covered by any objective in $\mathcal{B}$, the trace π cannot be minimal.

Let now $j < i$ be the largest such that $v(\pi_j) = v$ and for which there exists $O \in \mathcal{B}$, $\pi_j \in \mu(O)$, if it exists, $j = 0$ otherwise. Furthermore, let $k > i$ be the smallest such that $v(\pi_k) = v$ and $\pi^{:k-1}(\boldsymbol{x})_v = \pi^{:j}(\boldsymbol{x})_v$, if it exists, $k = |\pi| + 1$ otherwise.

Let us now consider a value update sequence ρ obtained from π by removing all the value updates of variable v in $\pi^{j+1:k-1}$. The removed value updates either form a loop on the value of v or, in case $j = |\pi| + 1$, have no causal successors modifying the value of v. The evolution of v along ρ is therefore valid with respect to evolving the value by steps of size 1. For ρ to satisfy the minimality condition in Definition 7.4 with respect to π, the regulator states use by each value update have to be the same.

Let us therefore assume there exists $l \in \{1, \ldots, |\pi|\}$ with $w = v(\pi_l)$ such that $\omega_w\left(\pi^{:l-1}(\boldsymbol{x})\right) \neq \omega_w\left(\rho^{:h-1}(\boldsymbol{x})\right)$ where h is the new index of π_l in ρ, $\pi_l = \rho_h$. As ρ only differs from π by the evolution of variable v value between π_j and π_k, π_l must be a value change of a variable $w \neq v$, such that $v \in R(w)$ and $j < l < k$.

We now show $\rho_l \notin \mu(O)$ for any $O \in \mathcal{B}$ by contradiction. Let thus $O' \in \mathcal{B}$ be such that $\rho_l \in \mu(O')$. $v \in R(w)$ and thus by rule (2) of the reduced objective set construction in Definition 7.11, $\boldsymbol{x}_v \rightsquigarrow z \in \mathcal{B}$ where $z = \pi^{:l}(\boldsymbol{x})_v$. We now conduct a discussion on the value of j.

- $j = 0$. $p\left(\rho^{:h}, \boldsymbol{x}\right) = \emptyset$ and $p\left(\pi^{:l}, \boldsymbol{x}\right) \neq \emptyset$ implies that $\rho^{:h}(\boldsymbol{x})_v \neq z$. There thus must exist a value change from $\boldsymbol{x}_v$ towards z. Such value change is however, covered by $\boldsymbol{x}_v \rightsquigarrow z$ contradicting $j = 0$.

- $j > 0$. By definition, $\pi_j \in \mu(O)$ where $O \in \mathcal{B}$. Let a be the target value of O. Then, by rule (3) of the reduced objective set construction in Definition 7.11, $v_a \rightsquigarrow v_z \in \mathcal{B}$. The objective $v_a \rightsquigarrow v_z$, respectively, O itself in case $\pi^{:j}(\boldsymbol{x})_v \neq a$ and $sign\left(z - \pi^{:j}(\boldsymbol{x})_v\right) = sign(O)$, covers the first value change of v after π_i. This is a contradiction with j being the largest index of a covered value change or, in case π_i is the first change of variable v value after π_j, with π_i not being covered.

As such, the value change ρ_h is not covered by any objective in $\mathcal{B}$ Therefore, ρ_h can be removed from ρ in the same fashion as π_i was removed from π. Repeating the whole procedure leads to an even shorter value update sequence. As any minimal trace is finite, all uncovered value updates are eventually purged, leaving a valid trace.

Thanks to rule (1) of the reduced objective set construction in Definition 7.11, the objective $\boldsymbol{x}_g \rightsquigarrow \top \in \mathcal{B}$ covers all value updates of variable g from $\boldsymbol{x}_g$ to $\top$. However, even covered value updates may be removed if they lie between π_j and π_k. We thus still have to show that for any covered value update π_l for $j < l < k$ which gets removed in ρ, there exists $h \leq j$ such that $\pi^{:h}(\boldsymbol{x})_v = \pi^{:l}(\boldsymbol{x})_v$.

Let thus $O' \in \mathcal{B}$ be such that $\pi_l \in \mu(O')$ and let z be the target value of the objective O'. By definition of the reduced objective set $\mathcal{B}$, any target value of an objective is first introduced by either rule (1) or (2) of Definition 7.11, giving us $\boldsymbol{x}_v \rightsquigarrow z \in \mathcal{B}$.

Let us first show $j > 0$ by contradiction. We can assume $\pi^{:l-1}(\boldsymbol{x})_v = \boldsymbol{x}_v$, otherwise the first value update of variable v towards $\pi^{:l-1}(\boldsymbol{x})_v$ is covered by $\boldsymbol{x}_v \rightsquigarrow z$ contradicting $j = 0$. Furthermore, π_l itself is covered, giving us $i < l$.

Finally, k is the smallest index beyond i of a value update changing the value of variable v from $\boldsymbol{x}_v$. Therefore, $k \leq l$, which is direct contradiction of $l < k$.

We thus know there exists a value update $\pi_j \in \mu(O)$ covered by the objective $O \in \mathcal{B}$. Let a be the target value of the objective O, thus $\boldsymbol{x}_v \rightsquigarrow a \in \mathcal{B}$. Furthermore, by rule (3) of the reduced objective set construction in Definition 7.11, $v_z \rightsquigarrow v_a, v_a \rightsquigarrow v_z \in \mathcal{B}$.

Let us now conduct a discussion on the sign $s \in \{-1, 1\}$ of the first removed value update, i.e. the first value update of variable v after π_j.

- $s = sign(\pi_j)$. Then, as no value update between π_j and π_i is covered, $a = \pi^{:j}(\boldsymbol{x})_v$. By a similar argument with the objective $v_a \rightsquigarrow v_z \in \mathcal{B}$, and using the fact that $k > l$ is the first value update of v starting from value a, $\pi^{:j}(\boldsymbol{x})_v = z = a = \pi^{:l}(\boldsymbol{x})_v$.

- $s = -sign(\pi_j)$. We know $O = \boldsymbol{x}_v \rightsquigarrow a$. Otherwise, O would have to have been added to $\mathcal{B}$ by rule (3) of Definition 7.11. The rule (3) being symmetrical, the reverse objective of O, which covers the first value update of variable v after π_j, would therefore also have to belong to $\mathcal{B}$. A similar argument can be used to obtain $O' = \boldsymbol{x}_v \rightsquigarrow z$. The objective from a to the initial value of O', which again covers the first value update of variable v after π_j, would have belonged to $\mathcal{B}$ otherwise. Finally, since j is the last index of a covered value update before π_i, it is in particular not covered by $v_a \rightsquigarrow v_z$, giving us $sign(O') = sign(O)$. Thanks to $l < k$, we know $\pi^{:l}(\boldsymbol{x})_v \leq \pi^{:j}(\boldsymbol{x})_v$ and due to the value updates being of steps of size 1, there must exist the coveted $h \leq j$ with $\pi^{:l}(\boldsymbol{x})_v = \pi^{:h}(\boldsymbol{x})_v$ facilitating the evolution of variable v value from $\boldsymbol{x}_v$ to $\pi^{:j}(\boldsymbol{x})_v$.

As such, none of the variable values reached by a covered value update may be lost during the removal procedure. The newly obtained trace is therefore guaranteed to reach the goal, contradicting the minimality of π. □

7.2 Computation of Regulation Cover Sets

This section introduces a sample algorithm for computation of the regulation cover sets. The algorithm relies on a simple heuristic for choosing partial regulators states to ensure each enabled regulator state is covered, while maintaining the condition that no bad regulator state is covered, thus complying with Definition 7.9. The regulation cover set computed by the algorithm in this section is guaranteed to not be larger than the concrete regulation cover set, with respect to the number of regulator value specifications across all the partial regulator states in the cover set (values other than the wildcard $\star$).

Throughout this section we limit ourselves to computation of a single regulation cover set, for a given parametric regulatory network $\mathcal{G}$, value change $v_x \to v_y$ and parametrisation set $\mathcal{P}$. Let now $\mathcal{A}_{ena} \triangleq \left\{\aleph \in \mathcal{A}_v \;\middle|\; \forall \boldsymbol{\omega} \in \aleph,\ \exists \boldsymbol{P} \in \mathcal{P},\ \boldsymbol{P}_{v,\boldsymbol{\omega}} = y\right\}$ denote the set of all partial regulator states that subsume no bad regulator state (contain only regulator states enabling $v_x \to v_y$). Further,

we use $\mathcal{A}_i \triangleq \{\, \aleph \in \mathcal{A}_v \mid |\{\, u \in R(v) \mid \aleph_u = \star \,\}| = i \,\}$ where $i \in \{0, \ldots, |R(v)|\}$ to denote the sets of all partial regulator states ranked by the number of wildcard values. $\mathcal{A}_0$ is thus isomorphic to the regulator state set Ω_v while $\aleph_{|R(v)|} = \left\{ \{\star\}^{|R(v)|} \right\}$ is the singleton set containing the partial regulator state which assigns wildcard value to each regulator.

The algorithm consists of choosing partial regulator state set, *local cover set* $\mathcal{A}_{\boldsymbol{\omega}}$, to cover each (concrete) regulator state $\boldsymbol{\omega}$ enabling the value change. The extension sets $\mathcal{A}_{\boldsymbol{\omega}}$ are computed separately for each regulator state in increasing order of a suitable weight function $\mathcal{W}$. The weight function is constructed to represent the flexibility of how the particular regulator state may be covered. I.e. The more partial regulator states $\aleph \in \mathcal{A}_{ena}$ such that $\boldsymbol{\omega} \in \aleph$ exist, the larger the result of the weight function for $\boldsymbol{\omega}$. By ensuring the regulator states $\boldsymbol{\omega}$ with few possible local cover sets $\mathcal{A}_{\boldsymbol{\omega}}$ are covered first using the weight function, it becomes possible to choose such cover sets $\mathcal{A}_{\boldsymbol{\omega}}$ for the remaining regulator states that are most compatible with the already included partial regulator states, minimising redundancy. To further amplify this benefit, the algorithm keeps track of partial regulator states that are *removed* from further computation, denoted $\mathcal{A}_{rmv}$. In particular, every time a local cover set $\mathcal{A}_{\boldsymbol{\omega}}$ is picked for a regulator state $\boldsymbol{\omega}$, all partial regulator states $\aleph \in \mathcal{A}_{ena} \setminus \aleph_{\boldsymbol{\omega}}$ such that $\boldsymbol{\omega} \in \aleph$ are removed, $\aleph \in \mathcal{A}_{rmv}$. This ensures that no redundancy is introduced to the covering of $\boldsymbol{\omega}$ in further computation. Finally, to reflect the possibility of many local cover sets for a particular regulator state $\boldsymbol{\omega}$ being disabled due to partial regulator state being removed, the weight function takes the removed states into consideration and depends on the size of $\{\, \aleph \in \mathcal{A}_{ena} \setminus \mathcal{A}_{rmv} \mid \boldsymbol{\omega} \in \aleph \,\}$.

The local cover set $\mathcal{A}_{\boldsymbol{\omega}}$ is chosen from subsets of $\mathcal{A}_i$ limited to partial regulator states $\aleph$ such that $\boldsymbol{\omega} \in \aleph$, in decreasing order of i. Such a local cover set surely exists among the subsets of $\mathcal{A}_i$ as for $i = 0$, the relevant subset contains only the singleton set containing the regulator state itself. Once a suitable local cover set $\mathcal{A}_{\boldsymbol{\omega}}$ is obtained, it is directly included in the final regulation cover set $\mathcal{A}_\mu$. As aforementioned, the remaining relevant partial regulator states, which do not belong to the selected local cover set, are removed at the same time.

As the weight function essentially counts partial regulator states, it in general only gives a partial order on the regulator states. As such, the algorithm is forced to make nondeterministic choices. Such a situation occurs, however, only when the outcomes of the choice are isomorphic from the perspective of further computation. Therefore, an arbitrary extension of the partial order given by the weight function (e.g. lexicographic order) can be chosen to obtain a fully deterministic algorithm. The pseudocode of the sample algorithm for regulation cover set inference is given in Algorithm 1.

