RELATING THE STRUCTURE AND DYNAMICS OF GENE REGULATORY NETWORKS

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von

Ling SUN

Berlin, 2017

Erstgutachter: Prof. Dr. Alexander Bockmayr

Zweitgutachter: Dr. Élisabeth Remy, HDR

Tag der Disputation: 25. Oktober 2017

Abstract

A key topic in systems biology is to understand the intricate relations between molecular network structures, dynamic properties and biological function. In this context, gene regulatory networks (GRNs) describing the regulatory interactions between genes and their products are of crucial importance. The general goal of this thesis is to explore the relationship between the structure and dynamics of GRNs. This is done in a discrete modelling framework using the Thomas formalism. A GRN is represented by a discrete model which includes an interaction graph (IG) and a logical parameter function that characterise the regulatory interactions. The dynamics of a GRN is modelled by an asynchronous state transition graph (ASTG), where the states of the system can only be changed by asynchronous and unitary updates. In 2011, T. Lorenz proposed two reverse engineering algorithms for inferring from a given ASTG models satisfying specific properties.

In the first part of the thesis, the focus is on the explanation, implementation, and generalisation of the Lorenz algorithms. In order to handle general inputs, three necessary and sufficient conditions are presented to characterise ASTGs among all graphs on a given state space. Furthermore, a fourth condition is derived which is necessary and sufficient for an ASTG to admit a realistic model. These four ASTG conditions provide the basis for a generalisation of Lorenz algorithms and several applications.

Multistationarity and homeostasis are two important dynamical properties of high biological relevance, which can be represented by attractors in the ASTG. In the second part of the thesis, two discrete modelling workflows are developed for exploring all those GRNs that are able to realise a given functionality. The forward modelling workflow includes enumerating all possible models and searching for those models whose ASTG exhibits the desired properties. The reverse engineering workflow starts from enumerating all graphs on the state space satisfying the dynamic properties and then infers all models of these graphs using the generalised Lorenz algorithms. To analyse the resulting functional IGs, a logical analysis method is developed, which encodes IGs by Boolean expressions, and then uses Boolean function minimisation to obtain a compact representation. The same logical analysis method can also be applied to the logical parameters.

In the last part of the thesis, the discrete modelling workflows are applied to explore the space of GRNs realising some typical dynamic behaviours of biological interest. Three case studies are presented. The first one concerns homeostasis in a simplified MAPK cascade, the second one multistationarity in cell differentiation, and the third one single-stripe forming in the embryogenesis of the fruit fly *Drosophila melanogaster*.

Contents

Chapter 1

Introduction

Systems biology is a highly interdisciplinary research field that studies various components of a biological system in a systematic way rather than dealing with them in isolation. A central theme in systems biology is to understand the intricate relationship between the structure of a biological system, its dynamical properties and biological function [Kitano, 2002]. A variety of mathematical and computational approaches has been developed to model and analyse these biological systems, including continuous, discrete, stochastic and various hybrid modelling methods, see for example [De Jong, 2002, Ingalls, 2013] for review. In continuous modelling, systems are described by ordinary differential equations (ODEs). ODE models are quantitative and precise but their application is limited by the hardness of solving large systems of nonlinear differential equations and determining the huge amount of kinetic parameters. Requiring less data and parameters, piecewise linear differential equations (PLDEs) can qualitatively describe a biological system using limited kinetic information. Needing much less data and parameters than ODEs and PLDEs, the discrete modelling approach can qualitatively describe a system using integer variables and a finite number of states. Existing discrete methods include logical networks [Thomas and D'Ari, 1990], Petri nets [Chaouiya et al., 2008] and many others. Computational methods and analysis may predict biological mechanisms, which can motivate verification-oriented experiments [Kitano, 2002]. Computational analysis is strongly needed for better understanding the mechanisms underlying different biological functions, phenomena and also various diseases.

Gene regulation is the mechanism of controlling gene expression. The product of a gene can act as a transcription factor (TF) which can activate or inhibit other gene expression. Gene regulation can happen at the transcriptional or translational level, in single cell systems adjusting to an external environment, or in multicellular systems in the context of cell differentiation and morphogenesis. A set of genes and gene products interacting with each other and regulating the gene expression is called a *gene regulatory network* (GRN). The regulators can be DNA, RNA, proteins and their complexes. The interaction can be direct or indirect (through their transcribed RNA or translated protein).

The goal of this thesis is to study the relationship between the structure and dynamics of GRNs using a discrete logical modelling approach. Section 1.1 presents basic propositional logic and the classic Quine-McCluskey algorithm, which will be needed to understand later work in Chapter 5, 6, 7, 8. A brief history from continuous to discrete modelling of GRNs is given in Section 1.2. Next the fundamental definitions and concepts used in discrete modelling of regulatory networks are introduced in Section 1.3. Finally, an overview of the whole thesis can be found in Section 1.4.

1.1 Propositional logic and Quine-McCluskey Algorithm

A statement that has a truth value, *i.e.*, TRUE or FALSE, is called a *proposition*. Propositional variables can have truth values TRUE (1) or FALSE (0), and can be called logical variables. Propositional variables can be connected by logical operations including *and, or, not, implication etc.* to form *logical expressions*. If we use 1 and 0 to represent TRUE and FALSE, then the logical operations can be written as arithmetic expressions. Let x and y be two logical variables, then:

1. $x \vee y = x + y$. Here OR is *inclusive or*, or *disjunction*.
2. $x \wedge y = xy$.
3. $\bar{x} = 1 - x$.

At the core of propositional logic are Boolean functions [Crama and Hammer, 2011]. Any logical expression can be described by a Boolean function $f : \{0,1\}^n \to \{0,1\}$ of the corresponding logical variables. The truth values of all the variables and the Boolean function can be described by a *truth table*, where the rows contain the combinations of all possible values for all the logical variables, and the last column gives the truth value of the Boolean function. For example, Table 1.1a is a truth table for a Boolean function.

There are two classes of logical expressions of special interest. Let x be a Boolean variable, then a *literal* is an expression of the form x or $\bar{x}$. A conjunction (*product*) of a set of literals is called an *elementary conjunction*, and a disjunction (*sum*) of a set of literals is called an *elementary disjunction*. A *disjunctive normal form* (DNF) is a disjunction of elementary conjunctions which are called *terms* of the DNF. A *conjunctive normal form* (CNF) is a conjunction of elementary disjunctions which are called *clauses* of the DNF. A well-known fact is that every Boolean function can be written as a DNF and a CNF in [Crama and Hammer, 2011].

Let $\mathcal{B} = \{0,1\}$. A *minterm* on $\mathcal{B}^n, n \geq 1$ is an elementary conjunction involving all n literals. Let $f : \mathcal{B}^n \to \mathcal{B}$ be a Boolean function and consider the truth table of f. Each row in the truth table where f is true can be represented by a minterm. An *implicant* of a Boolean function f is an elementary conjunction, such that if this implicant is true, then the Boolean function f is also true. In other words, there is no value assignment which makes the product term true and the Boolean function false. An implicant is said to *cover* a minterm, if this implicant is true when this minterm is true. An implicant is *prime* if the number of literals of it cannot be reduced any more, *i.e.*, removing any literal will change the implicant into a non-implicant. A prime implicant is *essential* if it covers a minterm of the Boolean function which no other prime implicant can cover. These definitions are illustrated later in Example 1.1.

One of the most important topics in the theory of Boolean functions is the representation of a Boolean function by a DNF, or some logical expression. A Boolean function can have multiple DNF representations, therefore some "optimal" one is of particular interest. A DNF can be *short* regarding the number of terms and the number of literals. The problem of constructing a shortest DNF representation of a Boolean function is also called *logical minimisation*, *two-level logic minimisation* or *Boolean function minimisation* [Crama and Hammer, 2011]. Hammer and Kogan proved in [Crama and Hammer, 2011, Theorem 3.14] that the logic minimisation problem is NP-hard when its input is a Boolean function given by the set of its true points. First developed by Quine and later extended by McCluskey [McCluskey, 1956], the Quine-McCluskey algorithm is a general method for the minimisation of Boolean functions. The Quine-McCluskey algorithm

includes the following two steps and is illustrated by Example 1.1.

1. Find all prime implicants of the Boolean function by applying repeatedly the rule $xy + x\bar{y} = x$;
2. Put these prime implicants in a *prime implicant chart* to find all essential prime implicants of the function, and other prime implicants that are necessary to cover the function.

Example 1.1 (Quine-McCluskey algorithm) Table 1.1 illustrates the algorithm for a Boolean function of three variables. (This table is generated by the online software (`http://www.mathematik.uni-marburg.de/~thormae/lectures/ti1/code/qmc/`) by the lecture [Thormählen, 2016].)

	x_2	x_1	x_0	f
0 :	0	0	0	1
1 :	0	0	1	1
2 :	0	1	0	0
3 :	0	1	1	1
4 :	1	0	0	0
5 :	1	0	1	0
6 :	1	1	0	0
7 :	1	1	1	1

(a) Truth table.

	x_2	x_1	x_0	
0 :	0	0	0	$\rightarrow$
1 :	0	0	1	$\rightarrow$
3 :	0	1	1	$\rightarrow$
7 :	1	1	1	$\rightarrow$

(b) Implicants (Order 0).

	x_2	x_1	x_0	
0, 1 :	0	0	$-$	$\surd$
1, 3 :	0	$-$	1	$\surd$
3, 7 :	$-$	1	1	$\surd$

(c) Implicants (Order 1).

prime implicants	x_2	x_1	x_0	0	1	3	7	
0, 1 :	0	0	$-$	•	○			$\bar{x_2}\bar{x_1}$
1, 3 :	0	$-$	1		○	○		$\bar{x_2}x_0$
3, 7 :	$-$	1	1			○	•	x_1x_0

(d) Prime implicant chart. "•" denotes essential prime implicants, and "○" for non-essential ones.

Table 1.1: Quine-McCluskey algorithm for a Boolean function f. (a) The truth table for f. (b) The rows in (a) with $f = 1$ are the minterms which are implicants of order 0. (c) Those implicants of order 0 with only one different variable can be combined (written as $-$), giving the combined implicants of order 1. Those implicants which cannot be combined any further are prime, marked with "$\surd$", otherwise with "$\rightarrow$". (d) The prime implicant chart is constructed to find the essential prime implicants of f and other prime implicants which are necessary to cover all true minterms. A minterm that is covered by only one prime implicant is marked with "•". A minterm is marked with "○" if it is covered by more than one prime implicant. The prime implicants which cover "•" minterms are essential. The minimal logical expression is $f = \bar{x_2}\bar{x_1} \vee x_1x_0$.

1. All true minterms are selected in Table 1.1b. Using the rule $xy + x\bar{y} = x$, all minterms are combined until every row is a prime implicant, shown in Table 1.1c (order 1).
2. A prime implicant chart is constructed based on all prime implicants, which are put in the table as rows and all minterms with $f = 1$ as columns. Petrick's method [Petrick, 1959] is applied to get the minimal DNF, as in Table 1.1d. If a minterm is covered by only one prime implicant, then it is marked with "•". If a minterm is covered by more than one prime implicant, then it is marked with "○". A prime implicant is *essential* if a true minterm is covered by it, but not by any other prime implicant. All essential prime implicants must be included in the minimal Boolean expression. After all the essential prime implicants have been found, a minimum set of prime implicants must be chosen to

cover the remaining minterm columns.

3. The sum of all the essential prime implicants and the other minimum prime implicants is the minimisation of the Boolean expression. In this example, $f = \bar{x_1} \vee x_2 x_0$.

△

For more information on the QuineMcCluskey algorithm, we refer to the original work in [Quine, 1952, McCluskey, 1956]. A standard reference on Boolean functions is [Crama and Hammer, 2011].

1.2 Continuous and discrete modelling frameworks

In continuous modelling, the concentration of a gene product is described by a real variable. A gene i can be activated or inhibited by another gene j if the concentration of j reaches a certain threshold θ_{ij}. The regulatory effect is modelled by a sigmoid curve, which is usually described by a *Hill function* [Hill, 1910], see Fig. 1.1.

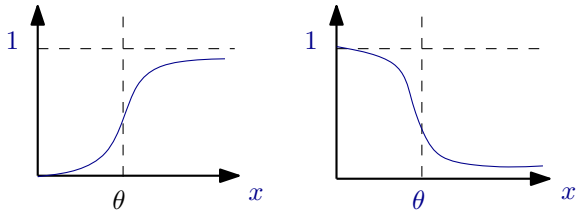


Figure 1.1: Hill functions, m is the Hill coefficient.

Assume a regulatory network has n genes modelled by n real variables, x_1, x_2, ..., x_n. The regulation (activation/inhibition) of i by j is described by a kinetic parameter k_{ij} and the degradation rate k_{-i}. The system can be modelled by the following ordinary differential equations (ODEs), see e.g. [Thomas and Kaufman, 2001]:

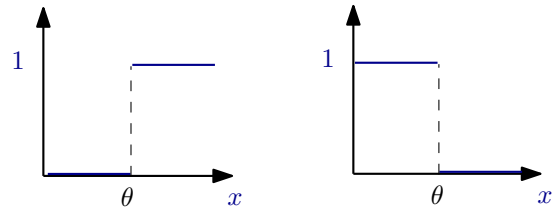
$$\frac{dx_i}{dt} = \sum_{j=1}^{n} k_{ij} F_m^{\alpha_{ij}}(x_j, \theta_{ij}) - k_{-i} x_i, \text{ for } i = \{1, \ldots, n\} \tag{1.1}$$

where $k_{ij} \geq 0$, $\alpha_{ij} \in \{+, -\}$, $k_{-i} > 0$ and $m \geq 1$.