The correctness of the algorithm comes directly from the construction. No bad states may be included as the algorithm works only with the set of partial regulator states which include no bad states. On the other hand, all regulator states which enable the value change are fully covered as the algorithm ensures this for each of them individually.

The resulting cover set computed by Algorithm 1 contains no more explicit

Algorithm 1 Computation of Regulation Cover Set

function WEIGHT($\boldsymbol{\omega}$)
  return $|\{\, \boldsymbol{\aleph} \in (\mathcal{A}_1 \cap \mathcal{A}_{ena} \setminus \mathcal{A}_{rmv}) \mid \boldsymbol{\omega} \in \boldsymbol{\aleph} \,\}| + \frac{|\{\, \boldsymbol{\aleph} \in \mathcal{A}_1 \cap \mathcal{A}_{ena} \mid \boldsymbol{\omega} \in \boldsymbol{\aleph} \,\}|}{|R(v)|+1}$
end function

function COMPUTEREGULATIONCOVERSET($v_x \to v_y$, $\mathcal{P}$)
  $\mathcal{A}_\mu \leftarrow \emptyset$
  $\mathcal{A}_{ena} \leftarrow \{\, \boldsymbol{\aleph} \in \mathcal{A}_v \mid \forall \boldsymbol{\omega} \in \boldsymbol{\aleph},\ \exists \boldsymbol{P} \in \mathcal{P},\ \boldsymbol{P}_{v,\boldsymbol{\omega}} = y \,\}$
  $\mathcal{A}_{rmv} \leftarrow \emptyset$
  for $i = 0$ to $|R(v)|$ **do**
    $\mathcal{A}_i \leftarrow \{\, \boldsymbol{\aleph} \in \mathcal{A}_v \mid |\{\, u \in R(v) \mid \boldsymbol{\aleph}_u = \star \,\}| = i \,\}$
  end for
  while $\mathcal{A}_0 \setminus \mathcal{A}_{rmv} \neq \emptyset$ **do**
    $\boldsymbol{\omega} \leftarrow \min_{\boldsymbol{\omega}' \in (\mathcal{A}_0 \cap \mathcal{A}_{ena}) \setminus \mathcal{A}_{rmv}} (\text{WEIGHT}(\boldsymbol{\omega}'))$
    $\mathcal{A}_{\boldsymbol{\omega}} \leftarrow \emptyset$
    $i \leftarrow |R(v)| - 1$
    while $\boldsymbol{\omega}$ is not covered by $\mathcal{A}_\mu \cup \mathcal{A}_{\boldsymbol{\omega}}$ **do**
      $\mathcal{A}_{\boldsymbol{\omega}} \leftarrow (\mathcal{A}_i \cap \mathcal{A}_{ena}) \setminus \mathcal{A}_{rmv}$
      $i \leftarrow i - 1$
    end while
    $\mathcal{A}_\mu \leftarrow \mathcal{A}_\mu \cup \mathcal{A}_{\boldsymbol{\omega}}$
    $\mathcal{A}_{rmv} \leftarrow \mathcal{A}_{rmv} \cup \{\, \boldsymbol{\aleph} \in \mathcal{A}_v \mid \boldsymbol{\omega} \in \boldsymbol{\aleph} \,\}$
  end while
  return $\mathcal{A}_\mu$
end function

regulator value specifications than the concrete regulation cover set. This is a consequence of the order in which the individual regulator states are handled. Suppose a regulator state $\boldsymbol{\omega}$ is covered by several partial regulator states which contain more regulator value specifications than $\boldsymbol{\omega}$ itself. Each partial regulator state $\boldsymbol{\aleph} \in \mathcal{A}_1$ with $\boldsymbol{\aleph}_u = \star$ is shared with exactly $\max(u) - 1$ other regulator states. Thus, the partial regulator states included to cover $\boldsymbol{\omega}$ can be utilised while covering $\max(u)-1$ other regulator states. Finally, since $\mathcal{W}(\boldsymbol{\omega}) \geq 2$ is the smallest weight among all uncovered regulator states, all the other uncovered regulator states are also sharing partial regulator states among themselves, thus closing the loop and guaranteeing the regulator value specification debt eventually gets 'payed off'.

The fractional part of the weight function is included to introduce bias towards states that have less partial regulator states in the beginning due to sharing more partial regulator states with bad regulator states. If there are two regulator states $\boldsymbol{\omega}$ and $\boldsymbol{\omega}'$ such that $\lfloor \mathcal{W}(\boldsymbol{\omega}) \rfloor = \lfloor \mathcal{W}(\boldsymbol{\omega}') \rfloor$ but $\mathcal{W}(\boldsymbol{\omega}) < \mathcal{W}(\boldsymbol{\omega}')$, we know that both regulator states have equally many partial regulator states to choose from for their respective cover sets. However, more of the partial regulator states containing $\boldsymbol{\omega}'$ have been removed and thus, quite possibly,

have also been included in the regulation cover set $\mathcal{A}_\mu$. $\boldsymbol{\omega}'$ is therefore in all likelihood already covered to a higher degree than $\boldsymbol{\omega}$, and likely has more covering options. The bias thus ensures $\boldsymbol{\omega}$ is covered first in order to avoid introducing potentially redundant partial regulator states into the regulation cover set.

Algorithm 1 is quasilinear in the number of regulator states and quadratic in the number of regulators. The main complexity comes from computing the local cover sets $\mathcal{A}_{\boldsymbol{\omega}}$. Whether a regulator state $\boldsymbol{\omega} \in \Omega_v$ is covered by some partial regulator state set, in particular, $\mathcal{A}_\mu \cup \mathcal{A}_{\boldsymbol{\omega}}$, can be decided in $\mathcal{O}(|R(v)|)$. Only the relevant subsets of $\mathcal{A}_i$ for each $i \in \{0, \ldots, |R(v)|\}$ are considered for the local cover sets. Thus, each $\boldsymbol{\omega} \in \Omega_v$ tests at most $|R(v)|$ local cover sets, although usually much less. As such, the local cover set of a single regulator state can be computed in $\mathcal{O}\left(|R(v)|^2\right)$. Given that local cover sets are computed for each regulator state which enables the value change, computing all the local cover sets takes asymptotically $\mathcal{O}\left(|\Omega_v| \cdot |R(v)|^2\right)$.

Finally, one has to consider the complexity of keeping the regulator states in a priority queue according to the weight function $\mathcal{W}$. The asymptotic time complexity of the complete Algorithm 1 is therefore quasilinear in the number of regulator states $\mathcal{O}\left(|\Omega_v| \cdot \left(log\left(|\Omega_v|\right) + |R(v)|^2\right)\right)$.

Algorithm 1 does not require explicit enumeration of parametrisations when coupled with the abstract parametric regulator network semantics. The parametrisation set is only used to determine which regulator states enable the value change (queries to $\mathcal{A}_{ena}$). This information is readily available using the parametrisation lattices in the form of parameter values for the relevant regulator state in the minimum parametrisation and the maximum parametrisation.

7.3 Examples

In this section we present an example of directed parametric regulatory network reduction by the means of the reduction procedure from Definition 7.11, including the computation of the regulation cover set by the algorithm Algorithm 1.

Example 7.1. *Consider the parametric regulatory network $G'_{\{1\}^4}$ from Example 5.1 as a directed parametric regulatory network $\mathcal{G} = \left(G'_{\{1\}^4}, \boldsymbol{l}^A, \boldsymbol{l}^I\right)$, where $\boldsymbol{l}^A = \{1\}^4$ and $\boldsymbol{l}^I = \{0\}^4$ are unrestrictive. Let further $\mathcal{P} = \{\boldsymbol{P}, \boldsymbol{P}'\}$ be a parametrisation set containing only two parametrisations, $\boldsymbol{P}$ being the parametrisation from Figure 5.2 defining the Boolean network F_B from Example 2.2. And $\boldsymbol{P}' = \boldsymbol{P}[a, 100 \mapsto 0]$ be a parametrisation differing from $\boldsymbol{P}$ only on the value of parameter $K_{a,100}$. Finally, let a_1 be a goal and $\boldsymbol{x} = (a = 0, b = 0, c = 0, d = 0)$ an initial state.*

In Figure 7.1 we recall the dynamics of the Boolean network F_B in the form of the state space graph with asynchronous semantics. As opposed to Figure 2.6, the behaviour enabled solely by parametrisation $\boldsymbol{P}'$ is also represented. The bold

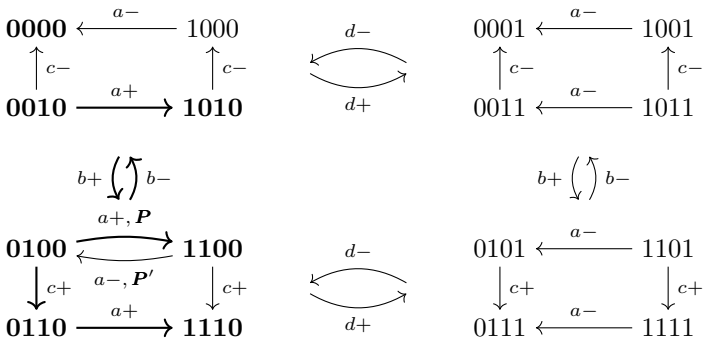


Figure 7.1: The state space graph of the directed parametric regulatory network $\mathcal{G}$ with parametrisation set $\{\boldsymbol{P}, \boldsymbol{P}'\}$. Transitions changing the value of b and d are displayed schematically. Transitions only enabled by a single of the parametrisations are labelled by the respective parametrisations. Bold font and lines indicate states and transitions used by at least one minimal trace from the initial state to the goal a_1.

lines and text indicate transitions belonging to a minimal trace to the goal a_1 and states visited by the minimal traces, respectively.

In our example, three distinct minimal traces from the initial state $\boldsymbol{x}$ to the goal a_1 exist. Here, we list the minimal traces including the traversed states for clarity. The value updates are annotated by their variable, the nature of the value change and the regulator state:

$$0000 \xrightarrow{b+,0} 0100 \xrightarrow{a+,100} 1100$$
$$0000 \xrightarrow{b+,0} 0100 \xrightarrow{c+,1} 0110 \xrightarrow{a+,110} 1110$$
$$0000 \xrightarrow{b+,0} 0100 \xrightarrow{c+,1} 0110 \xrightarrow{b-,1} 0010 \xrightarrow{a+,010} 1010$$

Notice that all the listed traces share a common prefix. In fact, if only the value updates, without the regulator states were considered in Definition 7.4, only the first, shortest, trace would me considered minimal. All traces, however utilise a different regulator state, thus effectively a different transition to increase the value of variable a and reach their final state. Observe also that the first, shortest, minimal trace is only available under the parametrisation $\boldsymbol{P}$. Thanks to the regulator state equality condition, we thus preserve minimal traces also for $\boldsymbol{P}'$ without the need to separate them.

Observe that variable d never changes value along any of the minimal traces. This follows from the fact that variable a is never allowed to increase while variable d value is 1. Thus, if variable d value increases, it has to decrease again before the goal can be reached. Variable d is only regulator of itself and variable a and has therefore no other effects on the network. Unlike the increase

and decrease loop of variable b value in the third, longest, minimal trace, which is necessary for the increase of variable c value, the increase and decrease loop of variable d can therefore always be stripped from the trace to obtain a shorter, more minimal, trace. One might thus expect the value updates of variable d to be pruned by the reduction procedure, which is, indeed the case:

We start with $\mathcal{B} := \{a_0 \rightsquigarrow a_1\}$ *according to rule (1) of Definition 7.11.*

Inference of the regulator cover set used in $\tau_{a_0 \rightsquigarrow a_1}(\boldsymbol{x}) = \{(a_0 \to a_1, \aleph) \mid \aleph \in \mathcal{A}_{a_0 \to a_1}\} = \{(a_0 \to a_1, 100), (a_0 \to a_1, 010), (a_0 \to a_1, 110)\}$ *is illustrated in Example 7.2. By rule (2) of Definition 7.11:* $\mathcal{B} := \mathcal{B} \cup \{b_0 \rightsquigarrow b_0, b_0 \rightsquigarrow b_1, c_0 \rightsquigarrow c_0, c_0 \rightsquigarrow c_1, d_0 \rightsquigarrow d_0\}$.