For $m \to \infty$ the Hill function can be approximated as a step function (see Fig. 1.2). Thus, the ODE can be transformed into a piecewise linear differential equation (PLDE).

$$\frac{dx_i}{dt} = \sum_{j=1}^{n} k_{ij} F^{\alpha_{ij}}(x_j, \theta_{ij}) - k_{-i} x_i, \text{ for } i = \{1, \ldots, n\} \tag{1.2}$$

where $k_{ij} \geq 0$, $\alpha_{ij} \in \{+, -\}$ and $k_{-i} > 0$.



$$F^+(x,\theta) = \begin{cases} 1, & if\ x > \theta \\ 0, & if\ x < \theta \end{cases} \qquad F^-(x,\theta) = 1 - F^+(x,\theta)$$

Figure 1.2: The step function.

Thomas and co-workers developed kinetic logic and a generalised logical description of gene regulatory networks using discrete logical variables [Thomas, 1973, Thomas, 1981], which can be Boolean (only 0 and 1) or multi-valued.

Suppose a gene i acts on n_i different genes in the system. Then it is natural to introduce a threshold for each of the genes: $0 < \theta^1 < \theta^2 < \cdots < \theta^{n_i}$ (following Thomas, we assume that all thresholds are different). Using these thresholds, continuous variables can be discretized as follows:

$$d_i : \mathbb{R}^+ \to \{0, 1, \ldots, n_i\}, \quad d_i(x_i) = \begin{cases} 0, & \text{if } 0 < x_i < \theta^1 \\ 1, & \text{if } \theta^1 < x_i < \theta^2 \\ \vdots & \\ n_i, & \text{if } x_i > \theta^{n_i} \end{cases} \tag{1.3}$$

The PLDEs (1.2) can be transformed into the discrete equations [Thomas and D'Ari, 1990]:

$$X_i^{\uparrow} = d_i(\sum_{j=1}^{n} \mathrm{K}_{ij} F^{\alpha_{ij}}(X_j, \vartheta_{ij})), \tag{1.4}$$

with discrete variables $X_i, X_i^{\uparrow} \in \{0, \ldots, n_i\}$, where $X_i^{\uparrow}$ is the new value of X_i, $\mathrm{K}_{ij} = k_{ij}/k_{-i} \geq 0$, and ϑ_{ij} are discrete thresholds. Furthermore, we obtain discrete logical parameters defined by

$$\begin{aligned} \mathbf{K}_{ij} &:= d_i(\mathrm{K}_{ij}) \\ \mathbf{K}_{ij+ij'} &:= d_i(\mathrm{K}_{ij} + \mathrm{K}_{ij'}) \end{aligned} \tag{1.5}$$

Here both genes j and j' are regulating gene i, and $\mathbf{K}_{ij}, \mathbf{K}_{ij+ij'} \in \{0, \ldots, n_i\}$, where n_i is the maximal concentration level of gene i.

Example 1.2 Figure 1.3 shows two genes regulating each other.

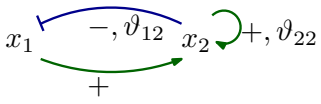


Figure 1.3: An example of a multi-valued interaction graph. $x_1 \in \{0, 1\}$, $x_2 \in \{0, 1, 2\}$. Let $\vartheta_{12} = 1 < \vartheta_{22} = 2$. The arrows denote activation, and the blunt edge represents inhibition.

In Figure 1.3, the ODEs for the system will be:

$$\frac{dx_1}{dt} = k_{12} F_m^-(x_2, \theta_{12}) - k_{-1} x_1 \tag{1.6}$$

$$\frac{dx_2}{dt} = k_{21} F_m^+(x_1, \theta_{21}) + k_{22} F_m^+(x_2, \theta_{22}) - k_{-2} x_2$$

After discretization, we get the equations:

$$X_1^{\uparrow} = d_1(\mathrm{K}_{12}F^{-}(X_2, 1)) \tag{1.7}$$

$$X_2^{\uparrow} = d_2(\mathrm{K}_{21}F^{+}(X_1, 1) + \mathrm{K}_{22}F^{+}(X_2, 2))$$

Let $\mathbf{K}_{12} = d_1(\mathrm{K}_{12})$, $\mathbf{K}_{21} = d_2(\mathrm{K}_{21})$, $\mathbf{K}_{22} = d_2(\mathrm{K}_{22})$ $\mathbf{K}_{21+22} = d_2(\mathrm{K}_{21} + \mathrm{K}_{22})$. Then the discrete dynamics can be described by the following table:

X_1	X_2	$X_1^{\uparrow}$	$X_2^{\uparrow}$
0	0	0	0
0	1	$\mathbf{K}_{12}$	0
0	2	$\mathbf{K}_{12}$	$\mathbf{K}_{22}$
1	0	0	$\mathbf{K}_{21}$
1	1	$\mathbf{K}_{12}$	$\mathbf{K}_{21}$
1	2	$\mathbf{K}_{12}$	$\mathbf{K}_{21+22}$

△

Biological feedbacks are an important characteristic of regulatory networks. Feedback circuits are defined as simple closed cycle of directed interactions [Thieffry, 2007]. They can be positive or negative depending on whether there is an even or odd number of inhibitions. R. Thomas formulated two conjectures: a positive circuit in a regulatory network is a necessary condition for multistationarity and a negative circuit is necessary for stable periodic behaviour [Thomas and D'Ari, 1990]. Two of the most important biological properties are differentiation and homeostasis or oscillations. In terms of modelling, they correspond to multistationarity and stable periodicity. In ODE or PLDE systems, these are related to asymptotically stable steady states and homeostasis. In discrete dynamical systems, the relevant dynamic behaviours are called stable states or fix points, and cyclic attractors. Note that asymptotically stable steady states in a continuous model may correspond to both fix points and cyclic attractors in a related discrete model [Snoussi, 1989, Jamshidi, 2013].

R. Thomas, E. H. Snoussi, D. Thieffry and co-workers initiated the research on (asynchronous) logical modelling of GRNs and made it an active research area, see for example [Bérenguier et al., 2013, Chaouiya and Remy, 2013] for recent surveys.

Lots of work has focused on positive/negative feedback circuits and related dynamic properties including multistationarity and homeostasis, for both Boolean and multi-valued systems [Soulé, 2003, Remy et al., 2006, Remy and Ruet, 2006, Richard and Comet, 2007, Remy et al., 2008, Richard, 2009, Richard, 2010, Didier and Remy, 2012, Comet et al., 2013, Melliti et al., 2015, Remy et al., 2016, Ruet, 2017].

Important work on network inference includes [Bernot et al., 2004, Batt et al., 2010, Corblin et al., 2010, Klarner et al., 2012b, Klarner et al., 2012a, Klarner, 2015, Videla et al., 2015, Ostrowski et al., 2016, Streck, 2016]. Note, however, that in contrast to the work presented in this thesis, all these methods do not start from a complete asynchronous state transition graph as input for network inference. In the next section, we formally introduce these concepts.

1.3 Preliminaries on discrete modelling of regulatory networks

The following definitions are based on [Lorenz, 2011] and [Lorenz et al., 2013]. In discrete models, the activity levels of a gene are represented by integer numbers $0, 1, \ldots, k$, for some

$k \in \mathbb{N}^+ = \{1, 2, \dots\}$. In this section, we give a formal treatment of a purely discrete version of the Thomas modelling approach introduced in Section 1.2.

A gene and the product of it are modelled by a node, and an interaction from one gene product to another gene by a directed edge from one node to the other. The resulting graph on all nodes in the system is called an *interaction graph* [Lorenz, 2011].

Definition 1.3 (Interaction graph) An *interaction graph (IG)* $I = (V, E, \varepsilon, \vartheta, \max)$ is a directed graph, with a set of vertices V representing regulatory components, and a set of edges E denoting regulatory interactions among V. The function $\varepsilon : E \to \{+, -\}$ specifies the sign of the edges in E, where $+$ means activation and $-$ means inhibition. The function $\vartheta : E \to \mathbb{N}^+$ defines for each edge (u, v) a threshold $\vartheta(u, v)$,which is the minimal activity level of the component u that is required by the interaction (u, v) to be effective. The function $\max : V \to \mathbb{N}^+$ assigns a positive integer to each component, which is the maximal activity level.

Definition 1.4 (Successor and predecessor) Let $(u, v) \in E$ be an edge. Then v is a *successor* of u, and u is a *predecessor* of v. The set of all predecessors (resp. successors) of u is denoted by $\mathrm{Pre}(u) := \{v \in V \mid (v, u) \in E\}$ (resp. $\mathrm{Suc}(u) := \{v \in V \mid (u, v) \in E\}$).

The state space X is the Cartesian product of the sets of values of all the components. It describes all possible states of the regulatory network defined by an IG I.

Definition 1.5 (State Space) For an IG $I = (V, E, \varepsilon, \vartheta, \max)$, the *state space* is defined as $X := \prod_{u \in V} X_u$, where $X_u = \{0, \dots, \max_u\}$ is the activity level interval of the component $u \in V$. A *state* is a vector $x := (x_1, \dots, x_{|V|}) \in X$.

The state of a component u depends on a combined effect of all components v regulating u. The next state of a component u is specified by the so-called resources and the logical parameter function, which are defined next.

Definition 1.6 (Resource) The *resource* for a component $u \in V$ in a state $x \in X$ is defined as the set of predecessors of u which in state x have a present activation or an absent inhibition on u, *i.e.*,

$$\mathrm{Res}_u(x) = \left\{ v \in \mathrm{Pre}(u) \;\middle|\; \begin{array}{lll} \varepsilon(v,u) = + & \wedge & x_v \geq \vartheta(v,u) \\ \varepsilon(v,u) = - & \wedge & x_v < \vartheta(v,u) \end{array} \right\}.$$

Definition 1.7 (Logical parameter function) A *logical parameter function* K is a function of the components and their resources. For all $u \in V$ and $\omega \subseteq \mathrm{Pre}(u)$, the logical parameter $K(u, \omega) \in \{0, \dots, \max_u\}$ gives a value to which a component u in a state x with resource $\mathrm{Res}_u(x) = \omega$ tends.

Definition 1.8 (Model) A *model* $M = (I, K)$ is composed of an interaction graph I and a logical parameter function K.

The transition of one state to another is described by the state transition function based on the logical parameter function.

Definition 1.9 (State transition function) The *state transition function* $\delta : V \times X \to \{-1, 0, 1\}$ indicates how a component u can change in a given state $x \in X$:

$$\delta(u, x) := \begin{cases} 1 & \text{if } x_u < K(u, \mathrm{Res}_u(x)), \\ 0 & \text{if } x_u = K(u, \mathrm{Res}_u(x)), \\ -1 & \text{if } x_u > K(u, \mathrm{Res}_u(x)). \end{cases}$$

We mainly focus on two strategies for describing the transitions from one state to another. According to whether different components of a state are updated simultaneously or not, there are *synchronous* and *asynchronous* update strategies. Depending on whether the value of a

component is changed by one or more than one, there can be *unitary* and *non-unitary* updates. In non-unitary updating a component x_u can be updated to any value in $\{0, \ldots, \max_u\}$, while in unitary updating, a component can change at most by value one. According to [Thomas and D'Ari, 1990], the likelihood of two components being updated in exactly the same moment is very small, and the accumulation of a gene product is more a continuous process than a sudden jump. Therefore, the *unitary asynchronous* update strategy is considered to be more realistic in describing the dynamics of biological systems. Thus, in this thesis we will mainly use the unitary asynchronous dynamics.

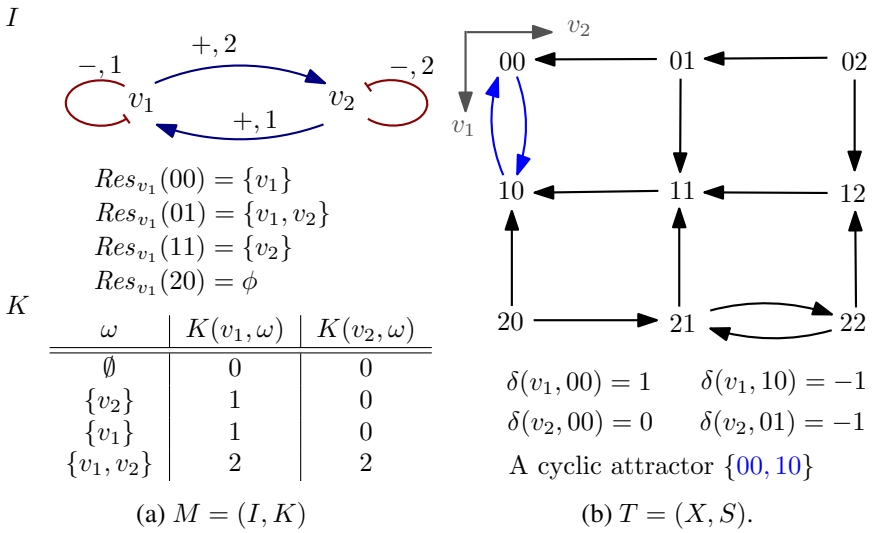
Definition 1.10 (Asynchronous state transition graph) The dynamics generated by a model M is described by the *asynchronous state transition graph (ASTG)* $T_M = (X, S)$, where

$$S := \bigcup_{x \in X} \{(x, x + \delta(u, x)\mathbf{e}^u) \mid u \in V : \delta(u, x) \neq 0.\}$$

Here, $\mathbf{e}^u$ is the u-th unit vector in X.