For arbitrary variable v, the objective $v_0 \rightsquigarrow v_0$ *has an empty valid transition set* $\tau_{v_0 \rightsquigarrow v_0}(\boldsymbol{x}) = \emptyset$ *and thus neither of rules (2) or (3) are applicable.*

For the remaining objective $b_0 \rightsquigarrow b_1$ *and* $c_0 \rightsquigarrow c_1$*, rule (2) produces only duplicate objectives* $b_0 \rightsquigarrow b_0$ *and* $b_0 \rightsquigarrow b_1$*, respectively. Rule (3), however, may be applied to* $b_0 \rightsquigarrow b_1$ *and* $b_0 \rightsquigarrow b_0$*, as well as* $c_0 \rightsquigarrow c_1$ *and* $c_0 \rightsquigarrow c_0$ *to obtain* $\mathcal{B} := \mathcal{B} \cup \{b_1 \rightsquigarrow b_0, c_1 \rightsquigarrow c_0\}$.

Only duplicate objectives are obtained by further application of either rule (2) or (3). The construction of the $\mathcal{B}$ *thus concludes with* $\mathcal{B} = \{a_0 \rightsquigarrow a_1, b_0 \rightsquigarrow b_0, b_0 \rightsquigarrow b_1, c_0 \rightsquigarrow c_0, c_0 \rightsquigarrow c_1, d_0 \rightsquigarrow d_0, b_1 \rightsquigarrow b_0, c_1 \rightsquigarrow c_0\}$*, with the valid partial transition set* $\tau_{\mathcal{B}}(\boldsymbol{x}) = \{(a_0 \to a_1, 100), (a_0 \to a_1, 010), (a_0 \to a_1, 110), (b_0 \to b_1, 0), (b_1 \to b_0, 1), (c_0 \to c_1, 1), (c_1 \to c_0, 0)\}$*. One may observe that the computed transition set indeed covers all the transitions used by any of the minimal traces (thick edges in Figure 7.1).*

Finally, the limit vectors for the new DPRN $\mathcal{G}' = \left(G'_{\{1\}^4}, \boldsymbol{l}^{A'}, \boldsymbol{l}^{I'}\right)$ *are as follows:*

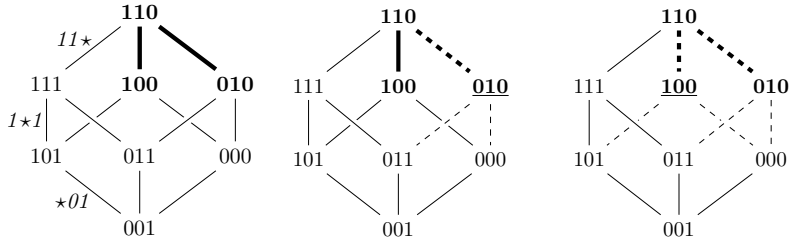
$$\boldsymbol{l}^{A'} = (a = 1, b = 1, c = 1, d = -\infty)$$
$$\boldsymbol{l}^{I'} = (a = \infty, b = 0, c = 0, d = \infty)$$

The variable d is indeed completely forbidden changing value in the reduced model, considerably decreasing the reachable state space that has to be explored. Notice also that decrease of variable a value is also disabled, however, in our Boolean case this has no practical effect w.r.t. reachability of the goal a_1.

Example 7.2. *Let us consider the directed parametric regulatory network* $\mathcal{G} = \left(G'_{\{1\}^4}, \boldsymbol{l}^A, \boldsymbol{l}^I\right)$ *from Example 7.1.*

We now show the regulation cover set computation for value updates of variable a. Let us start with $a_0 \to a_1$*. We visualise the computation directly on the regulator states of variable a represented by the Hasse diagram of the lattice* $(\Omega_a, \preceq_a)$*. The initial configuration and first two iterations, covering of the first two regulator states, are depicted in Figure 7.2.*

Bold font in Figure 7.2 indicates the three regulator states which enable the increase of variable a value. The partial regulator states from $\mathcal{A}_1$ *correspond to edges in the Hasse diagram. E.g. the partial regulator state* $11\star$ *is represented by the edge connecting regulator states* $111, 110 \in 11\star$*. Thick edges indicate*



(a) Initial configuration (b) Configuration after the first iteration covering 010. (c) Configuration after the second iteration covering 100.

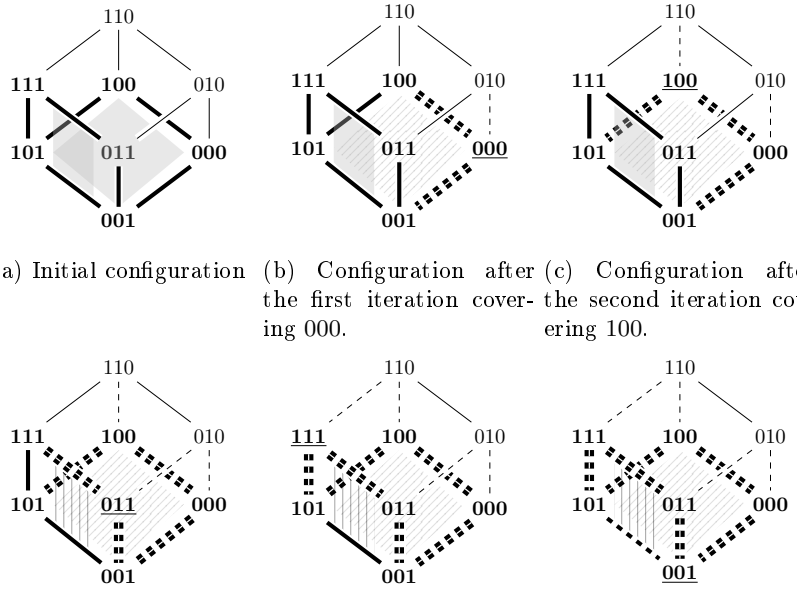
Figure 7.2: Regulator states of variable a during computation of regulation cover set for value update $a_0 \to a_1$ in the form of the Hasse diagram of the monotonicity order $\preceq_a$. Only the leftmost edges in (a) are labelled by the corresponding partial regulator states $11\star, 1\star1$ and $\star01$ for the sake of readability. Bold text and lines indicate partial regulator states which enable the value update, $\mathcal{A}_{ena}$. Underlined regulator state is the state covered in the respective iteration and dashed lines represent removed partial regulator states, $\mathcal{A}_{rmv}$.

partial regulator states which contain no bad regulator states. Partial regulator states from $\mathcal{A}_2$ could in turn be viewed as squares in the diagram. In our case all the partial regulator state in $\mathcal{A}_2$ contain at least one bad regulator state. Assuming all influences are monotonic, a partial regulator state $\aleph$ belonging to $\mathcal{A}_i$ corresponds to a i-dimensional hypercube, in the Boolean case, or i-dimensional hyper-rectangular cuboid, in the general case, in the Hasse diagram, with all contained vertices representing regulator states $\boldsymbol{\omega} \in \aleph$.

The Hasse diagram representation in Figure 7.2 allows computing the weight function at a glance. The weight corresponds to number of neighbouring thick, non-dashed edges plus, the number of neighbouring thick edges divided by $|R(a)|+1$, in our case 4. Consequently, in the initial configuration, Figure 7.2 (a), the regulator states 100 and 010 share an equal minimal weight $\mathcal{W}(010) = \mathcal{W}(100) = 1.25$. The weight equality comes from the regulator states 100 and 010, being in a perfectly symmetrical position in the hypercube structure of the Hasse diagram.

The run of the Algorithm 1 as illustrated in Figure 7.2 assumes lexicographic order is used to distinguish between regulator states with equal weights. Thus, 010 is covered in the first iteration. Only one partial regulator state in $\mathcal{A}_1$, $\star10$ which does not cover $b = 0$, contains 010. The local cover set used for 010 is therefore taken from $\mathcal{A}_0$ and contains only 010 itself, $\mathcal{A}_{010} = \{010\}$. Figure 7.2 (b) depicts the configuration after the first iteration, including the removed partial regulator states, $\mathcal{A}_{rmv}$ represented by the dashed lines.

In the second iteration 100 is covered as $1.25 = \mathcal{W}(100) < \mathcal{W}(110) = 1.5$. As hinted by the symmetric position with respect to 010, 100 is covered in the same fashion, by local cover set $\mathcal{A}_{100} = \{100\}$. The result is shown



(a) Initial configuration

(b) Configuration after the first iteration covering 000.

(c) Configuration after the second iteration covering 100.

(d) Configuration after the third iteration covering 011.

(e) Configuration after the fourth iteration covering 111.

(f) Configuration after the fifth iteration covering 001.

Figure 7.3: Regulator states of variable a during computation of regulation cover set for value update $a_1 \to a_0$ in the form of the Hasse diagram of the monotonicity order $\preceq_a$. Bold text, lines and shaded areas indicate partial regulator states which enable the value update, $\mathcal{A}_{ena}$. The underlined regulator state is the regulator state covered in the respective iteration. Dashes represent removed partial regulator states, $\mathcal{A}_{rmv}$, and double lines represent partial regulator states included in the regulation cover set $\mathcal{A}_{a_1 \to a_0}$.

in Figure 7.2 (c).

The only partial regulation state remaining in $\mathcal{A}_{ena} \setminus \mathcal{A}_{rmv}$ *is the regulator state* 110 *itself. Thus, the local cover set for* 110 *is also explicit,* $\mathcal{A}_{110} = \{110\}$. *The algorithm therefore concludes with the concrete regulation cover set* $\mathcal{A}_{a_0 \to a_1} = \{010, 100, 110\}$, *which is the optimal solution in our case.*

Let us now also consider the decreasing case $a_1 \to a_0$. *Again we depict the computation using the Hasse diagrams of the lattice* $(\Omega_a, \preceq_a)$. *All iterations of the Algorithm 1 using lexicographic order on regulator states of equal weight up to the final one are given in Figure 7.3.*

Four regulator states are symmetrical in the initial configuration, $\mathcal{W}(000) = \mathcal{W}(011) = \mathcal{W}(100) = \mathcal{W}(111) = 2.5$. *The regulator state* 000 *is therefore covered first. Unlike the case of* $a_0 \to a_1$, $\mathcal{A}_2 = \{\star 0\star, \star\star 1\}$ *is not empty. However, only* $\star 0\star$ *contains* 000, *which is not enough for a local cover set. The local cover set is therefore chosen from* $\mathcal{A}_1$, $\mathcal{A}_{000} = \{00\star, \star 00\}$. *The local cover*

set is represented by the double lines in Figure 7.3 (b). Notice that in this case, the regulator state 000 *gets covered by two partial regulator states having one more regulator value specification (a total of* 4 *specifications against the explicit* 3*).*

100 *is covered next (*$\mathcal{W}(100) = 1.5$*).* $\star 0\star$ *is no longer available and only* $10\star$ *contains* 100 *in* $\mathcal{A}_1 \setminus \mathcal{A}_{rmv}$*. The local cover set* $\mathcal{A}_{100} = \{10\star\}$ *is sufficient, however, as* $\mathcal{A}_{a_1 \to a_0}$ *already contains* $\star 00$ *which provides the missing* $d = 0$*. Thus,* 100 *is covered at an additional cost of only* 2 *regulator value specifications, effectively paying off the depth incurred while covering* 000*.*

011 *is covered next thanks to the fractional part of the weight function,* $2.5 = \mathcal{W}(011) = \mathcal{W}(111) < \mathcal{W}(001) = \mathcal{W}(101) = 2.75$*. Owing to the symmetry of the hypercube diagram,* 011 *and* 111 *are covered by the partial regulator state* $0\star 1$*,* $\star 11$ *and* $1\star 1$ *following the same reasoning (Figure 7.3 (d) and (e)).*

The remaining regulator states 001 *and* 101 *get covered by empty local cover sets,* $\mathcal{A}_{001} = \mathcal{A}_{101} = \emptyset$ *as* 001 *and* 101 *are already covered by* $00\star$ *and* $0\star 1$*, respectively* $10\star$ *and* $1\star 1$ *which are already in* $\mathcal{A}_{a_1 \to a_0}$ *(dashed lines). The algorithm therefore concludes with regulation cover set* $\mathcal{A}_{a_0 \to a_1} = \{00\star, \star 00, 10\star, 0\star 1, \star 11, 1\star 1\}$ *using* 12 *regulator value specifications as opposed to the* 18 *of the concrete regulation cover set.*

The fractional part of the weight function is crucial to distinguish between 011 *and* 001 *after the second iteration, Figure 7.3 (c)). Covering* 001 *before* 011 *would include* $\star\star 1$ *in regulation cover set* $\mathcal{A}_{a_0 \to a_1}$*. As covering* 011 *and* 111 *would still require all three partial regulator states* $0\star 1$*,* $\star 11$ *and* $1\star 1$*, the inclusion of* $\star\star 1$ *would be redundant.*

Part III

Applications

Chapter 8

Related Work

Modelling of biological systems is a typical reverse engineering application, and as usual, any further analysis is highly dependant on the quality of the model. Considering additionally the complexity of the biological phenomena studied, it is of no surprise that model inference and validation is a central topic of discrete regulatory network studies since the first applications to biological systems [44, 69].