As mentioned in Section 1.3, multistationarity and stable periodic behaviour are two important biological properties. In the discrete modelling framework, these correspond to stable states and closed cycles (so called cyclic attractors) in the ASTG, which are are defined next.

Definition 1.11 (Attractor) Given an ASTG $T = (X, S)$, a subset $A \subseteq X$ is called an *attractor*, if A is a strongly connected component and there is no transition leaving A. An attractor containing only one element is called a *stable state*, and an attractor with more than one element is called a *cyclic attractor*.



(a) $M = (I, K)$ (b) $T = (X, S)$.

Figure 1.4: (a) A model $M = (I, K)$. In the IG I, the arrows represent activation, and the edges with a small dash in the end represent inhibition. At the bottom, there is a logical parameter function K, where ω denotes the resource. In between I and K, there are a few examples of resources. (b) The corresponding ASTG $T = (X, S)$, where the elements of X are given by sequences of numbers, S is the set of all the transitions in X. A few examples of the state transition function $\delta(v, x)$ are also given. Note that $\{00, 10\}$ is the only attractor in T, and $\{21, 22\}$ is not an attractor because it has outgoing transitions.

1.4 Overview of the thesis

This chapter has introduced some background and preliminaries including propositional logic, the Quine-McCluskey algorithm, the Thomas formalism, and discrete modelling of regulatory networks.

Chapter 2 will present and elucidate four reverse engineering algorithms originally proposed by Therese Lorenz [Lorenz, 2011]. In the sequel, these will be called *Lorenz Algorithms*. The principles and details of Lorenz Algorithms will be introduced and illustrated by example.

The main theoretical contribution of this thesis is Chapter 3. First three conditions which are necessary and sufficient for a graph on the state space to be an ASTG are presented in Section 3.1.1. Based on these three conditions, rules are given in Section 3.1.3 for the enumeration of all possible ASTGs on a given state space. A model is called *compatible* if it satisfies the observability and Snoussi-conditions. In Section 3.1.2, a necessary and sufficient condition on an ASTG for being able to carry a compatible model is proposed. Lorenz algorithms as introduced in Chapter 2 require as input a valid ASTG, otherwise the output model is not guaranteed to regenerate the input. Based on the four conditions developed in Section 3.1.1 and 3.1.2, Lorenz Algorithms are generalised in Section 3.2 to handle also general inputs.

In Chapter 4, first a discrete robustness measure, named *realizability*, is proposed in Section 4.1 in order to describe the ability of an IG of realising a required attractor. Next, a new logical analysis method to identify regulatory patterns in functional IGs and logical parameter functions is proposed in Section 4.2.1. Based on the two analysis methods above and the updated Lorenz Algorithms, two discrete modelling workflows are proposed for exploring the regulatory structures which can reproduce some given biological function. Starting from the target function which is modelled by certain attractors, these two workflows are able to find all functional models which can generate ASTGs with the required attractors. Both workflows rely on enumeration. Thus they suffer from exponential complexity with an increasing number of components and even worse if the components have high maximal activity values. After this, a logical analysis method can be applied on the functional IGs to generate IG patterns. If the functionality for the required attractors is a Boolean function which is true for each functional IG, then one can derive a minimal logical description of this set of IGs, usually the minimal DNF or CNF. Similarly this method can be applied on the logical parameters to identify potential interesting patterns.

Chapter 5, 6, 7 describe three applications of the methodologies in Chapter 3 and 4, with different biological targets.

Chapter 5 studies all possible 3-node IGs that are able to reproduce the cyclic attractor from a simplified 3-node MAPK cascade signalling network [Thobe et al., 2014]. Using the reverse engineering workflow, 8 functional IGs are identified, where a core motif and 3 interaction-pairs are found to be relevant IG patterns. When applying the logical analysis method on this set of IGs, the minimal CNF turns out to provide shorter Boolean expressions than the minimal DNF.

Chapter 6 explores all possible 3-node IGs that are able to reproduce a set of 3 steady states for a cell differentiation process originally studied by a continuous modelling approach [Breindl et al., 2011]. Two opposite hypotheses on the 3 stable states are used to define the target attractors. The discrete modelling workflows are applied to obtain all functional IGs and IG patterns. The IG patterns from the discrete method are compared with the building blocks from the continuous approach. The realizability and logical analysis method are applied on the resulting functional models.

Chapter 7 studies all possible underlying GRNs which are able to generate the single stripe phenomenon observed in the development of the *Drosophila* embryo [Cotterell and Sharpe, 2010]. The single stripe phenomenon is modelled by a sequence of attractors under a morphogen gradient input signal. The details on modelling the morphogen gradient and the single stripe phenomenon are presented in Section 7.1. Using the discrete modelling workflows, all functional IGs are found and also the IG patterns are obtained. A compact representation of all the functional IGs is achieved by applying the logical analysis method, see in Section 7.3.

The thesis finishes in Chapter 8 with a summary and a discussion of possible future work.

Chapter 2

From dynamics to structures: reverse engineering algorithms

In the discrete modelling framework introduced in Chapter 1, models are defined in a very general way, so that they may contain unnecessary interactions whose influences are not visible in the ASTG, or there may be inconsistencies between the IG and the logical parameter function. Inferring structures from dynamics, called reverse engineering in the sequel, is significant for understanding the relationship between the dynamics of GRNs and their structure. This chapter will focus on different model conditions and some reverse engineering algorithms originally developed by T. Lorenz [Lorenz, 2011].

Section 2.1 provides some theoretical background and the basic principles for the reverse engineering algorithms. A discrete model can satisfy different model conditions, which will be discussed in Section 2.1.1. ASTGs are described by row structures, which are introduced in Section 2.1.2. Furthermore in multi-valued modelling, it is possible that different models can have the same ASTG. Section 2.1.3 gives definitions and basic properties of equivalent models.

T. Lorenz [Lorenz, 2011] designed four algorithms: two for reverse engineering and two auxiliary ones. Given a complete ASTG, each of the two reverse engineering algorithms can infer an *optimal* model satisfying specific model conditions. Here *optimality* means that there are no unnecessary edges regarding the model conditions required by the corresponding algorithm. The row properties introduced in Section 2.1.2 form the basis for understanding the two auxiliary algorithms *Logical Parameters* in Section 2.2.1 and *Activity Value* in Section 2.2.2. The model conditions defined in Section 2.1.1 and the notion of equivalent models introduced in Section 2.1.3 are important to understand the other two algorithms in Section 2.2.3 and 2.2.4. Section 2.2 gives pseudocode for each of the four algorithms and illustrates them by example.

2.1 Model conditions and ASTG characterisation

Section 2.1.1 will introduce the three model conditions, *i.e.*, *visibility*, *observability* and the *Snoussi*-conditions. Section 2.1.2 gives a detailed characterisation of the structure of ASTGs, important principles and theorems. Specifically, the minimisation of violations of the Snoussi-condition in Section 2.1.3 is needed for understanding the algorithms in Section 2.2.4.

2.1.1 Model conditions

As defined in Chapter 1, a model $M = (I, K)$ includes an IG $I = (V, E, \varepsilon, \vartheta, \max)$ and a logical parameter function K. Following [Lorenz, 2011], three model conditions are defined below.

Definition 2.1 (Visibility condition) In a model $M = (I, K)$, an interaction $(v, u) \in E$ is called *visible*, if there exists a resource $\omega \subseteq \mathrm{Pre}(u)$ such that

$$K(u, \omega) \neq K(u, \omega \cup \{v\}).$$

A model $M = (I, K)$ satisfies the *visibility condition* if all edges $(v, u) \in E$ are visible.

Definition 2.2 (Observability condition) In a model $M = (I, K)$, an interaction (v, u) is *observable* if there exists a resource $\omega \subseteq \mathrm{Pre}(u)$ such that

$$K(u, \omega) < K(u, \omega \cup \{v\}).$$

A model $M = (I, K)$ satisfies the *observability condition* if all edges $(v, u) \in E$ are observable.

Remark 2.3 An observable edge (v, u) is always visible, but a visible edge (v, u) is not always observable.

The logical parameter function K is based on resources, so that for the observability condition, the relation "$<$" holds for both the positive and negative interactions. The "$<$" is rather natural for an activation. Consider an inhibition (v, u) and two states x, $y \in X$ with $x_v \geq \vartheta(v, u)$ and $y_v < \vartheta(v, u)$. Then $v \notin \mathrm{Res}_u(x) = \omega$, and $v \in \mathrm{Res}_u(y) = \omega \cup \{v\}$. The inhibition (v, u) is observable if $K(u, \omega) = K(u, \mathrm{Res}_u(x)) < K(u, \omega \cup \{v\}) = K(u, \mathrm{Res}_u(y))$, where the inhibitor v is present in state x and absent in state y.

Definition 2.4 (Snoussi-condition)

1. An *interaction* $(v, u) \in E$ of a model $M = (I, K)$ satisfies the *Snoussi-condition*, if

$$\forall \omega \subseteq \mathrm{Pre}(u) \backslash \{v\}, \;\; K(u, \omega) \leq K(u, \omega \cup \{v\}).$$

2. A *component* $u \in V$ of a model $M = (I, K)$ satisfies the *Snoussi-condition*, if

$$\forall \omega \subseteq \varsigma \subseteq \mathrm{Pre}(u), \;\; K(u, \omega) \leq K(u, \varsigma).$$

3. A *model* $M = (I, K)$ satisfies the *Snoussi-condition* if each component $u \in V$ satisfies the Snoussi-condition. This means that when adding positive influences, no component tends to a lower value.

Remark 2.5 A *model* $M = (I, K)$ satisfies the *Snoussi-condition* if every interaction $(v, u) \in E$ satisfies the Snoussi-condition.

Proof. If every interaction satisfies the Snoussi-condition, then it holds that for all $u, v \in V$, for all $\omega \subseteq \mathrm{Pre}(u) \backslash \{v\}$, $K(u, \omega) \leq K(u, \omega \cup \{v\})$. For all resources of u, $\omega \subseteq \varsigma \subseteq \mathrm{Pre}(u)$, we can prove that $K(u, \omega) \leq K(u, \varsigma)$.

Let $\omega \subseteq \varsigma \subseteq V$ that $\varsigma \backslash \omega = \{v_1, \ldots, v_b\}$, where $v_1, \ldots, v_b \in \mathrm{Pre}(u)$. Let $\omega^1 = \omega \cup \{v_1\}$, $\omega^2 = \omega^1 \cup \{v_2\}$ until $\omega^b = \omega^{b-1} \cup \{v_b\} = \varsigma$. Each interaction satisfies the Snoussi-condition, infers that $K(u, \omega) \leq K(u, \omega^1) \leq K(u, \omega^2) \leq \cdots \; K(u, \omega^b)$. So that $K(u, \omega) \leq K(u, \varsigma)$. Therefore, component u satisfies the Snoussi-condition.

A *model* $M = (I, K)$ satisfies the *Snoussi-condition* if every interaction $(v, u) \in E$ satisfies the Snoussi-condition. □

Definition 2.6 (Violation of the Snoussi-condition) In a model $M = (I, K)$, there exists a *violation of the Snoussi-condition*:

1. *in a component* $u \in V$, if for some $\omega \subseteq \varsigma \subseteq \mathrm{Pre}(u)$, $K(u,\omega) > K(u,\varsigma)$.
2. *in an interaction* (v,u), if $K(u,\omega) > K(u,\omega \cup \{v\})$, for some $\omega \subseteq \mathrm{Pre}(u)\backslash\{v\}$.

2.1.2 Row properties of an ASTG

The analysis of u-row properties will be presented in this section, which is one of the key contributions from [Lorenz, 2011, Lorenz et al., 2013].

Let $M = (I, K)$ be a model with corresponding ASTG $T_M = (X, S_M)$, where X is the state space and S_M describes the transitions in X. Some definitions and lemmas from [Lorenz, 2011, Lorenz et al., 2013] about rows in X are introduced next.

Definition 2.7 (u-row) [Lorenz, 2011] Given $u \in V$, a *u-row* is a sequence of states $\tau^u = (x^0, \ldots, x^{\max_u})$ in X, with $x^0_u = 0$ and $x^l = x^0 + l\mathbf{e}^u$, for all $l \in \{1, \ldots, \max_u\}$, where $\mathbf{e}^u$ is the u-th unit vector.

Definition 2.8 (State transitions of a u-row in direction of u) For a u-row $\tau^u = (x^0, \ldots, x^{\max_u})$, the *state transitions of τ^u in direction u* are given by

$$\delta(u, \tau^u) = (\delta(u, x^0), \ldots, \delta(u, x^{\max_u})).$$

The symmetric difference of two sets A and B is defined by $A\Delta B = (A\backslash B) \cup (B\backslash A)$.

Next we recall a Lemma from [Lorenz et al., 2013].