Model inference of biological regulatory networks is traditionally conducted by hand with the aid of ad-hoc simulations to provide a trial-and-error method [46, 70]. Introduction of simulation software [26, 39] facilitated the use of systematic simulations which allows the model space to be uniformly sampled, improving on the ad-hoc method.

However, simulation based approaches work with variable reliability, as it is closely tied to the sampling density of the model space. With the model space size of biological regulatory network growing fast (asymptotically double exponentially) in the number of biological species considered (variables), it may easily become infeasible to obtain a sufficiently dense sampling due to the computation cost of the individual simulations. Thus, in order to guarantee that all relevant models are retrieved, formal methods, such as model checking, are necessary. Simulation based approaches to model inference and approaches based on combination of simulation and formal methods, however, remain highly relevant today, especially where the biological knowledge is abundant and allows for sufficient restriction of the model space.

The inference of discrete regulatory networks can be generally split into two phases. First, the influence graph is constructed, giving the topology of the network. Second, the regulation function is specified. As the regulation function can be fully specified using the parameters of a parametric regulatory network, we refer to the second phase as *parameter inference*. Our methods consider the influence graph as an input, thus we accordingly shift our focus towards parameter inference for the remainder of the chapter.

Numerous methods have been applied to parameter inference of discrete regulatory networks, ranging from comprehensive formal methods, such as model

checking, through analyses tailored to available dynamic data based on reachability or attractor analysis, all the way to constraint programming and in recent years, also machine learning [64]. In the following sections we explore in detail some of the parameter inference approaches most comparable to our work.

8.1 Model Checking

Model checking, in its various forms, is one of the most widespread methods of mathematical model verification across numerous fields. It is therefore not surprising that model checking was the first formal method introduced to discrete regulatory networks, in particular for the purpose of parameter inference [10].

The work of Bernot et al. [10] relies on model checking of discrete regulatory networks against temporal properties, given as formulae of CTL (Computational Tree Logic [18]). Many properties of interest in the biological setting can be expressed in CTL, which being a branching-time temporal logic, is well suited for properties of discrete regulatory networks with non-deterministic semantics (e.g. asynchronous semantics). The method of Bernot et al. [10], however, relies on explicitly enumerating the possible parametrisations and model checking the parametrised networks individually. The sheer amount of possible parametrisations thus imposes strict computational limits.

Several subsequent works aim to improve the scalability of model checking based approaches to parameter inference. In [48], the authors aim to improve the scalability by model checking the parametric network directly, rather than the individual parametrised networks, as well as restrict the admissible parametrisation set by influence constraints (akin to the constraints in Chapter 5). We utilise the same idea in unfolding the parametric regulatory network directly, rather than unfolding the individual parametrised networks in Chapter 6. As the different parametrised networks tend to share large portions of the expressed behaviours, avoiding repeated analysis of the shared behaviour segments benefits the methods greatly.

The ability to model check the parametric regulatory network while discriminating the inconsistent behaviour obtained by a simple union over the semantics of the individual parametrised networks, the authors utilise a novel model checking methods, called coloured model checking [7]. Although originally introduced for properties expressed in LTL (Linear Temporal Logic), coloured model checking has later been extended to also handle CTL properties [12].

In principle, coloured model checking operates similarly to the traditional temporal model checking. For each state satisfying the given property (accepting state), first, the set of all states that can reach the given accepting state is computed (reverse reachability). Second, the accepting cycles on the given accepting state are computed. Instead of simply keeping the sets of initial states and accepting cycles as in classical temporal model checking, coloured model checking annotates each state with a Boolean vector, where each bit (colour)

represents a single parametrisation. The colour vectors allow the algorithm to determine exactly which parametrisations allow a particular initial state to reach the accepting state, or which parametrisations enable the whole accepting cycle. The colour vector thus essentially translates to the parametrisation set we use to annotate states in the abstract parametric regulatory network semantics in Section 4.2.

Having the initial states and accepting cycles annotated with parametrisations, it is easy to determine which parametrisations (respectively, parametrised networks) satisfy the coveted property. The Boolean vector representation of the parametrisation set is suitable for the model checking application, as it allows for fast computation of intersections and unions, however, it relies on explicit enumeration of all parametrisations. Even if filtered by some initial conditions, such as influence constraints, the number of admissible parametrisations remains in the general case exponential, making the coloured model checking of larger networks or networks with high in-degrees in the influence graph computationally intractable.

A different approach appears in [35]. The authors aim to avoid explicit enumeration of parametrisations by omitting the classical, Kripke structure based, temporal model checking procedure. Instead, the CTL property is translated into constraints on the parameters, which exactly characterise all the parametrisations that satisfy the given property. The approach is shown to be significantly faster than traditional model checking of individual parametrised networks on a small example. The nontrivial translation of the until operator in CTL into constraints on parameters, however, introduces a new complexity limitation.

While the exact scope of model checking applicability is dependant on the expressivity of the associated logic, it is safe to assume that any model checking application to discrete regulatory networks subsumes reachability, which is easily expressible by a simple temporal formula. Indeed, to model check reachability properties only the first step of the temporal model checking is required as no accepting cycle is necessary to validate the formula. Our unfolding application being limited to reachability properties, it is natural to ask if other types of questions could be answered. While model checking using the unfolding semantics of transition system products has been studied extensively [30], the feasibility of conducting model checking on the parametric unfolding remains largely unexplored.

8.2 Reachability Analysis

Many of the common regulatory network questions can be formulated as reachability properties. Reachability can be easily expressed within temporal logic using a single temporal operator, rendering much of the model checking apparatus redundant. Several works on the regulatory networks therefore aim at improving the scalability by foregoing model checking in favour of the simpler reachability analysis.

Similar to our abstract semantics, in [61], the authors also rely on computing an over-approximation of the admissible parametrisation set. Unlike with unfolding, where the state space is explored explicitly, the authors rely on encoding the reachability problem into constraints on the parametrisation set. Although technically similar to the approach in [35], which allowed translating CTL properties into constraints, Ostrowski et al. [61] limit themselves to reachability properties, obtaining simpler constraints.

The constraints being on the parametrisations rather than the state space, the method of Ostrowski et al. [61] computes an over-approximation of the parametrisation set enabling given dynamical properties directly. By computing the over-approximation rather than the precise set of parametrisations, the authors managed to obtain simpler constraints on the parametrisation set. In turn, the constraints are solvable using efficient methods, such as answer set programming. Ostrowski et al. [61] propose using model checking on the restricted parametrisation set to filter out false positives or further analysis. The tractability of the model checking is thus improved by restriction of the input set of parametrised models. A similar approach is possible to filter out false positives within the parametrisation sets computed within our abstract semantics. Additionally, we also allow computing complete finite prefixes to represent the dynamics on the reachable state space, allowing it to be exploited during the model checking.

Corblin et al. [21] also rely on constraints to over-approximate reachability properties. The constraints for a given reachability problem are formulated directly on the dynamics, essentially describing a trace. The method of Corblin et al. [21] relies on translating the computed constraints to a Boolean formula, allowing them to capitalise on the efficient SAT implementations. The authors also tackle the problem of a minimal influence graph able to express the coveted dynamical property. This is done using influence properties akin to our observability constraint. While we do not directly support such inference, it is straightforward to obtain the influences which are not observable under a parametrisation (or a set of parametrisations) enabling the dynamical property.

Another approach is tailored for time series data, i.e. sequence of measurements over time during an experiment. Represented as sequence of (partially) observed states, time series data are common for regulatory networks.

Cummins et al. [24] use pattern matching of graphs to determine whether a model can reproduce the time series data. To achieve this, both the time series data and the model dynamics are represented as directed graphs representing the possible evolution of the variable values. Matching a path (trace) within the pattern graph of the time series data with a path in the search graph representing the regulatory network dynamics thus validates the model with respect to the coveted behaviour.

The approach in [24] relies on modelling of regulatory networks as switching systems [23]. Unlike our purely discrete representation, the switching systems describe the dynamics by means of differential equations, however, the values of the variables are interpreted in discrete fashion, based on established thresholds. Following the switching system semantics, the directed graphs

used in [24] use nodes to represent monotonic evolution of a variable, e.g. variable a is increasing, and edges to represent a variable reaching a local extrema. Discretisation based on the piecewise linearity of the variable value evolution can be more precise than classical Boolean discretisation, especially if the local maxima or minima of a single variable differ along the evolution, making it impossible to differentiate every local extrema by a single threshold. The most-permissive semantics of Boolean networks [63], discussed in Chapter 11, represent states in a similar fashion.

Of particular interest is the work of Gallet et al. [36] due to the distinct similarity with our approach. Much like the parametrisation sets we use in the abstract parametric regulatory network semantics, constraints on the parametrisation set are computed on the run in [36]. The constraints on the parametrisation space take shape of a Boolean formulae and, while exact, the formulae grow in size as new constraints are added during the computation. The parametrisation set representation using the Boolean formulae constraints can thus easily exceed our parametrisation lattice in size and complexity.

Another similarity to our work spans from the representation of the state space itself. In order to combat the combinatorial explosion of the state space, Gallet et al. [36] use symbolic execution trees to represent the reachable state space. The symbolic execution trees are similar to the unfoldings. As the name suggests, the tree structure offers an acyclic representation of the reachable state space. Coupled with the constraint based parametrisation sets, representing behaviour of multiple parametrisations collectively becomes possible. The unfolding semantics, however, additionally allow us to exploit concurrency. While the symbolic execution trees are model checking ready [36], the size of the complete finite prefix is generally significantly smaller than the symbolic execution trees, as illustrated by experimental results in Chapter 9.

Finally, a very elegant related work is the modification of Hoare logic for the gene regulatory networks [9]. Hoare logic has been introduced for proves of correctness of imperative programs. More precisely, a Hoare triple consists of a pre-condition, the program itself and post-condition. The Hoare triple is satisfied (reducible by inference rules) if running the program under the pre-condition, the program finishes and the postcondition holds. The genetically modified Hoare logic of [9] uses time series data in place of the program, allowing one to prove that under given pre-condition, the model can replicate the specified trace and the post-condition holds. Similar to the graph pattern matching approach of [24] which is also tailored for time series data, the measurement data is interpreted as a sequence of monotonic variable value evolutions rather than the standard Boolean discretisation based on thresholds.

Constructing proves of Hoare triples allows one to prove that a given dynamical property is enabled under a chosen parametrisation or set of parametrisations. The true power of the approach in [9], however, lies in the ability to compute the weakest pre-condition from the time series data and the post-condition. The weakest pre-condition is then the specification of all parametrisations which enable the coveted dynamical behaviour. The pre-condition and post-conditions are arbitrary propositional formulae on the variable and

parameter values, leading to considerable flexibility of the Hoare triple representation of the regulatory network validation against time series data. The framework therefore allows for experiment interventions, such as disabling a particular variable (knockouts), to be modelled accurately.

The weakest precondition computation of [9] is equivalent to computing the branches of complete finite prefix which correspond to the given time series data and collecting the associated parametrisation sets. Unlike our abstract parametric regulatory network semantics, however, the weakest precondition computation is precise. The parametrisation set being represented by the precondition, i.e. a propositional formula on the parameters, the size of the formula may grow significantly larger than the parametrisation lattice, especially if monotonicity influence constraints are represented explicitly.

8.3 Other Applications

In this section we introduce other works of interest, which do not directly fall in line with one of the two main approaches pinpointed for the parameter inference.

Streck et al. [67] propose a method for statistical labelling and ranking of the admissible parametrisations. Several labels, both variable and influence specific as well as spanning the entire parametrisation are proposed. Using the labels, a partial order on the parametrisations is obtained, ranking them in terms of cost, i.e. how many transitions does the associated parametrised network require to satisfy the dynamical property, robustness, i.e. what is the probability of a random trace of the parametrised network satisfying the dynamical property, or impact of a particular influence, i.e. how often does the value of a variable update to the one proposed by the sole action of the given influence, etc. The ranking is then used to refine the model in line with the best scoring parametrisations. As the parametrisations are labelled on individual basis, the method relies on explicit representation of the parametrisations. While only the admissible parametrisations have to be enumerated, such as the parametrisation lattice, the explicit enumeration still negatively impacts tractability.

A commonly used characteristic of gene regulatory networks are the attractors, i.e. sets of states from which the model cannot escape (terminal or bottom strongly connected components of the state transition graph). While closely related to reachability, attractor analysis, that is identification of the attractors, is a significantly more challenging problem. While many methods of attractor analysis have been proposed for discrete regulatory networks [2, 14, 17, 47, 3, 28, 40, 59], it is only recently that a method emerged for the parametric regulatory networks [6].

The method of Barnat et al. [6] relies on a parallelisable algorithm searching for terminal strongly connected components. To take parametrisations into effect, each reachability check is conditioned by the parametrisations that enable said reachability, effectively annotating states with admissible parametrisation

sets akin to our parametric regulatory network semantics. To avoid explicit parametrisation enumeration, binary decision diagrams, effectively equivalent to propositional formulae, are used to represent the parametrisation sets. In case monotonicity influence constraints are used, the binary decision diagram representation suffers from the same explosion in complexity as propositional formulae due to the difficulty in enumerating the monotonic Boolean functions. Unlike the parametrisation lattices, however, binary decision diagrams support unions which are necessary for the attractor analysis algorithm.

Chapter 9

Experimental Results

In this chapter we present experimental results for construction of the reachable state space using the parametric regulatory network unfolding semantics coupled with the abstraction of the parametrisation sets. The unfolding procedure and complete finite prefix construction for parametric regulatory networks as per Chapter 6 have been implemented in a prototype tool *Pawn* written in Python.[1] The experiments make use of several well-known Boolean and general multivalued parametric regulatory networks that have been studied in the literature. These results have first been published in [50].

Several regulatory network models were selected for the experiments varying in size of the network, in average connectivity of the nodes in the influence graph and in the network type (only Boolean versus general multivalued). Each experiment consists of constructing the full representation of the reachable state space as a complete finite prefix of the unfolding from a given initial state and all possible parametrisations ($\mathring{\mathcal{P}} = \mathbb{P}(G_{\boldsymbol{m}})$). All parametric regulatory networks considered are also equipped with a well-formed influence constraint set according to which each influence is considered both monotonic and observable. The number of events outside of cut-offs corresponds to the number of reachable state and parametrisation set combinations. The number of non-cut-off events therefore gives a good notion of size of the computed complete finite prefix. We construct multiple complete finite prefixes from different initial states for some of the models, where the initial state significantly impacts the size of the reachable state space. By default, however, we consider the initial states as introduced in the original model from the literature.

To illustrate the compaction achieved by the combination of unfolding semantics and parametrisation set abstraction, we compare the size of the unfoldings with the size of the complete symbolic execution tree computed from the same initial state. To construct the symbolic execution trees, we employ the tool `SPuTNIk` [36] which implements automata-based LTL model checking of parametric regulatory networks by (finite) symbolic execution of the product automaton. `SPuTNIk` explicitly traverses the product states using a depth-first

[1] Pawn is available online: `https://github.com/GeorgeKolcak/Pawn`.

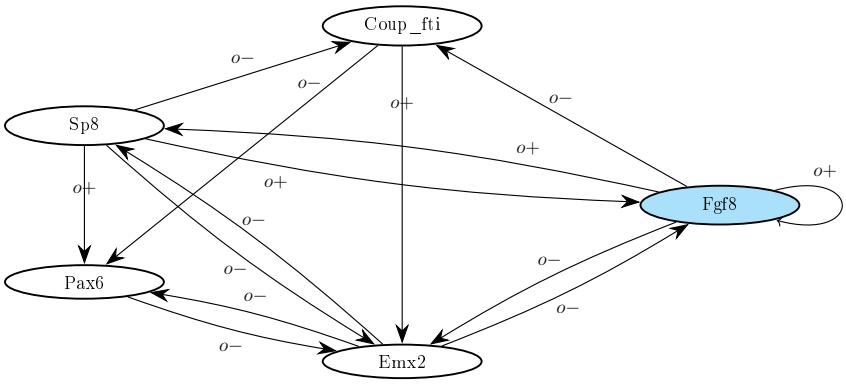


Figure 9.1: The influence graph of the parametric regulatory network modelling mammalian cortical area development. All variables are Boolean and initialised to zero in the initial state, with the exception of *Fgf8* (in blue) which has been considered with both zero and one for the initial value. The influences are labelled with the influence constraints considered in the experiments.

search approach while symbolically executing the transitions representing constraints on the parameters, such as the influence constraints or constraints based on previously executed transitions. To achieve exactly the reachable states of the state space graph of the regulatory network, we employ a Büchi automaton with a single state looping over an atomic proposition satisfied in every state of the model.

In [36], the authors consider an additional constraint on the parametrisation sets called *Min-Max*, which is also implemented in the `SPuTNIk` tool. The Min-Max constraint requires that in every state of the parametric regulatory network where all the activators (respectively inhibitors) are at their maximum values and all of the inhibitors (respectively activators) are zero at the same time, the regulation function for the variable in question must point to the maximum (respectively minimum) possible value. Such states correspond to the $\preceq_v$-maximal (respectively $\preceq_v$-minimal) regulator states of the relevant variable. As such, Min-Max constraint translates to fixing the value of the $\preceq_v$-maximal regulator state to maximum in $\mathbf{0}$ (respectively the $\preceq_v$-minimal regulator state to minimum in $\mathbf{1}$) within the initial parametrisation lattice $[\mathbf{0}, \mathbf{1}]$. To this end, we have also included the Min-Max constraint in *Pawn*.

As aforementioned, the influence constraint set of all the considered parametric regulatory networks contain both a monotonicity and an observability constraint for each influence in the network. Application of the additional Min-Max constraint is explicitly indicated.

First of the parametric regulatory networks we use is a Boolean model of the gene regulatory network underlying mammalian cortical area development [38], shown in Figure 9.1. We consider two different initial conditions, or more

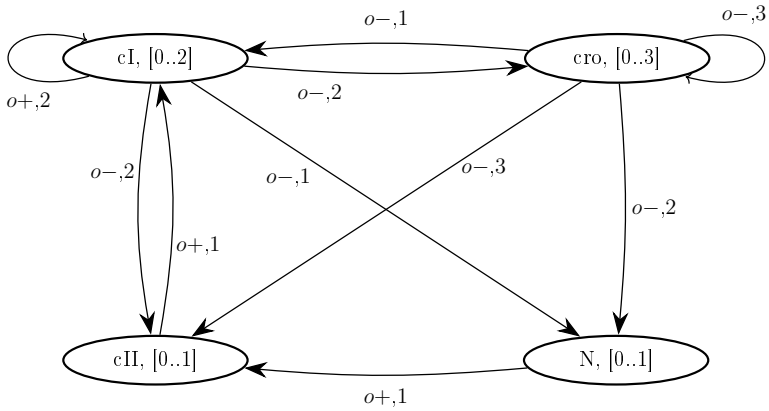


Figure 9.2: The influence graph of the parametric Thomas network of bacteriophage λ life cycle. Variable domains as given by the maximum vector $\boldsymbol{m} = (cI = 2, cII = 1, cro = 3, N = 1)$ are included in variable labels. The influences are labelled with the influence constraints considered in the experiments as well as their respective threshold values.

precisely, initial states for the unfolding. First, with all the variables initialised to zero and second, with all variable inactive (zero) with the exception of the *Fgf8*, which is initialised to one.

The smallest of the multivalued parametric regulatory networks we consider is the extensively studied regulatory network of the bacteriophage λ life cycle [68] (known colloquially as λ-switch) shown in Figure 9.2 This model has also been studied in other works aimed at analysis of parametric networks [36, 48]. We consider only one initial state for the λ-switch, which sets all variables to zero. We do, however, utilise the model in two experiments, one with and one without the Min-Max constraint.

As an example of a larger Boolean model, we consider a model of EGF-TNFα signalling pathway [54, 61] shown in Figure 9.3. In the case of this parametric regulatory network, the initial state of the unfolding sets the variables *tnfa* and *egf* to active (value one) whereas all other variables are considered inactive (value zero).

Finally, we consider a couple of larger multivalued networks (with more than 10 variables). First, we analyse a parametric regulatory network adopted from [57]. The model, illustrated in Figure 9.4, represents several key signalling pathways of Drosophila, including influences between the pathways (cross-talks).

Second, we analyse a model describing the control of the developmental process in primary sex determination of placental mammals [65]. While slightly smaller in terms of number of variables, 14, than the Drosophila network, the primary sex determination model, depicted in Figure 9.5, is highly intercon-

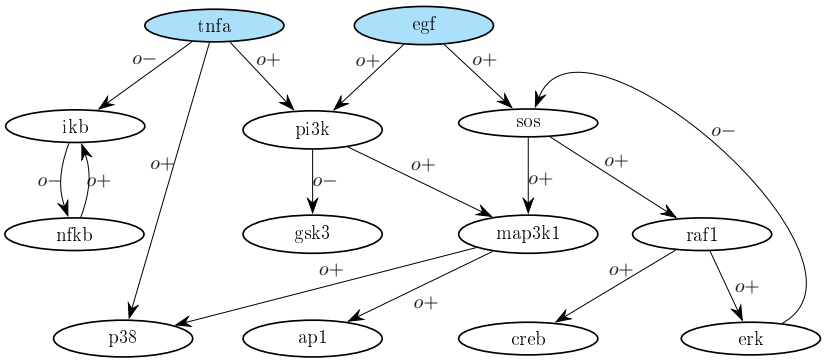


Figure 9.3: The influence graph of the parametric Boolean network of EGF-TNFα signalling pathway. The variables shown in blue are set to value 1 in the initial state. The influences are labelled with the influence constraints considered in the experiments.

nected, leading to more possible parametrisations.

Computations conducted on all the defined models have led to results shown in Table 9.1. Complete finite prefixes of the unfoldings constructed by *Pawn* are characterised by their size given by both total number of events and events without cut-offs. A relatively large portion of cut-off events indicates the large number of different behaviours spread among the different parametrisations. The number of symbolically executed states computed by `SPuTNIk` is given for comparison.

Since both tools are implemented as prototypes without any optimisations, we do not include computation times but rather focus on size of the reachable state space representation. However, in all models with the only exception of the Primary Sex Determination model, the computation by *Pawn* concluded within a couple of minutes. In case of the Primary Sex Determination model, *Pawn* constructed the complete finite prefix in 2 hours whereas `SPuTNIk` has been stopped in 3 days without achieving results. In case of the Drosophila model, `SPuTNIk` has been stopped after 2 days of computations whereas *Pawn* needed a couple of minutes to compute the complete finite prefix. `SPuTNIk` reached a symbolic execution tree of size at least $7,000,000$ before being timed out in all three relevant cases. As timing was not a concern, all experiments have been conducted on a standard laptop computer.