Lemma 2.9 Given a u-row $\tau^u = (x^0, \ldots, x^{\max_u})$ in T_M, let $\mathrm{Res}_u(x^0) = \omega$.
If $(u,u) \notin E$, then $\mathrm{Res}_u(x^i) = \omega$ for all $i \in \{0, \ldots, \max\}$.
If $(u,u) \in E$ and $\vartheta(u,u) = t \in \{1, \ldots, \max_u\}$, then $\mathrm{Res}_u(x^i) = \omega$ for $i < t$, and $\mathrm{Res}_u(x^i) = \omega\Delta\{u\}$ for $i \geq t$.

Proof. Let $\tau^u = (x^0, \ldots, x^{\max_u})$ be a u-row.

If a component u has no self loop $(u,u) \notin E$ and $\mathrm{Res}_u(x^0) = \omega$, then $\mathrm{Res}_u(x^i) = \omega$, for all $i \in \{0, \ldots, \max_u\}$.

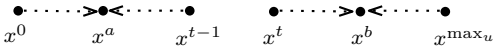
Suppose u has a self loop $(u,u) \in E$.

- If $\varepsilon(u,u) = +$, $\vartheta(u,u) = t$ and $\mathrm{Res}_u(x^0) = \omega$, then $\mathrm{Res}_u(x^i) = \omega$, for $i < t$ and $\mathrm{Res}_u(x^i) = \omega \cup \{u\}$, for $i \geq t$.
- If $\varepsilon(u,u) = -$, $\vartheta(u,u) = t$ and $\mathrm{Res}_u(x^0) = \omega'$, note that $u \in \omega'$, by the definition of resources. It follows $\mathrm{Res}_u(x^i) = \omega'$, for $i < t$ and $\mathrm{Res}_u(x^i) = \omega'\backslash\{u\}$, for $i \geq t$.

□

Proposition 2.10 (Row types) [Lorenz, 2011, Lorenz et al., 2013] Given an ASTG T_M, for each u-row $\tau^u = (x^0, \ldots, x^{\max_u})$, exactly one of the following situations holds:

(1) τ^u has the form:



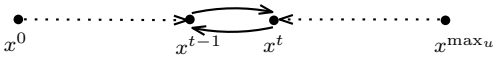
In the row structure above, there exist $a, b \in \{0, \ldots, \max_u\}$ with $a < b$ and $\delta(u, x^a) = \delta(u, x^b) = 0$. There exists $t \in \{1, \ldots, \max_u\}$ so that $K_I(u, \mathrm{Res}_u(x^i)) = a$ for all $i < t$ and $K_I(u, \mathrm{Res}_u(x^i)) = b$ for all $i \geq t$. Let $\mathrm{Res}_u(x^0) = \omega$ and $\mathrm{Res}_u(x^{\max_u}) = \varsigma$. It follows that

$$\begin{array}{ll} (u,u) \in E, & \vartheta(u,u) = t, \\ K_I(u,\omega) = a, & K_I(u,\varsigma) = b, \end{array}$$

and $\varsigma = \omega\Delta\{u\}$, as $(u,u) \in E$.

If the model $M = (I, K_I)$ satisfies the Snoussi-condition, then $\varepsilon(u,u) = +$.

(2) τ^u has the form:



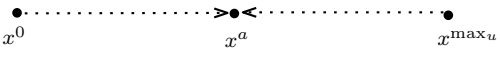
In the row structure above, there exists $t \in \{1, \ldots, \max_u\}$, with $\delta(u, x^i) = 1$ for all $i < t$ and $\delta(u, x^i) = -1$ for all $i \geq t$. Thus $K_I(u, \mathrm{Res}_u(x^i)) > t-1$ for all $i \in \{0, \ldots, t-1\}$ and $K_I(u, \mathrm{Res}_u(x^i)) < t$ for all $i \in \{t, \ldots, \max_u\}$. Let $\mathrm{Res}_u(x^0) = \omega$ and $\mathrm{Res}_u(x^{\max_u}) = \varsigma$. It follows that

$$\begin{array}{ll} (u,u) \in E, & \vartheta(u,u) = t, \\ K_I(u,\omega) > t-1, & K_I(u,\varsigma) < t, \end{array}$$

and $\varsigma = \omega\Delta\{u\}$, as $(u,u) \in E$.

If the model $M = (I, K_I)$ satisfies the Snoussi-condition, then $\varepsilon(u,u) = -$.

(3) τ^u has the form:



In the row structure above, there exists $a \in \{0, \ldots, \max_u\}$ such that $K_I(u, \mathrm{Res}_u(x^i)) = a$ for all $i \in \{0, \ldots, \max_u\}$. Let $\mathrm{Res}_u(x^0) = \omega$ and $\mathrm{Res}_u(x^{\max_u}) = \varsigma$. There are three possibilities:

$$\begin{array}{lll} \text{a)} & (u,u) \notin E, & K_I(u,\omega) = a. \\ \text{b)} & (u,u) \in E, & \vartheta(u,u) \geq a, \\ & K_I(u,\omega) = a, & K_I(u,\varsigma) < \vartheta(u,u). \\ \text{c)} & (u,u) \in E, & \vartheta(u,u) < a, \\ & K_I(u,\omega) \geq \vartheta(u,u), & K_I(u,\varsigma) = a. \end{array}$$

A u-row in T_M of form (1) or (2) also gives the threshold value $\vartheta(u,u)$. If there are no such u-rows of form (1) or (2) in T_M, then $(u,u) \notin E$.

Definition 2.11 gives a name for each type of row structure introduced above.

Definition 2.11 (*pos*, *neg*, *open* type) The three row structures in Proposition 2.10 are named as *pos*, *neg*, and *open* type.

- Row structure (1) is called *pos* type, because this kind of u-row indicates a positive self loop.
- Similarly, row structure (2) is called *neg* type, because it indicates a negative self loop.
- Finally, row structure (3) is called *open* type, due to the possibility of either having a loop or no loop at all.

A u-row of *pos* or *neg* type with $\vartheta(u,u) = t$ is called a *u-row of threshold t*.

Corollary 2.12 and Remark 2.13 directly follow from Proposition 2.10.

Corollary 2.12 Consider a model M which satisfies the observability and the Snoussi-condition and the corresponding ASTG T_M. For any component $u \in V$, the following holds:

1. If $(u,u) \notin E$, then all u-rows in T_M are of type *open*.
2. If $(u,u) \in E$ and $\varepsilon(u,u) = +$ (resp. $-$), then there exists at least one u-row in T_M of type *pos* (resp. *neg*).
3. If there are two u-rows in T_M of type *pos* (resp. *neg*), then the threshold values $\vartheta(u,u)$ for both u-rows are the same.

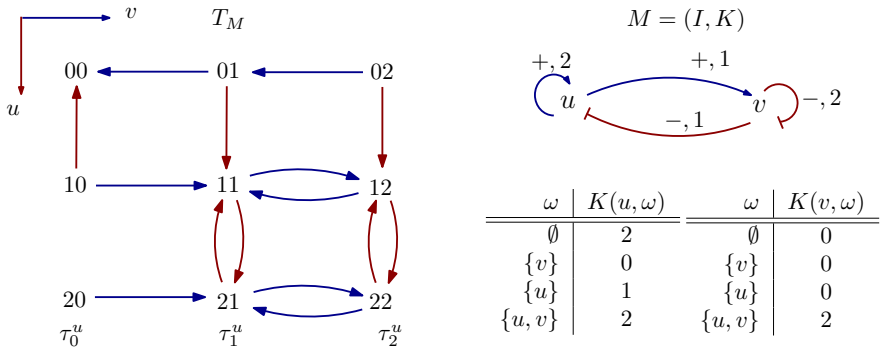
Proof. Assume that M satisfies the observability and the Snoussi-condition.

1. According to Proposition 2.10, there are only three possible row types. Any u-row of either *pos* or *neg* type will indicate a self loop on u. Therefore, if $(u,u) \notin E$, then all u-rows are of *open* type.
2. It is a direct conclusion from Proposition 2.10 in row type 1 and 2.
3. If two u-rows (either of *pos* or *neg* type) in T_M are of different thresholds, then u has multiple self loops, which violates the definition of the model and the hypothesis that T_M corresponds to M.

□

Remark 2.13 If for an ASTG T_M and a component $u \in V$ there are both type *pos* and *neg* u-rows, then the corresponding model M does not satisfy the Snoussi-Condition.

Example 2.14 Figure 2.1 shows an ASTG T_M and a corresponding model M. In T_M, the u-row τ_0^u is of *pos* type, and τ_2^u is of *neg* types.



ω	$K(u,\omega)$
$\emptyset$	2
$\{v\}$	0
$\{u\}$	1
$\{u,v\}$	2

ω	$K(v,\omega)$
$\emptyset$	0
$\{v\}$	0
$\{u\}$	0
$\{u,v\}$	2

Figure 2.1: An ASTG T_M and a corresponding model M.

In T_M, from τ_0^u and τ_2^u, according to Proposition 2.10, it holds for any model that:

$$K(u, \mathrm{Res}_u(00)) = 0, K(u, \mathrm{Res}_u(20)) = 2,$$
$$K(u, \mathrm{Res}_u(02)) > 1, K(u, \mathrm{Res}_u(22)) < 2.$$

It holds that $K(u, \mathrm{Res}_u(00)) < K(u, \mathrm{Res}_u(20))$ and $K(u, \mathrm{Res}_u(02)) < K(u, \mathrm{Res}_u(22))$. Moreover, $\mathrm{Res}_u(00)$ and $\mathrm{Res}_u(20)$ correspond to a pair of resources with and without $\{u\}$, and so do $\mathrm{Res}_u(02)$ and $\mathrm{Res}_u(22)$, as shown in Table 2.1. Therefore, for any model of T_M, there always exists a violation of the Snoussi-condition on the interaction (u,u).

for any model of T_M		if $\varepsilon(u,u)=+$	if $\varepsilon(u,u)=+$
extremal states x	$K(u,\mathrm{Res}_u(x))$	$\mathrm{Res}_u(x)$	$\mathrm{Res}_u(x)$
00	0	$\{v\}$	$\{u,v\}$
20	2	$\{u,v\}$	$\{v\}$
02	>1	$\emptyset$	$\{u\}$
22	<2	$\{u\}$	$\emptyset$

Table 2.1: The resources of u under different signs of the interaction (u,u).

Definition 2.15 (Extremal state and extremal row) [Lorenz, 2011] A state $x \in X$ is called *extremal*, if $x_u \in \{0, \max_u\}$, for all $u \in V$. A u-row $\tau^u = (x^0, \ldots, x^{\max_u})$ is *extremal*, if one of x^0 and $x^{\max_u}$ is an extremal state.

Lemma 2.16 [Lorenz, 2011] Given an IG I, a component $u \in V$ and a resource $\omega \subseteq \mathrm{Pre}(u)$, there always exists an extremal state $x \in X$ such that $\mathrm{Res}_u(x) = \omega$.

Proof. For all $v \in V$, define

$$x_v := \begin{cases} 0 & v \notin \mathrm{Pre}(u) \\ 0 & \varepsilon(v,u) = + \wedge v \notin \omega \\ \max_v & \varepsilon(v,u) = + \wedge v \in \omega \\ \max_v & \varepsilon(v,u) = - \wedge v \notin \omega \\ 0 & \varepsilon(v,u) = - \wedge v \in \omega \end{cases} \tag{2.1}$$

Then by construction, $\mathrm{Res}_u(x) = \omega$. □

Definition 2.17 (Isomorphic rows) [Lorenz, 2011] Two u-rows $\tau^u_x = (x^0, \ldots, x^{\max_u})$ and $\tau^u_y = (y^0, \ldots, y^{\max_u})$ are *isomorphic*, if for all $i \in \{0, \ldots, \max_u\}$ we have $\mathrm{Res}_u(x^i) = \mathrm{Res}_u(y^i)$, $\delta(u, x^i) = \delta(u, y^i)$.

Lemma 2.18 [Lorenz, 2011] Let $u \neq v \in V$ and $x \in X$ with $x_v = x_u = 0$. For every $j \in \{0, \ldots, \max_v\}$, let τ^u_j be the u-row with starting state $x^0_j = x^0 + j\mathbf{e}_v$.

- If $(v,u) \notin E$, all u-rows $\tau^u_0, \ldots, \tau^u_{\max_v}$ are isomorphic to each other.
- If $(v,u) \in E$, then there are two groups of isomorphic u-rows. More precisely, the u-rows $\tau^u_0, \ldots, \tau^u_{\vartheta(v,u)-1}$ are isomorphic to each other and the u-rows $\tau^u_{\vartheta(v,u)}, \ldots, \tau^u_{\max_v}$ are isomorphic to each other.

Proof. According to Definition 2.8, the state transition vector of a u-row $\tau^u = (x^0, \ldots, x^{\max_u})$ is $\delta(u, \tau^u) = (\delta(u, x^0), \ldots, \delta(u, x^{\max_u}))$, where $\delta(u, x^i) = \mathrm{sgn}(K(u, \mathrm{Res}_u(x^i)) - i)$, $i \in \{0, \ldots, \max_u\}$ and $\mathrm{sgn} \in \{+1, 0, -1\}$.

If $(v,u) \notin E$, then for all $i \in \{0, \ldots, \max_u\}$ we have $\mathrm{Res}_u(x^i_0) = \ldots = \mathrm{Res}_u(x^i_{\max_v})$, so that $\mathrm{Res}_u(\tau^u_0) = \ldots = \mathrm{Res}_u(\tau^u_{\max_u})$. Thus $\delta(u, \tau^u_0) = \ldots = \delta(u, \tau^u_{\max_v})$, and all u-rows τ^u_0, $\ldots$, $\tau^u_{\max_v}$ are isomorphic to each other.

If $(v,u) \in E$, let $\vartheta(v,u) = t \in \{1, \ldots, \max_u\}$. For all $i \in \{0, \ldots, \max_u\}$, we have $\mathrm{Res}_u(x^i_0) = \ldots = \mathrm{Res}_u(x^i_{t-1})$ and $\mathrm{Res}_u(x^i_t) = \ldots = \mathrm{Res}_u(x^i_{\max_v})$. Thus $\delta(u, \tau^u_0) = \ldots = \delta(u, \tau^u_{t-1})$ and $\delta(u, \tau^u_t) = \ldots = \delta(u, \tau^u_{\max_u})$. Therefore, the u-rows $\tau^u_0, \cdots, \tau^u_{t-1}$ are isomorphic to each other, and the u-rows $\tau^u_t, \ldots, \tau^u_{\max_v}$ are also isomorphic to each other. □

The next Lemma 2.19 shows two u-rows are isomorphic if they contain some states for which the resources of u only differ in $\{u\}$ itself.

Lemma 2.19 [Lorenz et al., 2013] Let $x, y \in X$ such that there exists a component $u \in V$ with $Res_u(x)\backslash u = Res_u(y)\backslash u$. Then the u-row containing x is isomorphic to the u-row containing y.

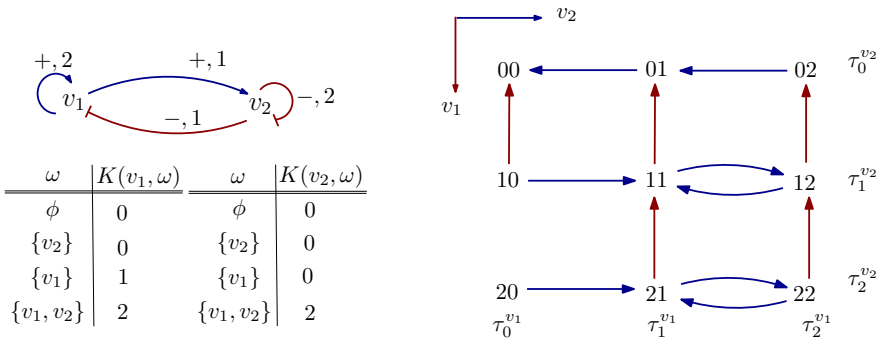
The following main theorem from [Lorenz, 2011, Lorenz et al., 2013] shows that an ASTG can be uniquely characterised by its extremal rows and the corresponding IG.

Theorem 2.20 [Lorenz, 2011, Lorenz et al., 2013] For any model $M = (I, K)$, the state transition graph T_M is uniquely determined by I and the extremal rows of T_M.

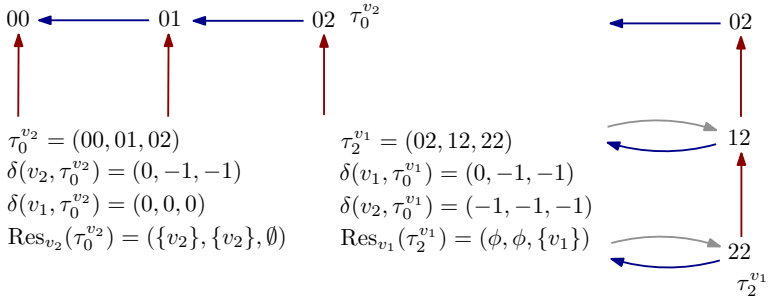
Proof. For any u-row $(x^0, \cdots, x^{\max_u})$, one can always find an extremal state y with $\mathrm{Res}_u(x^0) = \mathrm{Res}_u(y)$, according to Lemma 2.16. By Lemma 2.19, the extremal row including y is isomorphic to the u-row $(x^0, \cdots, x^{\max_u})$. □

Example 2.21 Figure 2.2a shows a model $M = (I, K)$ and its ASTG T_M. The definitions of a u-row 2.7 and the state transitions of a u-row 2.8 are illustrated in Figure 2.2b. See the v_2-row $\tau_0^{v_2}$ and v_1-row $\tau_2^{v_1}$.

The v_1- and v_2-rows of T_M also verify Corollary 2.12. v_1 has a self-activation in I, and v_1-row $\tau_0^{v_1}$ is of *pos* type. Similarly, v_2 has a self-inhibition, and v_2-row $\tau_2^{v_2}$ is of *neg* type.



(a) A model M and its ASTG T_M.



(b) $\tau_0^{v_2}$ and $\tau_2^{v_1}$.

Figure 2.2: (a) A model $M = (I, M)$ and its ASTG T_M with v_1-rows: $\tau_0^{v_1}$, $\tau_1^{v_1}$, $\tau_2^{v_1}$, and v_2-rows: $\tau_0^{v_2}$, $\tau_1^{v_2}$, $\tau_2^{v_2}$. (b) shows for $\tau_0^{v_2}$ and $\tau_2^{v_1}$, the state transition vectors in direction of v_1 and v_2, the resource vectors of them.

Lemma 2.18 can be illustrated on the edges (v_2, v_1) and (v_1, v_2). For $(v_2, v_1) \in E$, the v_1-row $\tau_0^{v_1}$ is different from $\tau_1^{v_1}$ and $\tau_2^{v_1}$ which are isomorphic to each other. For $(v_1, v_2) \in E$, there are also two isomorphic groups of v_2-rows: $\{\tau_0^{v_2}\}$ and $\{\tau_1^{v_2}, \tau_2^{v_2}\}$.

To illustrate Theorem 2.20, we take the same IG and the extremal rows of the ASTG from Figure 2.2.

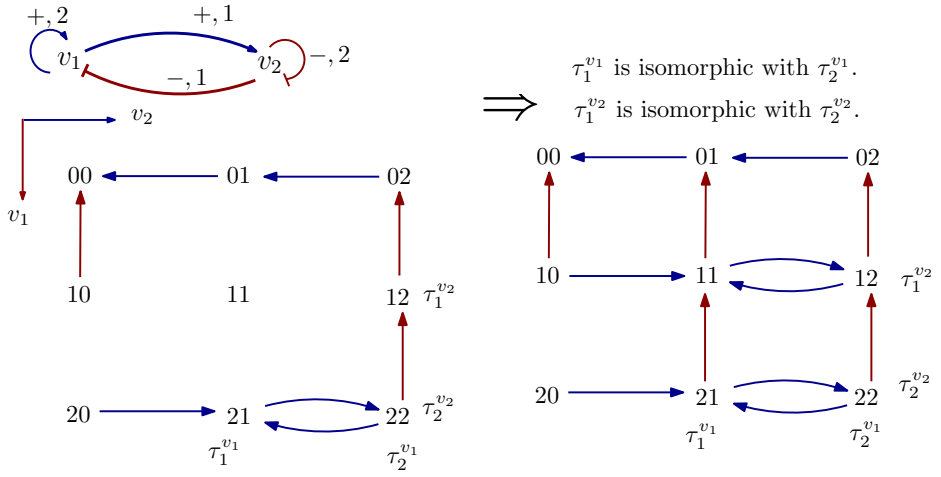


Figure 2.3: An IG I and the extremal rows of the ASTG uniquely determine the complete ASTG. $\vartheta(v_1, v_2) = 1$ implies that $\tau_1^{v_2}$ is isomorphic to $\tau_2^{v_2}$. Similarly, $\vartheta(v_2, v_1) = 1$ implies that, $\tau_1^{v_1}$ is isomorphic to $\tau_2^{v_1}$.

△

2.1.3 Equivalent models

A model $M = (I, K)$ may contain unnecessary interactions that are neither visible nor observable. Therefore, for a given ASTG T, there may be many models that generate T. Even two models with the same necessary interactions that differ only in the logical parameter function can still generate the same ASTG. We now further explore this issue.

Definition 2.22 (Isomorphic models) Let $M_1 = (I, K_1)$ and $M_2 = (I, K_2)$ be two models with the same IG $I = (V, E, \varepsilon, \vartheta, \max)$. M_1 and M_2 are *isomorphic* if

$$\delta_{M^1}(u, x) = \delta_{M^2}(u, x), \forall x \in X,\ \forall u \in V.$$

Note that two isomorphic models M_1 and M_2 have the same state space X and identical ASTGs, because the state transition functions $\delta(\cdot, \cdot)$ are the same.

Definition 2.23 (ω-side of u) [Lorenz, 2011]

- Given $a, b, t \in \mathbb{N}$, we say that a and b lie on *different sides* of t if $a < t \leq b$ or $b < t \leq a$.
- Given $u \in V$, $\omega \subseteq \mathrm{Pre}(u)$, we say that $a \in \{0, \ldots, \max_u\}$ lies *on the ω-side of* u if there exists a state $x \in X$, with $x_u = a$ and $\mathrm{Res}_u(x) = \omega$.

For illustration, we consider in Figure 2.4 the row types *pos* and *neg* from Proposition 2.10. We will look whether $K(u, \mathrm{Res}_u(x))$ lies on $\mathrm{Res}_u(x)$-side of u.

(a) *Pos* type

$\text{Res}_u(x^a) = \omega$ $\quad$ $K(u,\omega) = a$

$\text{Res}_u(x^b) = \zeta$ $\quad$ $K(u,\zeta) = b$

(b) *Neg* type

$\text{Res}_u(x^{t-1}) = \omega$ $\quad$ $K(u,\omega) > t-1$

$\text{Res}_u(x^t) = \zeta$ $\quad$ $K(u,\zeta) < t$

Figure 2.4: Two u-rows τ^u. (a) *Pos* type, $\text{Res}_u(x^a) = \omega$, $K(u,\omega) = a$ lies on ω-side of u; $\text{Res}_u(x^b) = \varsigma$, $K(u,\varsigma) = b$ lies on ς-side of u. (b) *Neg* type, for all x^i with $i \in \{0,\ldots,t-1\}$, $K(u,\text{Res}_u(x^i)) > t-1$, and for all x^i with $i \in \{t,\ldots,\max_u\}$, $K(u,\text{Res}_u(x^i)) < t$. $K(u,\text{Res}_u(x^i))$ does not lie on $\text{Res}_u(x)$-side of u for all states $x^i \in \tau^u$.

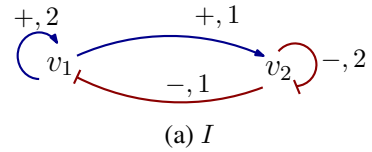
Definition 2.23 allows characterising the isomorphism of two models with the same interaction graph in terms of their logical parameters, as shown by the following theorem.

Theorem 2.24 [Lorenz, 2011] Two models $M^1 = (I, K^1)$ and $M^2 = (I, K^2)$ with the same interaction graph I are isomorphic if and only if, for all components $u \in V$ and all $\omega \subseteq \text{Pre}(u)$, the following is true:

- If $K^1(u,\omega)$ lies on the ω-side from u, then $K^2(u,\omega) = K^1(u,\omega)$.
- If $K^1(u,\omega)$ does not lie on the ω-side from u, neither does $K^2(u,\omega)$.

The following example shows that for a model $M = (I, K)$, the logical parameter function K may satisfy the Snoussi-condition, while the logical parameter function of an isomorphic model need not satisfy this condition.

Example 2.25 Figure 2.5 shows two isomorphic models $M = (I, K)$ and $M' = (I, K')$ generating the same ASTG T. While the logical parameter function K satisfies the Snoussi-condition, K' does not.

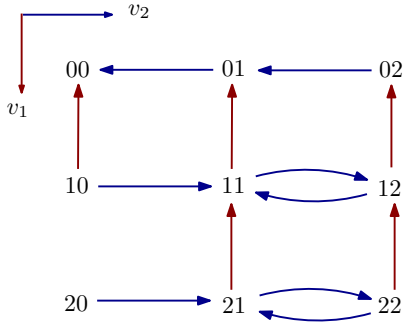


(a) I

ω	$K(v_1,\omega)$
$\emptyset$	0
$\{v_2\}$	0
$\{v_1\}$	1
$\{v_1,v_2\}$	2

ω	$K(v_2,\omega)$
$\emptyset$	0
$\{v_2\}$	0
$\{v_1\}$	0
$\{v_1,v_2\}$	2

(b) K



(c) T

ω	$K'(v_1,\omega)$
$\emptyset$	0
$\{v_2\}$	0
$\{v_1\}$	1
$\{v_1,v_2\}$	2

ω	$K'(v_2,\omega)$
$\emptyset$	1
$\{v_2\}$	0
$\{v_1\}$	0
$\{v_1,v_2\}$	2

(d) K'

Figure 2.5: (a) IG I. (b) Logical parameter function K. (c) Corresponding ASTG T. (d) Alternative logical parameter function K'. $M = (I, K)$ and $M' = (I, K')$ are two isomorphic models generating the same ASTG T where K satisfies the Snoussi-condition but K' does not.

To find a logical parameter function for an IG which can satisfy the Snoussi-condition as much as possible, the following lemma from [Lorenz, 2011] is used.