Using the concurrency aware, partial order semantics shows a great improvement in the compactness of the resulting structure. It is striking in the case of models of signalling pathway cross-talks (Drosophila and EGF-TNFα) where the amount of concurrency among the variables is high due to the sparsity of the influence graph. The size of unfolding prefixes remains very compact even in cases with more interwoven topology. It is worth noting that the constructed

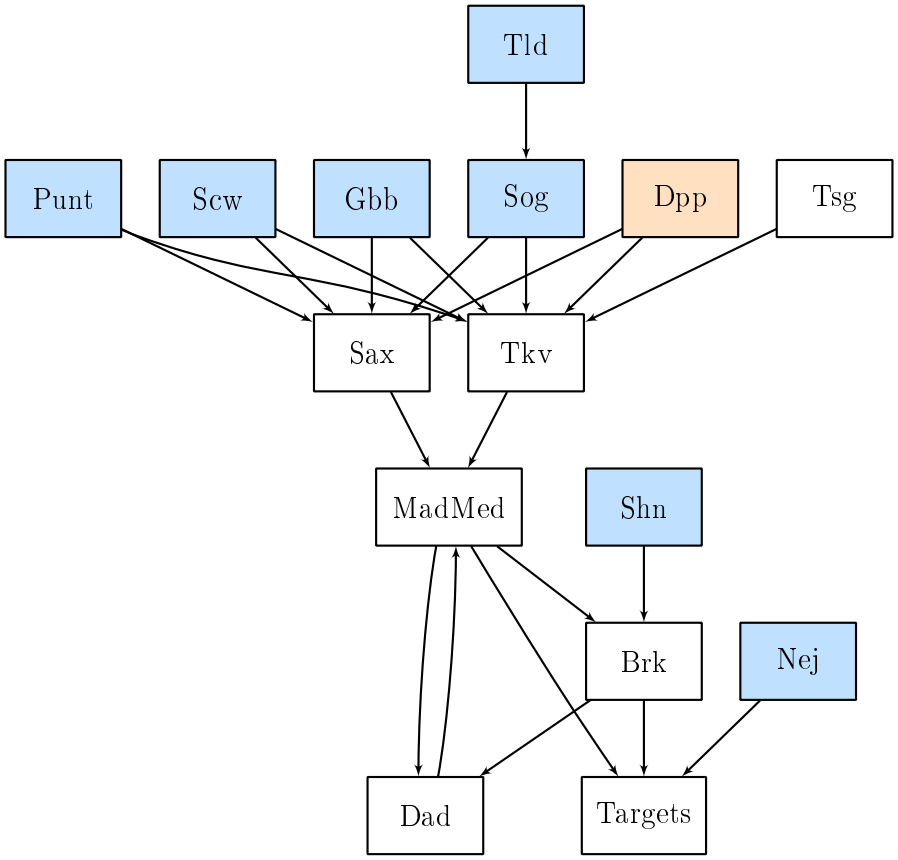


Figure 9.4: The influence graph of the parametric multivalued network of signalling pathways of Drosophila. All variables with non-zero values in the initial state are initialised with their maximum values, 1 for the variables shown in blue and 2 for the variables shown in orange.

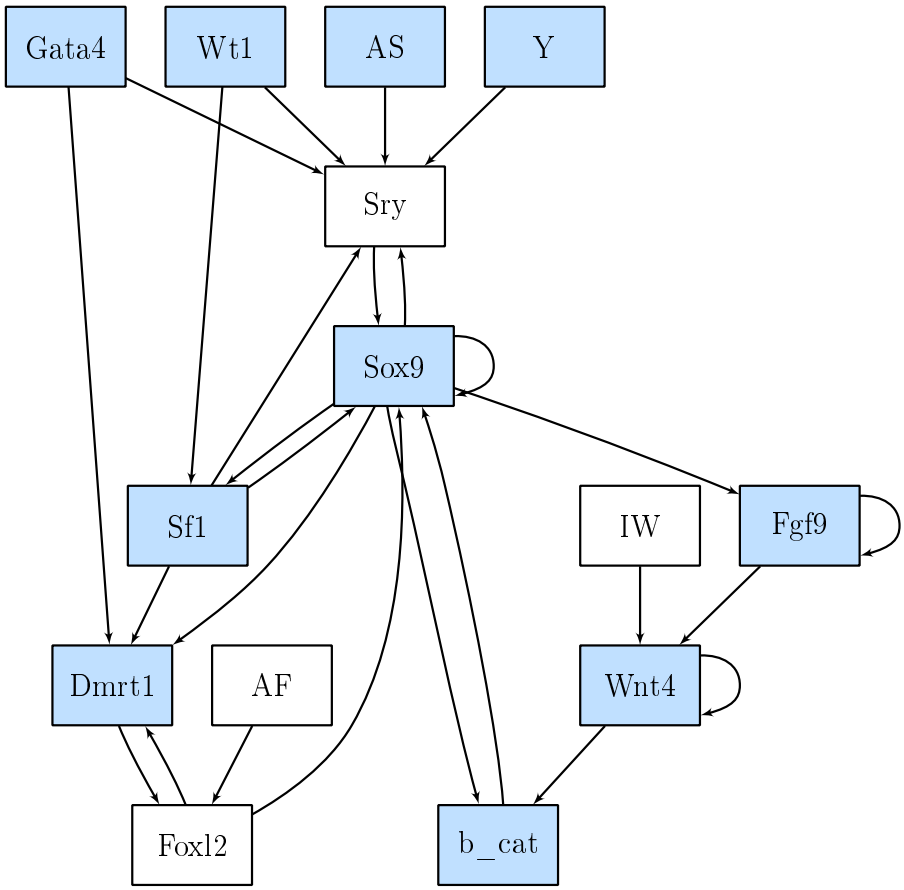


Figure 9.5: The influence graph of the parametric multivalued network of primary sex determination of placental mammals. The variables with value 1 in the initial state are shown in blue.

Model (init. state)	Type	# nodes	# events (incl. cut-offs)	Sym. exec. size
Cortical Dev. (Fgf8=0)	BN	5	554 (1,939)	8,312
Cortical Dev. (Fgf8=1)	BN	5	1,054 (3,530)	8,312
EGF-TNFα	BN	13	1,057 (2,658)	534,498
λ-switch	MN	4	170 (575)	68,011
λ-switch w/ Min-Max	MN	4	157 (527)	15,139
Prim. Sex Det. w/ Min-Max	MN	14	19,954 (88,994)	>7,000,000
Drosophila Signalling	MN	15	781 (2,698)	>7,000,000
Drosophila w/ Min-Max	MN	15	731 (2,507)	>7,000,000

Table 9.1: Comparison of the size of the obtained structures between complete finite prefixes of the unfolding and the symbolic representation for different models. The number of unfolding events is specified as a total number of non-cut-off events. The number including cut-off events is given in brackets. Symbolic representation size is the number of states of the complete execution tree constructed by `SPuTNIk`. The notation '>7,000,000' refers to the size being over $7,000,000$ by the time the particular experiment has been stopped after 2 or more days of computation.

complete finite prefixes preserve the set of reachable states and any process can be reconstructed from the prefix with an additional computation cost [32].

Another interesting observation can be made on the model of cortical development in mammals, showing that the unfolding and consequently the complete finite prefix is sensitive to the initial state. In this model, the considered initial states give the same reachable state space. However, depending on the initial state, the respective unfoldings have substantially different size. This can be attributed especially to the dependence on the parametrisation sets, and their incompatibility with the total adequate order [31], which may result in higher fragmentation of the parametrisation sets in some cases.

Theorem 5.1 ensures that the set of reachable states in the complete finite prefix is exact despite the over-approximation of the parametrisation sets. I.e. for each reachable state there exists at least one parametrisation which is a true positive within the computed parametrisation set. The false positives can be identified by running model checking or other exact algorithm on the over-approximated parametrisation set, thus obtaining a much smaller and much more manageable set of initial parametrisations.

Part IV

Discussion

Chapter 10

Summary

This thesis explores in detail the parametric model of regulatory networks and associated algorithms. The parametric regulatory network analysis is largely hampered by combinatorial explosion in the number of states, same as discrete regulatory networks, as well as the number valid combinations of parameter value assignments called parametrisations. We tackle those challenges by specialised semantics, which allow exploration of the parametric regulatory network behaviour without explicit enumeration of states or parametrisations.

To avoid explicit enumeration of parametrisations, we introduce an abstraction of parametrisation sets by the means of their convex cover. We show that this abstraction is exact for the parametric regulatory networks without influence constraint and it leads to a sound and minimal over-approximation if influence constraints are considered.

Rather than abstraction, we evade explicit state enumeration by the means of partial order reduction. We elevate the unfolding semantics from Petri nets to parametric regulatory networks, thus being able to capitalise on the concurrency abundant in biological systems.

Both of the introduced approaches are expressed as different semantics of parametric regulatory networks. Thanks to being orthogonal not only in their purpose, but also in their design, the two semantics can be naturally combined, allowing us to alleviate both sources of combinatorial explosion at once. The resulting combined semantics have been implemented for the purposes of reachable state space exploration and experimental results show the resulting representation of the reachable state space is significantly smaller compared to other approaches.

We further investigate the possibility of using known target state to optimise the state space exploration of parametric regulatory networks. To this end, we elevate a model reduction method based on pruning transitions which cannot lead to a target state from automata networks to parametric regulatory networks.

The thesis offers a thorough analysis of parametrisation set abstraction and unfolding semantics for parametric regulatory networks. Many questions

related to parametric regulatory networks remain unanswered, however, opening up multiple possibilities for future development, not only in the area of parametric regulatory networks themselves, but also in exploration of related concepts where parametric regulatory network might help in providing the necessary insight.

Chapter 11

Ongoing and Future Work

In this chapter we explore ongoing and future work building up on the analysis of the parametric regulatory networks as well as related concepts. The work conducted and potential future work contains several refinements and extension of the abstract and unfolding semantics of the parametric regulatory networks, new application areas beyond reachability, as well as questions of expressivity and monotonicity of continuous models of regulatory networks and their discrete counterpart.

The first potential area for future work is the abstraction of parametrisation space. The parametrisation set abstraction as we introduce it uses very basic algebraic structures. While this ensures many nice properties for the abstraction, including a small static size, it also makes the abstraction considerably rigid. This opens up potential for introduction of a more complex as well as more permissible structure which could help reduce the over-approximation, optimise the restriction to influence constraints or even tackle fundamental limitations of the convex sublattice approach which renders efficient unions impossible (unions are crucial for some applications, see Section 11.1).

On a related note, the abstract semantics of the parametric regulatory networks have the potential to be amended to account for more constraints than the monotonicity and observability influence constraints we consider. This potential is illustrated by the adaptation of the Min-Max constraint considered in [36] in Chapter 9. While the Min-Max constraint is relatively simple, more complex constraints could be considered on both the influences or the parameters directly.

Refinement of the unfolding semantics also opens up many interesting avenues for future work. Due to the nature of the asynchronous multivalued network semantics as we define them, only one variable changes values with each transition. However, the transition being enabled may be dependent on the values of numerous other variables. As a result, each event in the parametric regulatory network unfolding has to consider such a variable among its preconditions only to include a postcondition labelled by the same variable and value. Instead, this dependency could be represented by an equivalent of

a *read arc* used in contextual Petri nets. Petri net unfolding semantics have been extended to contextual Petri nets relying on asymmetric conflict relation, which allows a condition to be read without producing a copy [5]. This not only contributes to smaller prefixes overall but also to overall improvement of the running time. As the parametric regulatory network unfoldings are very similar to Petri net unfoldings there is good reasoning to believe that the techniques used in [5] could be adapted to parametric regulatory network unfolding almost effortlessly and should certainly be considered for implementation of an efficient tool for parametric regulatory network unfolding to replace the prototype tool *Pawn*.

Another potential for improvement of the unfolding semantics lies in the adequate order. Designing a total adequate order that could minimise the required amount of backwards cut-offs would be greatly beneficial for reducing the runtime of the unfolding algorithm. While finding a suitable adequate order is a highly nontrivial problem, there is a potential for obtaining insights into other unfolding applications that rely on annotated events and suffer from a similar difficulty during complete finite prefix construction.

Another significant potential for expansion of the presented results is the exploration of further application areas. In our work we have focused mostly on reachability problems. As discussed in Chapter 8, however, other applications are highly relevant for the study of biological regulatory networks.