Lemma 2.26 [Lorenz, 2011] Given a model $M = (I, K)$, the logical parameter function K^S is defined in the following way for all $u \in V$ and $\omega \subseteq \mathrm{Pre}(u)$:

1. If $K(u,\omega)$ lies on the ω side from u, let

$$K^S(u,\omega) := K(u,\omega)$$

2. If $K(u,\omega)$ does not lie on the ω side from u, let

$$K^S(u,\omega) := \begin{cases} \vartheta(u,u) - 1 & \varepsilon(u,u) = +,\ u \in \omega \\ \vartheta(u,u) & \varepsilon(u,u) = +,\ u \notin \omega \\ 0 & \varepsilon(u,u) = -,\ u \notin \omega \\ \max_u & \varepsilon(u,u) = -,\ u \in \omega \end{cases}$$

If in $M^S = (I, K^S)$ there is a violation of the Snoussi-condition in component u for some $\omega \subseteq \varsigma \subseteq V$, then the same is true in all its *isomorphic* models.

K^S has the fewest violations of the Snoussi-condition among all models of the same isomorphism class. This Lemma has been proved in [Lorenz, 2011]. Here we recall the basic idea. The first case, $K^S(u,\omega)$ is the same as $K(u,\omega)$. In the second case of Lemma 2.26, if $K(u,\omega)$ does not lie on the ω side from u, then by Definition 2.23, for all $x \in X$ with $\mathrm{Res}_u(x) = \omega$, we have $K(u,\omega) \neq x_u$, and there are the following two cases:

1. In the case $\varepsilon(u,u) = +$, for those states $x \in X$ with $\mathrm{Res}_u(x) = \omega$, if $u \in \omega$, then $x_u \geq \vartheta(u,u)$ and $K(u,\omega) \geq \vartheta(u,u)$; if $u \notin \omega$, then $x_u < \vartheta(u,u)$ and $K(u,\omega) < \vartheta(u,u)$. In this case, $K^S(u,\omega)$ takes the closest value to $\vartheta(u,u)$ which still does not lie on the ω-side of u.

2. In the case of $\varepsilon(u,u) = -$, for those states $x \in X$ with $\mathrm{Res}_u(x) = \omega$, if $u \in \omega$, then $x_u < \vartheta(u,u)$ and $K(u,\omega) \geq \vartheta(u,u)$; if $u \notin \omega$, then $x_u \geq \vartheta(u,u)$ and $K(u,\omega) < \vartheta(u,u)$. In this case, $K^S(u,\omega)$ takes the furthest value to $\vartheta(u,u)$ so that $K^S(u,\omega)$ does not lie on ω-side from u.

Thus the parameters $K^S(u,\omega)$ are chosen in such a way that they always remain on the same side of the threshold values as $K(u,\omega)$. By Theorem 2.24, this implies the isomorphism of M and M^S. It follows from Lemma 2.26 that $M^S = (I, K^S)$ has the minimal number of violations of the Snoussi-condition in all models of the *isomorphism class*. This lemma gives an intuition on how to build a model that satisfies the Snoussi-condition as much as possible.

Corollary 2.27 Given a model $M = (I, K)$ the following is true.

- If there exists a violation of the Snoussi-condition in an interaction $(u, v) \in V$, then there is also a violation in the component u.
- If there exists a violation of the Snoussi-condition in a component $u \in V$, then there exists *at least one* violation in an interaction $(v, u) \in E$.

A Snoussi-condition violation in an interaction is itself a Snoussi-condition violation in a component, but a Snoussi-condition violation in a component can mean more than one Snoussi-condition violations in an interaction.

Proof. 1. If there exists a Snoussi-condition violation in an interaction $(v, u) \in V$, *i.e.*, there exists $\omega \subseteq V \backslash \{u\}$, $K(u,\omega) > K(u, \omega \cup \{u\})$, then, let $\zeta = \omega \cup \{u\}$, for $\omega \subseteq \zeta \subset \mathrm{Pre}(u)$, $K(u,\omega) > K(u,\zeta)$, so that there is a Snoussi-condition violation in u at the same place.

2. If there exists a Snoussi-condition violation in a component $u \in V$, *i.e.*, there exists $\omega \subseteq \zeta \subseteq \mathrm{Pre}(u)$, $K(u,\omega) > K(u,\zeta)$, then there must exist at least one component $v \in \zeta \backslash \omega$, such that $K(u,\omega) > K(u,\zeta) \geq K(u,\omega \cup \{v\})$, so that $K(u,\omega) > K(u,\omega \cup \{v\})$. Now it is easy to see that there exists *at least one* Snoussi-condition violation in an interaction $(v,u) \in E$.

□

In the next definition we generalise the notion of isomorphic models.

Definition 2.28 (Equivalent models) Let $M_1 = (I_1, K_1)$ and $M_2 = (I_2, K_2)$ be two models, where $I_1 = (V, E_1, \varepsilon_1, \vartheta_1, max)$ and $I_2 = (V, E_2, \varepsilon_2, \vartheta_2, max)$. M_1 and M_2 are called *equivalent* if

$$\delta_{M^1}(u,x) = \delta_{M^2}(u,x), \forall x \in X,\ \forall u \in V.$$

The following lemma from [Lorenz, 2011] allows constructing equivalent models.

Lemma 2.29 [Lorenz, 2011] Given a model $M^1 = (I^1, K^1)$ with $I^1 = (V, E, \varepsilon^1, \vartheta^1, \max)$.

(a) For IG $I^2 = (V, E, \varepsilon^2, \vartheta, \max)$, a logical parameter function K^2 is defined for all $u \in V$ and $\omega \in \mathrm{Pre}(u)$ in I^1:

$$K^2(u,\omega) = K^1(u,\omega') \quad \omega' := \omega \Delta \{v \in \mathrm{Pre}(u) \mid \varepsilon^1(v,u) \neq \varepsilon^2(v,u)\}$$

Then the model $M^2 = (I^2, K^2)$ defines the same ASTG as M^1.

(b) For an IG $I^3 = (V, E^3 = E \sqcup E', \varepsilon^3, \vartheta^3, \max)$, where "$\sqcup$" is a disjoint union, E' is a set of edges and $E \cap E' = \emptyset$. For all $u \in V$, one can define its predecessors in the following way:

$$\begin{aligned} \mathrm{Pre}_E(u) &:= \{v \in V \mid (v,u) \in E\} \\ \mathrm{Pre}_{E'}(u) &:= \{v \in V \mid (v,u) \in E'\} \\ \mathrm{Pre}_{E^3}(u) &:= \mathrm{Pre}_E(u) \cup \mathrm{Pre}_{E'}(u) \end{aligned}$$

If $\vartheta^3|_E \equiv \vartheta$, one can define K^3 as follows:

$$\forall u \in V\ \ \forall \omega \subseteq \mathrm{Pre}_{E^3}(u):\ \omega' := \omega \Delta \{v \in \mathrm{Pre}_E \mid \varepsilon^1(v,u) \neq \varepsilon^3(v,u)\}$$

$$K^3(u,\omega) := K^1(u,\omega')$$

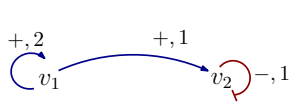
Then $M^3 := (I^3, K^3)$ defines the same ASTG as M^1. Moreover, K^3 is chosen such that no edges in E' is visible.

M^1, M^2 and M^3 are equivalent according to Definition 2.28. Lemma 2.29 will be the foundation of Algorithm 2.2.3. Intuitively, the lemma shows that:

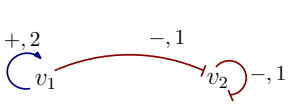
1. For a given model $M^1 = (I^1, K^1)$, one can construct an equivalent model $M^2 = (I^2, K^2)$, where I^2 differs from I^1 only in the signs of the interactions and K^2 can be constructed from K^1 according to Lemma 2.29 (a).

2. For a given model $M^1 = (I^1, K^1)$, one can construct an equivalent model $M^3 = (I^3, K^3)$ where I^3 includes all edges of I^1 and another set of invisible edges E'. K^3 can be constructed from K^1 according to Lemma 2.29 (b).

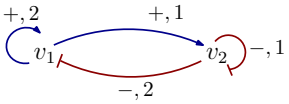
Example 2.30 Figure 2.6 shows three equivalent models M^1, M^2 and M^3, which illustrate Lemma 2.29. One can see that I^2 differs from I^1 in the interaction (v_1, v_2), and I^3 carries an unnecessary interaction (v_2, v_1). However, all the three models have the same ASTG, as shown in Figure 2.7.



(a) I^1.



(b) I^2.



(c) I^3.

ω	$K^1(v_1,\omega)$
ϕ	0
$\{v_1\}$	1

ω	$K^1(v_2,\omega)$
ϕ	0
$\{v_2\}$	1
$\{v_1\}$	2
$\{v_1,v_2\}$	2

(d) K^1

ω	$K^2(v_1,\omega)$
ϕ	0
$\{v_1\}$	1

ω	$K^2(v_2,\omega)$
ϕ	2
$\{v_2\}$	2
$\{v_1\}$	0
$\{v_1,v_2\}$	1

(e) K^2

ω	$K^3(v_1,\omega)$
ϕ	0
$\{v_2\}$	0
$\{v_1\}$	1
$\{v_1,v_2\}$	1

ω	$K^3(v_2,\omega)$
ϕ	0
$\{v_2\}$	1
$\{v_1\}$	2
$\{v_1,v_2\}$	2

(f) K^3

Figure 2.6: (a), (d) $M^1 = (I^1, K^1)$, (b), (e) $M^2 = (I^2, K^2)$, (c), (f)$M^3 = (I^3, K^3)$.

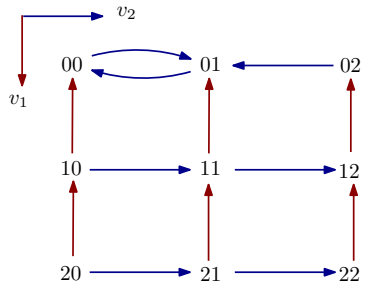


Figure 2.7: The ASTG T of the three models in Figure 2.6.

2.2 Reverse engineering algorithms

In this section, we present two reverse engineering algorithms *Visibility-Model* and *Observability-Snoussi-Model* together with the auxiliary algorithms *Logical-Parameters* and *Activation-Value* [Lorenz, 2011]. Each of them will be described in pseudocode and illustrated with an example.

2.2.1 Algorithm *Logical-Parameters*

The algorithm *Logical-Parameters* is based on Lemma 2.16 and Proposition 2.10. This algorithm requires as input an IG and a suitable ASTG . It returns a logical parameter function K for the IG, so that the IG and K can regenerate the input ASTG T. The algorithm does not use the threshold values of the IG. The algorithm was designed for so-called *simple* IGs $\tilde{I} = (V, E, \varepsilon, \max)$ containing no information on the threshold values. The output K and $\tilde{I}$ can only generate the transitions from all extremal states in T. Nevertheless, after adding suitable threshold values ϑ to $\tilde{I}$, one can regenerate the full ASTG T. The pseudocode is given in Algorithm 2.1 and

illustrated by Example 2.31. Finally, the complexity of the algorithm will be analysed.

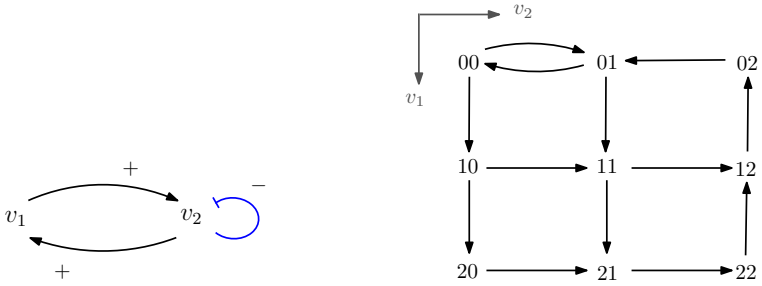
Algorithm 2.1: Logical-Parameters

Input: A simple IG $\tilde{I} = (V, E, \varepsilon, \max)$, an ASTG $T = (X, S)$. /* S is given by $\delta : V \times X \to \{-1, 0, +1\}$. */

Output: A logical parameter function K, such that $\tilde{I}$ and K can define the outgoing transitions of all extremal states in T.

1 **foreach** $u \in V$ **do**
2 **if** $(u, u) \notin E$ **then**
3 **foreach** $\omega \subseteq \mathrm{Pre}(u)$ **do**
4 Construct an extremal state $x \in X : \mathrm{Res}_u(x) = \omega$ /* Lemma 2.16 */
 $(x^0, \ldots, x^{\max_u}) := \tau^u$ /* $x \in \tau^u$ */;
5 $\exists! \, a \in \{0, \ldots, \max_u\} : \delta(u, x^a) = 0$ /* $\exists!a$: exists a unique a */;
6 $K(u, \omega) := a$;
7 **else**
8 **foreach** $\omega \subseteq \mathrm{Pre}(u) \backslash \{u\}$ **do**
9 Construct an extremal state $x \in X : \mathrm{Res}_u(x) = \omega$;
10 $(x^0, \ldots, x^{\max_u}) := \tau^u$ /* $x \in \tau^u$ */;
11 **if** $\exists! \, a < b \in \{0, \ldots, \max_u\} : \delta(u, x^a) = \delta(u, x^b) = 0$ **then** /* pos type */
12 $K(u, \mathrm{Res}_u(x^0)) := a$;
13 $K(u, \mathrm{Res}_u(x^{\max_u})) := b$;
14 **else if** $\exists! \, a \in \{0, \ldots, \max_u\} : \delta(u, x^a) = 0$ **then** /* open type */
15 $K(u, \mathrm{Res}_u(x^0)) := a$;
16 $K(u, \mathrm{Res}_u(x^{\max_u})) := a$;
17 **else** /* neg type */
18 $K(u, \mathrm{Res}_u(x^0)) := \max_u$;
19 $K(u, \mathrm{Res}_u(x^{\max_u})) := 0$;
20 **return** K

Illustration



(a) A simplified IG $\tilde{I}$ (b) An ASTG T

Figure 2.8: Inputs for algorithm *Logical-Parameters*. (a) A simple IG $\tilde{I} = (V, \tilde{E}, \tilde{\varepsilon}, \max)$. Standard arrows denote activations, and blunt arrows inhibitions. $\max = [\max_{v_1}, \max_{v_2}] = [2, 2]$. (b) ASTG $T = (X, S)$, where $X = \{0, 1, 2\}^2$ and each state $x = (x_{v_1}, x_{v_2})$.

Example 2.31 Figure 2.8 gives an ASTG T and a suitable simple IG $\tilde{I}$ as the input for algorithm *Logical-Parameters*. The execution is discussed below.

v_1: $(v_1, v_1) \notin \tilde{E}$, line 2–6 in Algorithm 2.2.1. For every resource of v_1 in $\tilde{I}$, $\omega \subseteq \mathrm{Pre}(v_1) = \{v_2\}$, an extremal state is constructed according to Lemma 2.16. Then the v_1-row including this state is found out to assign values for $K(v_1, \omega)$.

1. A resource of v_1, $\omega = \emptyset$.

   - Construct the extremal state x such that $\mathrm{Res}_{v_1}(x) = \emptyset$, according to Lemma 2.16. Because $v_1 \notin \mathrm{Pre}(v_1)$, $\tilde{\varepsilon}(v_2, v_1) = +$ and $v_1, v_2 \notin \omega$, $x = 00$. The extremal row including x is $\tau^{v_1} = (00, 10, 20)$.

   - $\delta(v_1, \tau^{v_1}) = (1, 1, 0)$, $\delta(v_1, 20) = 0$ and τ^{v_1} is of *open* type. Therefore, $K(v_1, \emptyset) := 2$.

2. A resource of v_1, $\omega = \{v_2\}$.

   - Construct the extremal state x such that $\mathrm{Res}_{v_1}(x) = \{v_2\}$, according to Lemma 2.16. Because $\tilde{\varepsilon}(v_2, v_1) = +$ and $v_2 \in \omega$, $x = 02$. The extremal row including x is $\tau^{v_1} = (02, 12, 22)$.

   - $\delta(v_1, \tau^{v_1}) = (0, -1, -1)$, $\delta(v_1, 12) = 0$ and τ^{v_1} is of *open* type. Therefore, $K(v_1, \{v_2\}) := 0$.

The logical parameters for v_1 are:

ω	x	τ^{v_1}	$\delta(v_1, \tau^{v_1})$	row type	$K(v_1, \omega)$
$\emptyset$	00	(**00**, 10, 20)	$(1, 1, 0)$	*open*	2
$\{v_2\}$	02	(**02**, 12, 22)	$(0, -1, -1)$	*open*	0

v_2: $(v_2, v_2) \in \tilde{E}$, line 7–19 in Algorithm 2.2.1. For every resource of v_2 in $\tilde{I}$, $\omega \subseteq \mathrm{Pre}(v_2) \backslash \{v_2\} = \{v_1\}$, an extremal state is constructed according to Lemma 2.16. Then the v_2-row including this state is found out to assign values for $K(v_2, \omega)$.

1. A resource of v_2, $\omega = \emptyset$.

   - Construct the extremal state x such that $\mathrm{Res}_{v_2}(x) = \emptyset$, according to Lemma 2.16. Because $\tilde{\varepsilon}(v_1, v_2) = +$, $\tilde{\varepsilon}(v_2, v_2) = -$ and $v_1, v_2 \notin \omega$, $x = 02$. The v_2-row including x is $\tau^{v_2} = (00, 01, 02)$.

   - $\delta(v_2, \tau^{v_2}) = (1, -1, -1)$, where $0 \notin \delta(v_2, \tau^{v_2})$. τ^{v_2} is of *neg* type. So that, $K(v_2, \emptyset) := 1$ and $K(v_2, \{v_2\}) := 0$.

2. A resource of v_2, $\omega = \{v_1\}$.

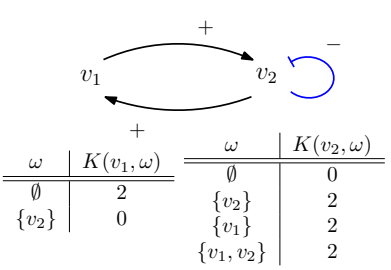
   - Construct the extremal state x such that $\mathrm{Res}_{v_2}(x) = \emptyset$, according to Lemma 2.16. Because $\tilde{\varepsilon}(v_1, v_2) = +$, $\tilde{\varepsilon}(v_2, v_2) = -$ and $v_1 \in \omega$, $v_2 \notin \omega$, $x = 22$. The v_2-row including x is $\tau^{v_2} = (20, 21, 22)$.

   - $\delta(v_2, \tau^{v_2}) = (1, 1, 0)$, $\delta(v_2, 22) = 0$ and τ^{v_2} is of *open* type. So that, $K(v_2, \{v_1\}) := 2$ and $K(v_2, \{v_1, v_2\}) := 2$.

The logical parameters for v_2 are:

ω	x	τ^{v_2}	$\delta(v_2, \tau^{v_2})$	row type	$K(v_2, \omega)$	$\omega \cup \{v_2\}$	$K(v_2, \omega \cup \{v_2\})$
$\emptyset$	02	(**00**, 01, 02)	$(1, -1, -1)$	*open*	1	$\{v_2\}$	0
$\{v_1\}$	22	(20, 21, **22**)	$(1, 1, 0)$	*pos*	2	$\{v_1, v_2\}$	2

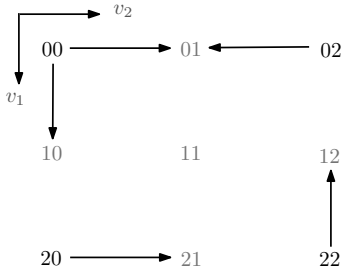
The logical parameter function K is obtained after visited each component. Because $\tilde{I}$ has no information about the threshold, the resource of a component under a non-extremal state is unclear. Therefore, $\tilde{I}$ together with K can only regenerate the outgoing transitions of the extremal states, see the output in Figure 2.9.



ω	$K(v_1,\omega)$
$\emptyset$	2
$\{v_2\}$	0

ω	$K(v_2,\omega)$
$\emptyset$	0
$\{v_2\}$	2
$\{v_1\}$	2
$\{v_1,v_2\}$	2

(a) Input $\tilde{I}$ and output K.



(b) Transitions from extremal states.

Figure 2.9: Example 2.31 for algorithm *Logical-Parameters*

Complexity Next we will discuss the complexity of the algorithm.

The space complexity of the algorithm can be checked from how the input and output are stored. Let $N = |V| \in \mathbb{N}^+$ be the number of the components in the system. The number of edges in an IG is at most N^2. Every component can have at most an activity level N, thus the size of the state space X is at most $(N+1)^N$. The number of edges in an ASTG is at most $|X| \times N = N(N+1)^N$. Altogether, the input size in terms of space is a sum of an IG and an ASTG, and we get the space complexity $S(N) = N^2 + N(N+1)^N = \mathcal{O}(N^{N+1})$.

The time complexity of the algorithm also depends on N. Starting from line 1, the main loop will be repeated for every component u, *i.e.*,in total $\mathcal{O}(N)$ times.

1. If u has no self loop, we go through lines 3–6. Lines 4–6 will be repeated for all resources of u, *i.e.*, at most 2^{N-1} times.

   (a) line 4: the cost of constructing an extremal state is N, by choosing a value for each component according to Lemma 2.16. Finding the u-row that includes the constructed state can be done by assigning the u-th component a value from 0 until $\max_u$, which requires $N+1$steps.

   (b) line 5 includes finding the only state which has state transition 0, which costs also at most $N+1$ steps.

   (c) line 6, assigning $K(u,\omega)$ is only one step.

   In total we get: $2^{N-1} \times (N + (N+1) + (N+1) + 1) = 3(N+1)2^{N-1}$.

2. If u has a self loop, we go through lines 8–19. Lines 9–19 are repeated for at most 2^{N-1} times. The cost of line 9, 10 and 11 (or 14), which can be calculated similarly, are N, $N+1$ and $N+1$, respectively. Assigning $K(u,\omega)$ and $K(u,\omega \cup \{u\})$ requires two steps, *i.e.*, 2. The sum is: $2^{N-1} \times (N + (N+1) + (N+1) + 2) = 3(N+1)2^{N-1}$, which equals to that from the previous step.

In total, the running time of the algorithm is $T(N) = N \times 3(N+1)2^{N-1} = 3(N^2 + N)2^{N-1} = \mathcal{O}(N^2 2^N)$.

2.2.2 Algorithm *Activity-Value*

Algorithm *Activation-Value* can find the threshold value of an interaction (u,v) from given state transitions in direction of v. It needs the following input: an edge (u,v), a resource ω with

$v \notin \omega$ and $K(v,\omega) \neq K(v,\omega \cup \{u\})$, and $\delta(v,\cdot)$. Algorithm *Activation-Value* is called within the two reverse engineering algorithms, where the input interaction is always positive.

1. For v and the resource ω, an extremal state x is constructed according to Lemma 2.16. Accordingly, the v-row containing x is directly known.

   (a) If $u = v$, the edge is a self-loop. According to Proposition 2.10, τ^v is either of type *pos* or *neg*and $\vartheta(v,v)$ can be obtained from the structure of τ^v.

   (b) Otherwise, find the first l where $\tau^v \ncong \tau^v_{x+l\cdot\mathbf{e}_u}$, and set $\vartheta(u,v) := l$. According to Lemma 2.18, the position from which the row structures start to change is the threshold value. τ^v is constructed from x with $\mathrm{Res}_v(x) = \omega$ and $u \notin \omega$. Therefore, we have always $x_u = 0 < y_u$, for x in τ^v and y in $\tau^v_{x+l\cdot\mathbf{e}_u}$.

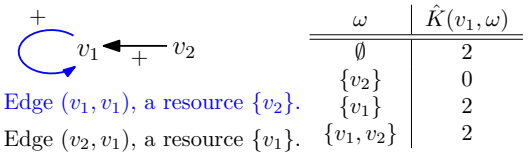
The details of this algorithm are given in the pseudocode below and illustrated by Example 2.32.

Algorithm 2.2: Activity-Value

Input: $u, v \in V, \omega \subseteq V\backslash\{u\}, \delta(v,\cdot)$ /* $K(v,\omega) \neq K(v,\omega\cup\{u\})$ */
Output: $\vartheta(u,v)$
Construct an extremal state $x \in X : \mathrm{Res}_v(x) = \omega$;
$(x^0,\ldots,x^{\max_v}) := \tau^v$; /* $x \in \tau^v$ */
if $u = v$ **then**
  if τ^v *is of type* pos **then**
    $\exists!\, i < \max_v : \delta(v,x^i) \leq 0 \wedge \delta(v,x^{i+1}) \geq 0$ /* $\exists!$: there is a unique */
  else /* is of type *neg* */
    $\exists!\, i < \max_v : \delta(v,x^i) = 1 \wedge \delta(v,x^{i+1}) = -1$;
  $\vartheta(u,v) := i + 1$;
else /* $u \neq v$ */
  $\vartheta(u,v) := \min\{l \mid \tau^v \ncong \tau^v_{x+l\cdot\mathbf{e}_u}\}$;
return $\vartheta(u,v)$

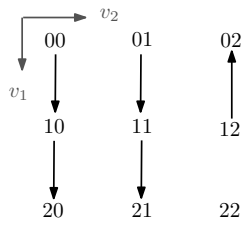
Illustration

Example 2.32 Algorithm *Activity-Value* is applied to construct the threshold values of two interactions from Example 2.31. The input is shown in Figure 2.10.



ω	$\hat{K}(v_1,\omega)$
$\emptyset$	2
$\{v_2\}$	0
$\{v_1\}$	2
$\{v_1,v_2\}$	2

(a) Edges on v_1 and $\hat{K}(v_1,\omega)$



(b) $\delta(v_1,\cdot)$

Figure 2.10: Two inputs for algorithm *Activity-Value*. (a) Two incoming edges of v_1. For (v_1,v_1), on resource $\{v_2\}$ we have $K(v_1,\{v_2\}) \neq K(v_1,\{v_1,v_2\})$. For (v_2,v_1), on resource $\emptyset$ we have $K(v_1,\emptyset) \neq K(v_1,\{v_2\})$. (b) The state transitions $\delta(v_1,\cdot)$ in direction of v_1.

1. Edge (v_1,v_1), resource $\{v_2\}$ with $K(v_1,\{v_2\}) \neq K(v_1,\{v_1,v_2\})$.
   Algorithm *Activity-Value*(v_1, v_1, $\{v_2\}$, $\delta(v_1,\cdot)$):

   (a) Construct an extremal state $x = 02$ with $\mathrm{Res}_{v_1}(x) = \{v_2\}$ according to Lemma 2.16. The v_1-row containing 02 is $(01, 12, 22)$.