One such application is model checking. Our results are already highly compatible with model checking approaches for discrete regulatory networks. In particular, by first utilising our abstract semantics to obtain a restricted set of parametrisations, one could greatly reduce the number of models for which model checking is necessary. Running model checking directly on the parametric regulatory networks, however, is a far more attractive application. While parametric regulatory networks with abstract semantics are in essence model checking ready and model checking algorithms have also been proposed for unfoldings [30], porting of model checking algorithms to parametric regulatory network unfoldings remains nontrivial and will likely result in the need to compute unions of abstract parametrisation sets.

An alternative to full scale model checking may be efficient algorithms for another common problem on regulatory networks beside reachability, such as attractor analysis briefly discussed in Section 8.3 of Chapter 8. As the topic of attractor analysis subsumes several nontrivial approaches, we explore it in the detail it warrants in Section 11.1.

Very interesting work has also been done on models that might be employed to model biological regulatory networks from a perspective similar to parametric regulatory networks. In particular, promising results have been obtained for new symbolic semantics of Boolean networks called most-permissive semantics [63]. While the most permissive semantics are fundamentally unrelated to parametric regulatory networks, both are essentially abstractions of Boolean networks or more broadly discrete regulatory networks. We compare the two approaches in Section 11.2.

Finally, work on relational properties of monotonic continuous systems [49]

shows promise for a further development of the notion of monotonicity across the spectrum of biological regulation models using different levels of abstraction, spanning from continuous and hybrid systems all the way to Boolean networks.

11.1 Attractor Analysis

An attractor of a discrete regulatory network is a set of states corresponding to a bottom strongly connected component in the state transition graph. In other words, for any state in the attractor it is possible to reach any other state in the same attractor, but no other state outside the attractor. Attractors of discrete regulatory networks therefore represent the long-term, stable behaviours of the system. The study of such behaviours is highly relevant in many high profile areas such as cell differentiation, oncology and synthetic biology [42, 43, 55].

While attractor analysis approaches based on the unfolding semantics have been successful [14], extension of such approaches to parametric regulatory networks is highly nontrivial. The technique of Chatain et al. [14] relies on identifying candidate markings as the markings of maximal configurations and then checking each candidate marking by constructing another unfolding with the candidate marking as initial marking. If a different candidate marking is discovered during the follow-up unfolding, the initial marking is removed from the set of candidate markings because it either does not belong to an attractor, in case it is not reachable from the other candidate marking, or the same attractor will be discovered when unfolding from the other candidate marking, owing to attractors being strongly connected.

Adapting the method of [14] to unfoldings of parametric regulatory networks, however, faces fundamental challenges. Parametric regulatory network unfolding may contain several instances of the same candidate marking with different parametrisation sets. As unions of abstract parametrisation sets cannot be efficiently represented in the general case, each instance of a candidate marking has to be unfolded separately, significantly increasing the number of unfoldings that have to be computed. Moreover, the unfoldings from the candidate markings tend to lead to larger complete finite prefixes due to candidate markings being disqualified per parametrisation. With the two above difficulties combined, the resulting algorithm has been found intractable for practical application. While an optimisation of the method might be envisioned to obtain practical algorithms for parametric regulatory networks, it is unlikely to be possible without efficient computation of abstract parametrisation set unions.

A promising future work on the attractor analysis of parametric regulatory networks might instead built upon the results of [6]. The method of Barnat et al. [6] relies on binary decision diagrams for encoding the admissible parametrisations. Computing unions for binary decision diagrams is simple, however, the size of the diagram grows with the number of variables. Indeed, the size of the binary decision diagram is exponential in the number of variables of a

monotonic Boolean function. This is a challenge which does not apply to the abstract parametrisation sets, whose encoding is of constant size.

A clever combination of the two approaches thus promises a fruitful collaboration. A modification of the binary decision diagrams to accommodate the bounded convex sublattices of parametrisations could thus help keep the size of the binary decisions diagrams manageable even for variables with numerous regulators. On the other hand, splitting the parametrisation lattices into a well designed decision diagram structure might allow for efficient unions. The application of the successful combination of the two formalisms might even extend beyond attractor analysis.

11.2 Most Permissive Semantics

The most permissive semantics of Boolean networks are symbolic semantics, assigning two transitional values $\searrow, \nearrow$ to variables on top of the two Boolean values. In simple terms, the transitional values represent a variable increasing value (tending towards the maximum), respectively decreasing value (tending towards the minimum). More precisely, any variable in a transitional value may collapse to the respective Boolean value, i.e. 0 for $\searrow$ and 1 for $\nearrow$ at any time, while any variable in the role of a regulator with transitional value may be read as either 0 or 1, irrespective of the direction. Thus, for the purposes of regulation, a variable has to either be in the Boolean value prescribed by the regulation function or in any of the transitional values $\searrow, \nearrow$.

Boolean networks with the most-permissive semantics exhibit more behaviours than the standard semantics we introduced in Chapter 2. While the increase in expressivity may appear to be far too liberal, most permissive semantics have been shown to successfully discriminate behaviours [63]. On the other hand, Boolean networks with most permissive semantics can reproduce any behaviour generated with generalised asynchronous semantics, or even any behaviour generated by a multivalued or continuous refinement[1] of the Boolean network. Moreover, analysis of Boolean networks with most permissive semantics is computationally cheaper. Reachability properties in the most permissive semantics, for instance, can be translated to SAT problems, facilitating the use of some of the fastest NP algorithms.

Albeit fundamentally different, the most permissive semantics of Boolean networks as well as parametric regulatory networks serve to lessen the mathematical rigidity of discrete regulatory networks with the standard semantics, thus reducing the number of parameters required for modelling. In spite of helping to achieve the same result, the approaches are different not only in nature but also by interpretation. Whereas parametric regulatory networks abandon the standard regulation function, but by the means of parametrisations retain full specification of the emergent behaviour, the most permissive semantics are applied to Boolean networks specified by the regulation function, but abstract away the exact specification of the resulting behaviour.

[1] Please refer to [63] for details.

The duality between most permissive semantics and parametric regulatory networks opens up several questions of interest. One such a question is naturally the possibility to combine the two approaches while preserving their respective strengths. While it might be possible to directly elevate the most permissive semantics to Boolean parametric regulatory networks, the use of symbolic states makes it unlikely that parametrisation sets could be efficiently restricted. A different point of view, a translation between traces of a Boolean network with most permissive semantics and parametrisation sets of a corresponding parametric regulatory network and vice versa, shows promise for the purposes of parameter inference and model refinement.

Study of the connection between traces or behaviours and parameters is of significance even beyond a precise translation. Linking behavioural patterns observable in Boolean networks with most permissive semantics to parameter values and their relations has the potential to uncover new meaningful constraints for parametric regulatory networks that could help restrict or otherwise shape the admissible parametrisation sets.

The development of most permissive semantics also stands to benefit from establishing relations to parametrisation. In particular, in regards to the local monotonicity of the regulation function, which is often at least partially known in the literature. The monotonicity constraints on influences form the cornerstone of parametrisation set restriction, the most permissive semantics of Boolean networks however, lack any means to discriminate between monotonic and nonmonotonic behaviour. This is underlined by the capability of a variable with increasing value, $\nearrow$ to be first used as if valued 1 and subsequently as if valued 0 for regulation of another variable, thus violating the intuition of increasing its value. A preliminary study of the monotonicity under the most permissive semantics shows that while enforcing monotonic behaviour is possible, the usual notion of monotonicity in multivalued and continuous models does not straightforwardly translate into the Boolean abstraction with transitional values $\searrow$, $\nearrow$. Classifying the behaviours generated by Boolean networks with most permissive semantics based on local monotonicity inferred from parametric representation might thus lead to valuable insights into monotonicity under most permissive semantics as such as well as relations of monotonicity across various refinements of Boolean networks.

Bibliography

[1] W. Abou-Jaoudé, P. T. Monteiro, A. Naldi, M. Grandclaudon, V. Soumelis, C. Chaouiya, and D. Thieffry. Model checking to assess t-helper cell plasticity. *Frontiers in Bioengineering and Biotechnology*, 2:86, 2015.

[2] W. Abou-Jaoudé, P. Traynard, P. T. Monteiro, J. Saez-Rodriguez, T. Helikar, D. Thieffry, and C. Chaouiya. Logical modeling and dynamical analysis of cellular networks. *Frontiers in Genetics*, 7:94, 2016.

[3] T. Akutsu, M. Hayashida, and T. Tamura. Integer programming-based methods for attractor detection and control of boolean networks. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 5610–5617, 2009.

[4] R. Albert 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(1):1 – 18, 2003.

[5] P. Baldan, A. Bruni, A. Corradini, B. König, C. Rodríguez, and S. Schwoon. Efficient unfolding of contextual petri nets. *Theoretical Computer Science*, 449:2 – 22, 2012. Descriptional Complexity of Formal Systems (DCFS 2011).

[6] J. Barnat, N. Beneš, L. Brim, M. Demko, M. Hajnal, S. Pastva, and D. Šafránek. Detecting attractors in biological models with uncertain parameters. In J. Feret and H. Koeppl, editors, *Computational Methods in Systems Biology*, pages 40–56, Cham, 2017. Springer International Publishing.

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

[8] G. Bernot, F. Cassez, J.-P. Comet, F. Delaplace, C. Müller, and O. Roux. Semantics of biological regulatory networks. *Electronic Notes in Theoretical Computer Science*, 180(3):3 – 14, 2007.

[9] G. Bernot, J.-P. Comet, Z. Khalis, A. Richard, and O. Roux. A genetically modified hoare logic. *Theoretical Computer Science*, 765:145 – 157,

2019. Formal Verification and Static Analysis of Molecular Devices and Biological Systems.

[10] G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339 – 347, 2004.

[11] G. Birkhoff. *Lattice Theory*. Number vb. 25,del 2 in American Mathematical Society colloquium publications. American Mathematical Society, 1940.

[12] L. Brim, M. Češka, M. Demko, S. Pastva, and D. Šafránek. Parameter synthesis by parallel coloured CTL model checking. In O. Roux and J. Bourdon, editors, *Computational Methods in Systems Biology*, volume 9308 of *Lecture Notes in Computer Science*, pages 251–263. Springer International Publishing, 2015.

[13] D. M. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems (Performance, Reliability, Digital)*. PhD thesis, Stanford, CA, USA, 1985. AAI8506166.

[14] T. Chatain, S. Haar, L. Jezequel, L. Paulevé, and S. Schwoon. Characterization of reachable attractors using Petri net unfoldings. In P. Mendes, J. Dada, and K. Smallbone, editors, *Computational Methods in Systems Biology*, volume 8859 of *Lecture Notes in Computer Science*, pages 129–142. Springer Berlin Heidelberg, Cham, 2014.

[15] T. Chatain, S. Haar, J. Kolčák, L. Paulevé, and A. Thakkar. Concurrency in boolean networks. *Natural Computing*, 19(1):91–109, Mar 2020.

[16] T. Chatain and L. Paulevé. Goal-Driven Unfolding of Petri Nets. In R. Meyer and U. Nestmann, editors, *28th International Conference on Concurrency Theory (CONCUR 2017)*, volume 85 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[17] S.-M. Choo and K.-H. Cho. An efficient algorithm for identifying primary phenotype attractors of a large-scale boolean network. *BMC Systems Biology*, 10(1):95, Oct 2016.

[18] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, pages 52–71, London, UK, 1981. Springer-Verlag.

[19] D. P. A. Cohen, L. Martignetti, S. Robine, E. Barillot, A. Zinovyev, and L. Calzone. Mathematical modelling of molecular pathways enabling tumour cell invasion and migration. *PLoS Comput. Biol.*, 11(11):e1004571, 2015.

[20] S. Collombet, C. van Oevelen, J. L. Sardina Ortega, W. Abou-Jaoudé, B. Di Stefano, M. Thomas-Chollier, T. Graf, and D. Thieffry. Logical modeling of lymphoid and myeloid cell specification and transdifferentiation. *Proc. Natl. Acad. Sci.*, 114(23):5792–5799, 2017.

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

[22] P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, 1992.

[23] B. Cummins, T. Gedeon, S. Harker, and K. Mischaikow. Database of dynamic signatures generated by regulatory networks (dsgrn). In J. Feret and H. Koeppl, editors, *Computational Methods in Systems Biology - 15th International Conference, CMSB 2017, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 300–308, Germany, jan 2017. Springer Verlag. 15th International Conference on Computational Methods in Systems Biology, CMSB 2017 ; Conference date: 27-09-2017 Through 29-09-2017.

[24] B. Cummins, T. Gedeon, S. Harker, and K. Mischaikow. Model rejection and parameter reduction via time series. *SIAM Journal on Applied Dynamical Systems*, 17(2):1589–1616, 2018.

[25] H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[26] H. de Jong, J. Geiselmann, C. Hernandez, and M. Page. Genetic Network Analyzer: qualitative simulation of genetic regulatory networks. *Bioinformatics*, 19(3):336–344, 02 2003.

[27] J. Demongeot, J. Aracena, F. Thuderoz, T.-P. Baum, and O. Cohen. Genetic regulation networks: circuits, regulons and attractors. *Comptes Rendus Biologies*, 326(2):171 – 188, 2003.

[28] V. Devloo, P. Hansen, and M. Labbé. Identification of all steady states in large networks by logical analysis. *Bulletin of Mathematical Biology*, 65(6):1025–1051, Nov 2003.

[29] J. Engelfriet. Branching processes of petri nets. *Acta Inf.*, 28(6):575–591, 1991.

[30] J. Esparza and K. Heljanko. *Unfoldings – A Partial-Order Approach to Model Checking*. EATCS Monographs in Theoretical Computer Science. Springer-Verlag, March 2008.

[31] J. Esparza, S. Römer, and W. Vogler. An Improvement of McMillan's Unfolding Algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.

[32] J. Esparza and C. Schröter. Unfolding based algorithms for the reachability problem. *Fundam. Inf.*, 47(3-4):231–245, 2001.

[33] J. Fisher, T. A. Henzinger, M. Mateescu, and N. Piterman. Bounded asynchrony: Concurrency for modeling cell-cell interactions. In J. Fisher, editor, *Formal Methods in Systems Biology*, pages 17–32, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[34] F. Fogelman-Soulie, M. Milgram, and G. Weisbuch. Automata networks as models for biological systems: (a survey). In M. Cosnard, J. Demongeot, and A. Le Breton, editors, *Rhythms in Biology and Other Fields of Application*, pages 144–172, Berlin, Heidelberg, 1983. Springer Berlin Heidelberg.

[35] J. Fromentin, 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*, pages 4595–4598, Aug 2007.

[36] E. Gallet, 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*, chapter An LTL Model Checking Approach for Biological Parameter Inference, pages 155–170. Springer International Publishing, Cham, 2014.

[37] A. Garg, L. Mendoza, I. Xenarios, and G. DeMicheli. Modeling of multiple valued gene regulatory networks. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1398–1404, 2007.

[38] C. E. Giacomantonio and G. J. Goodhill. A boolean model of the gene regulatory network underlying mammalian cortical area development. *PLOS Computational Biology*, 6(9):1–13, 2010.

[39] A. G. Gonzalez, A. Naldi, L. Sánchez, D. Thieffry, and C. Chaouiya. Ginsim: A software suite for the qualitative modelling, simulation and analysis of regulatory networks. *Biosystems*, 84(2):91 – 100, 2006. Dynamical Modeling of Biological Regulatory Networks.

[40] W. Guo, G. Yang, W. Wu, L. He, and M. Sun. A parallel attractor finding algorithm based on boolean satisfiability for genetic regulatory networks. *PLOS ONE*, 9(4):1–10, 04 2014.

[41] S. Haar, J. Kolčák, and L. Paulevé. Combining refinement of parametric models with goal-oriented reduction of dynamics. In C. Enea and R. Piskac, editors, *Verification, Model Checking, and Abstract Interpretation*, pages 555–576, Cham, 2019. Springer International Publishing.

[42] S. Huang, G. Eichler, Y. Bar-Yam, and D. E. Ingber. Cell fates as high-dimensional attractor states of a complex gene regulatory network. *Phys. Rev. Lett.*, 94:128701, Apr 2005.

[43] S. Huang, I. Ernberg, and S. Kauffman. Cancer attractors: A systems view of tumors from a gene network dynamics and developmental perspective. *Seminars in Cell & Developmental Biology*, 20(7):869 – 876, 2009. Structure and function of the Golgi apparatus and Systems Approaches to Cell and Developmental Biology.

[44] S. Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224(5215):177–178, 1969.

[45] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. Random Boolean network models and the yeast transcriptional network. *Proceedings of the National Academy of Sciences*, 100(25):14796–14799, 2003.

[46] M. Kaufman, J. Urbain, and R. Thomas. Towards a logical analysis of the immune response. *Journal of Theoretical Biology*, 114(4):527 – 561, 1985.

[47] H. Klarner, A. Bockmayr, and H. Siebert. Computing maximal and minimal trap spaces of boolean networks. *Natural Computing*, 14(4):535–544, Dec 2015.

[48] H. Klarner, A. Streck, D. Šafránek, J. Kolčák, and H. Siebert. Parameter identification and model ranking of thomas networks. In D. Gilbert and M. Heiner, editors, *Computational Methods in Systems Biology*, pages 207–226, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[49] J. Kolčák, J. Dubut, I. Hasuo, S.-y. Katsumata, D. Sprunger, and A. Yamada. Relational differential dynamic logic. In A. Biere and D. Parker, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 191–208, Cham, 2020. Springer International Publishing.

[50] J. Kolcák, D. Šafránek, S. Haar, and L. Paulevé. Parameter Space Abstraction and Unfolding Semantics of Discrete Regulatory Networks. *Theoretical Computer Science*, 765:120–144, 2019.

[51] R. Laubenbacher and B. Stigler. A computational algebra approach to the reverse engineering of gene regulatory networks. *Journal of Theoretical Biology*, 229(4):523 – 537, 2004.

[52] N. Le Novère. Quantitative and logic modelling of molecular and gene networks. *Nature reviews. Genetics*, 16:146–158, 2015.

[53] S. MacLane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer-Verlag New York, 1978.

[54] A. MacNamara, C. Terfve, D. Henriques, B. P. Bernabé, and J. Saez-Rodriguez. State–time spectrum of signal transduction logic models. *Physical Biology*, 9(4):045003, 2012.

[55] H. Mandon, C. Su, S. Haar, J. Pang, and L. Paulevé. Sequential reprogramming of boolean networks made practical. In L. Bortolussi and G. Sanguinetti, editors, *Computational Methods in Systems Biology*, pages 3–19, Cham, 2019. Springer International Publishing.

[56] A. Mazurkiewicz. Concurrent program schemes and their interpretations. *DAIMI Report Series*, 6(78), Jul. 1977.

[57] A. Mbodj, G. Junion, C. Brun, E. E. M. Furlong, and D. Thieffry. Logical modelling of drosophila signalling pathways. *Mol. BioSyst.*, 9:2248–2258, 2013.

[58] K. L. McMillan and D. K. Probst. A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1):45–65, Jan 1995.

[59] M. Mushthofa, S. Schockaert, and M. De Cock. Computing attractors of multi-valued gene regulatory networks using fuzzy answer set programming. In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1955–1962, 2016.

[60] M. Noual and S. Sené. Synchronism versus asynchronism in monotonic boolean automata networks. *Natural Computing*, 01 2017.

[61] M. Ostrowski, L. Paulevé, T. Schaub, A. Siegel, and C. Guziolowski. Boolean network identification from perturbation time series data combining dynamics abstraction and logic programming. *Biosystems*, 149:139 – 153, 2016.

[62] L. Paulevé. Goal-Oriented Reduction of Automata Networks. In *CMSB 2016 - 14th conference on Computational Methods for Systems Biology*, volume 9859 of *Lecture Notes in Bioinformatics*. Springer, 2016.

[63] L. Paulevé, J. Kolčák, T. Chatain, and S. Haar. Reconciling qualitative, abstract, and scalable modeling of biological networks. *Nature Communications*, 11(1), 2020.

[64] Z. Razaghi-Moghadam and Z. Nikoloski. Supervised learning of gene-regulatory networks based on graph distance profiles of transcriptomics data. *npj Systems Biology and Applications*, 6(1):21, Jun 2020.

[65] L. Sánchez and C. Chaouiya. Primary sex determination of placental mammals: a modelling study uncovers dynamical developmental constraints in the formation of sertoli and granulosa cells. *BMC systems biology*, 10:37, 2016.

[66] T. Stephen and T. Yusun. Counting inequivalent monotone boolean functions. *Discrete Applied Mathematics*, 167:15 – 24, 2014.

[67] A. Streck, K. Thobe, and H. Siebert. Comparative statistical analysis of qualitative parametrization sets. In A. Abate and D. Šafránek, editors, *Hybrid Systems Biology*, pages 20–34, Cham, 2015. Springer International Publishing.

[68] D. Thieffry and R. Thomas. Dynamical behaviour of biological regulatory networks—ii. immunity control in bacteriophage lambda. *Bulletin of Mathematical Biology*, 57:277–297, 1995.

[69] R. Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42(3):563 – 585, 1973.

[70] R. Thomas and M. Kaufman. Multistationarity, the basis of cell differentiation and memory. i. structural conditions of multistationarity and other nontrivial behavior. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 11(1):170–179, 2001.

[71] P. Traynard, A. Fauré, F. Fages, and D. Thieffry. Logical model specification aided by model-checking techniques: application to the mammalian cell cycle regulation. *Bioinformatics*, 32(17):i772–i780, 2016.

[72] R.-S. Wang, A. Saadatpour, and R. Albert. Boolean modeling in systems biology: an overview of methodology and applications. *Physical Biology*, 9(5):055001, 2012.

[73] N. Weinstein and L. Mendoza. A network model for the specification of vulval precursor cells and cell fusion control in caenorhabditis elegans. *Frontiers in Genetics*, 4:112, 2013.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Dépliages et interprétation abstraite pour réseaux de régulation biologiques paramétrés

Mots clés: Concurrence, Biologie des systèmes, Interprétation abstraite

Résumé: L'analyse de la dynamique des réseaux de régulation biologique est confrontée à l'incertitude du modèle informatique exact. Les connaissances disponibles concernent principalement l'existence d'interactions entre espèces biologiques. Les détails sur la façon dont les différents régulateurs coopèrent, et encore plus sur les taux pour ces interactions, sont cependant rarement disponibles. Les réseaux de régulation discrets offrent ainsi une abstraction appropriée car ils nécessitent peu de paramètres par rapport aux modèles quantitatifs. Néanmoins, la détermination des paramètres discrets est un défi bien connu.

L'ensemble des affectations de valeurs de paramètres admissibles (paramétrisations) est représenté par des réseaux de régulation paramétriques. L'analyse de la dynamique des réseaux de régulation paramétriques est cependant entravée par la double explosion combinatoire, de l'espace d'état et de l'espace de paramétrisation. Nous développons des méthodes visant à atténuer l'explosion combinatoire. Premièrement, nous introduisons une interprétation abstraite pour l'ensemble des paramétrisations admissibles, en obtenant un codage de taille constante, au prix d'une surapproximation conservatrice. Deuxièmement, nous soulevons la sémantique des ordres partiels sous la forme d'un déploiement des réseaux de Petri aux réseaux de régulation paramétriques, en exploitant la concurrence pour une représentation efficace de l'espace d'état.

Title: Unfoldings and Abstract Interpretation for Parametric Biological Regulatory Networks

Keywords: Concurrency, Systems Biology, Abstract Interpretation

Abstract: The analysis of dynamics of biological regulatory networks faces the uncertainty of the exact computational model. The available knowledge concerns predominantly the existence of interactions between biological species. The details on how different regulators cooperate, and even more so on rates for those interactions, however, are rarely available. Discrete regulatory networks thus offer an approprite abtraction as they require few parameters compared to quantitative models. Nevertheless, determining the discrete parameters is a well known challenge.

The set of admissible parameter value assignements (parametrisations) is represented by parametric regulatory networks. The analysis of parametric regulatory network dynamics is, however, hampered by dual combinatorial explosion, of the state space and of the parametrisation space. We develop methods aimed at alleviating the combinatorial explosion. First, we introduce abstract interpretation for the set of admissible parametrisations, achieving constant size encoding, at the cost of a conservative overapproximation. Second, we lift partial order semantics in the form of unfolding from Petri nets to parametric regulatory networks, harnessing concurrency for efficient state space representation.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France