(b) (v_1, v_1) is a self loop, follow line 3–8 in the pseudocode.

- $\delta(v_1, \tau^{v_1}) = (1, 0, 0)$, τ^{v_1} is of type *pos*. There is a unique position 1, where $\delta(u, x^1) = 0 \leq 0$ and $\delta(u, x^2) = 0 \geq 0$.
- $\vartheta(v_1, v_1) := 2$.

2. Edge (v_2, v_1), resource $\emptyset$ with $K(v_1, \emptyset) \neq K(v_1, \{v_2\})$.
Algorithm *Activity-Value*(v_2, v_1, $\{v_1\}$, $\delta(v_1, \cdot)$):

(a) Construct the extremal state $x = 20$ with $\mathrm{Res}_{v_1}(x) = \{v_1\}$ according to Lemma 2.16. The v_1-row containing 20 is $(00, 10, 20)$.

(b) (v_2, v_1) is not a self loop, follow line 10 in the pseudocode.

- $\delta(v_1, \tau^{v_1}) = (1, 1, 0)$.
  - ▷ let $l = 1$, $\tau_1^{v_1} = (01, 11, 21)$, $\delta(v_1, \tau_1^{v_1}) = (1, 1, 0)$, so that $\tau^{v_1} \cong \tau_1^{v_1}$.
  - ▷ let $l = 2$, $\tau_2^{v_1} = (02, 12, 22)$, $\delta(v_1, \tau_1^{v_1}) = (0, -1, 0)$, so that $\tau^{v_1} \cong \tau_2^{v_1}$.
- $l = 2$ is the smallest value that the structures of v_1-rows change. Therefore, $\vartheta(v_2, v_1) := 2$;

△

Complexity The space needed by this algorithm includes: the input including $\delta(v, \cdot)$ of size $|X| \times 1 = N^{N+1}$, an edge, a resource, a few v-rows as intermediate variables, and an output $\vartheta(u, v)$. Altogether, $S(N) = N^{N+1} + 2 + 2 + \max(N + 1, (N + 1)^2)$, which is at most $\mathcal{O}(N^{N+1})$.

The time complexity can be calculated from the pseudocode. For an input edge (u, v), the time for constructing an extremal state x is N, and finding the extremal row containing x τ^v is $N + 1$. Next the algorithm will choose one of the following two parts.

1. If $u = v$, lines 3–8 are chosen. The state transitions of τ^v are checked, with cost $N + 1$ (the length of τ^v); the threshold value is assigned. Therefore, this step costs $\mathcal{O}(N)$.
2. If $u \neq v$, line 10 is chosen. The worst case is that the row structures starting from τ^v until $\tau^v_{x_u = \max_u}$ need to be compared. Row τ^v is of length $(N + 1)$ and there are $\max_u$ other v-rows. Therefore, this step costs $\mathcal{O}(N^2)$.

The total time is $\mathcal{O}(N^2)$.

2.2.3 Algorithm *Visibility-Model*

Algorithm *Visibility-Model* is based on Lemma 2.29. For a given ASTG T, this algorithm constructs a model M which defines the same ASTG T and has minimal number of edges among all its isomorphic models.

A complete connected IG $\hat{I}$ is assumed for T with only positive edges, *i.e.*, an activation between every pair of components. Then algorithm *Logical-Parameters* is applied to obtain a pseudo logical parameter function $\hat{K}$ for $\hat{I}$. Next, all visible edges in $\hat{I}$ are kept in the real IG, assigned proper signs and threshold values (by algorithm *Activity-Value*). These visible interactions define a new IG. Finally, a proper logical parameter function for the new IG has to

be constructed, so that the interactions are visible and can regenerate the input ASTG T. Details of the algorithm can be found in the pseudocode and the illustration in Example 2.33.

Algorithm 2.3: Visibility-Model

Input: An ASTG $T = (X, S)$ with set of components V and maximal activity levels $\max(\cdot)$.
/* S is given by $\delta : V \times X \to \{-1, 0, +1\}$. */

Output: A model $M = (I, K)$ with ASTG$(M) \cong T$ which has minimal number of edges among its isomorphic model class. /* $\cong$, is isomorphic to */

1 $E := \emptyset$;
2 $\hat{E} := V \times V$;
3 $\hat{\varepsilon} :\equiv +$;
4 $\hat{I} := (V, \hat{E}, \hat{\varepsilon}, \max)$;
5 $\hat{K}(\cdot, \cdot) :=$ `Logical-Parameters(`$\hat{I}$, T`)`;
6 **foreach** $(u, v) \in \hat{E}$ **do**
7 **if** $\exists \omega \subseteq V \backslash \{u\} : \hat{K}(v, \omega) \neq \hat{K}(v, \omega \cup \{u\})$ **then**
8 $E := E \cup \{(u, v)\}$;
9 $\varepsilon(u, v) := \begin{cases} +, & \text{if } \hat{K}(v, \omega) < \hat{K}(v, \omega \cup \{u\}) \\ -, & \text{if } \hat{K}(v, \omega) > \hat{K}(v, \omega \cup \{u\}) \end{cases}$;
10 $\vartheta(u, v) :=$ `Activity-Value(`u, v, ω, $\delta(v, \cdot)$`)`;
11 $I := (V, E, \varepsilon, \vartheta, \max)$;
12 **foreach** $v \in V$ **do**
13 **foreach** $\omega \subseteq \mathrm{Pre}_I(v)$ **do**
14 $\varsigma := \{u \in \omega \mid \varepsilon(u, v) = +\} \cup \{u \in \mathrm{Pre}_I(v) \backslash \omega \mid \varepsilon(u, v) = -\}$;
15 $K(v, \omega) := \hat{K}(v, \varsigma)$;
16 **return** $M = (I, K)$

The main parts of the algorithm can be briefly explained in three steps.

1. *Initialisation*, line 1–5. Start from a simple IG $\hat{I}$ defined as a complete digraph with only positive edges and without any threshold values. Applying algorithm *Logical-Parameters* (Section 2.2.1), a suitable logical parameter function $\hat{K}$ is constructed for $\hat{I}$, called *pseudo* logical parameter function.

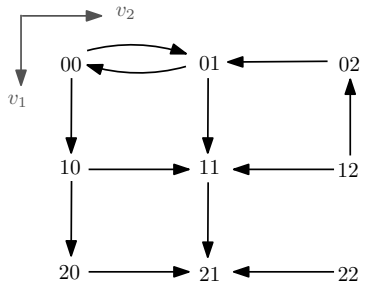
2. *Inferring* I, line 6–11. An edge $(u, v) \in \hat{E}$ is visible if one can find a resource $\omega \subset \mathrm{Pre}_{\hat{I}}(v) \backslash \{u\}$, with $\hat{K}(v, \omega) \neq \hat{K}(v, \omega \cup \{u\})$. If (u, v) is visible, then it is added to E. Furthermore, if $\hat{K}(v, \omega) < \hat{K}(v, \omega \cup \{u\})$, then $\varepsilon(u, v) := +$; if $\hat{K}(v, \omega) > \hat{K}(v, \omega \cup \{u\})$, then $\varepsilon(u, v) := -$. Next, taking ω and $\delta(v, \cdot)$ as the input for algorithm *Activity-Value*, a proper value $\vartheta(u, v)$ will be found. This inference is repeated for every edge of $\hat{I}$. After that, a new IG $I = (V, E, \varepsilon, \vartheta, \max)$ has been obtained containing only visible edges of $\hat{I}$.

3. *Assigning* K, line 12–15. A proper logical parameter function K has to be found for I, so that ASTG$(M) \cong T$ for $M = (I, K)$. For each component $v \in V$, for each state $x \in X$, let ω be the resource of v under x in I and let ς be the one in $\hat{I}$. Note that ς is just the symmetric difference of ω with a set of components for which the outgoing edges to v are negative. Therefore, K can be obtained by rearranging $\hat{K}$. M will generate the same ASTG as T.

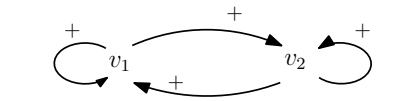
Illustration

Example 2.33 Examples for algorithm *Visibility-Model*
Figure 2.11a shows an ASTG as the input for algorithm *Visibility-Model*.
Let $V = \{v_1, v_2\}$, $\max = [2, 2]$.

1. *Initialisation.* $\hat{I} = (V, \hat{E}, \hat{\varepsilon}, \max)$ is constructed and algorithm *Logical-Parameters* is called to obtain a pseudo logical parameter function $\hat{K}$, see Figure 2.11b.



(a) ASTG T



ω	$\hat{K}(v_1,\omega)$	ω	$\hat{K}(v_2,\omega)$
$\emptyset$	2	$\emptyset$	2
$\{v_2\}$	0	$\{v_2\}$	0
$\{v_1\}$	2	$\{v_1\}$	1
$\{v_1, v_2\}$	2	$\{v_1, v_2\}$	1

(b) $\hat{I}$ and $\hat{K}$

Figure 2.11: (a) ASTG T. (b) A simple IG $\hat{I}$ and the pseudo logical parameter function $\hat{K}$.

2. *Inferring I.* The visibility of every interaction in $\hat{I}$ is checked. For the visible ones, signs are assigned and the threshold values are obtained by applying algorithm *Activity-Value*. For the 4 edges in the example, the details are listed below.

   (a) (v_1, v_1), a resource $\{v_2\}$ is found, where $\hat{K}(v_1, \{v_2\}) = 0 < \hat{K}(v_1, \{v_1, v_2\}) = 2$,
   $\rightsquigarrow E := E \cup \{(v_1, v_1)\}$, $\varepsilon(v_1, v_1) = +$,
   Activity-Value$((v_1, v_1), \{v_2\}, \delta(v_1, \cdot)) \rightsquigarrow \vartheta(v_1, v_1) = 2$.

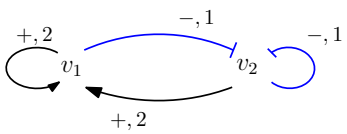
   (b) (v_2, v_1), a resource $\emptyset$ is found, where $\hat{K}(v_1, \emptyset) = 2 > \hat{K}(v_1, \{v_2\}) = 0$,
   $\rightsquigarrow E := E \cup \{(v_2, v_1)\}$, $\varepsilon(v_2, v_1) = -$,
   Activity-Value$((v_2, v_1), \emptyset, \delta(v_1, \cdot)) \rightsquigarrow \vartheta(v_2, v_1) = 2$.

   (c) (v_1, v_2), a resource $\emptyset$ is found, where $\hat{K}(v_2, \emptyset) = 2 > \hat{K}(v_2, \{v_1\}) = 1$,
   $\rightsquigarrow E := E \cup \{(v_1, v_2)\}$, $\varepsilon(v_1, v_2) = -$,
   Activity-Value$((v_1, v_2), \emptyset, \delta(v_2, \cdot)) \rightsquigarrow \vartheta(v_1, v_2) = 1$.

   (d) $(v_2, v_2) \in \hat{E}$, $\hat{K}(v_2, \emptyset) = 2 > \hat{K}(v_2, \{v_2\}) = 0$,
   $\rightsquigarrow E := E \cup \{(v_2, v_2)\}$, $\varepsilon(v_2, v_2) = -$,
   Activity-Value$((v_2, v_2), \emptyset, \delta(v_2, \cdot)) \rightsquigarrow \vartheta(v_2, v_2) = 1$.

   Now I is constructed, see Figure 2.12a.

3. *Assigning K_I.* For all $u \in V$, for all $\omega \subseteq \mathrm{Pre}_I(u)$, $\varsigma = \{v \in \omega \mid \varepsilon(v, u) = +\} \cup \{v \notin \mathrm{Pre}_I(u) \backslash \omega \mid \varepsilon(v, u) = -\}$, $K(u, \omega) := \hat{K}(u, \varsigma)$, K is set by rearranging $\hat{K}$, as shown in Figure 2.12b.



(a) I.

ω	ς	$\hat{K}(v_1,\varsigma) = K(v_1,\omega)$
$\emptyset$	$\emptyset$	2
$\{v_2\}$	$\{v_2\}$	0
$\{v_1\}$	$\{v_1\}$	2
$\{v_1, v_2\}$	$\{v_1, v_2\}$	2
ω	ς	$\hat{K}(v_2,\varsigma) = K(v_2,\omega)$
$\emptyset$	$\{v_2\}$	1
$\{v_2\}$	$\emptyset$	1
$\{v_1\}$	$\{v_1, v_2\}$	0
$\{v_1, v_2\}$	$\{v_1\}$	2

(b) K.

Figure 2.12: (a) The real IG I. (b) Through ω and ς, K is rearranged from $\hat{K}$.

A model $M = (I, K)$ which carries only visible interactions is constructed for T. $\triangle$

Complexity To determine the space complexity, we need to calculate the size of the input and output, and the intermediate variables. Let $N = |V|$ be the number of components, the maximal activity of each is at most N. The input is an ASTG $T = (X, S)$, S is given by $\delta(\cdot, \cdot)$, of size $N^{N+1}N$. Within the algorithm, the space for $\hat{I}$ and $\hat{K}$ is needed, which are of size N^2 resp. $N2^N$. The output is an IG I and K, which are of the same size as $\hat{I}$ and $\hat{K}$. Therefore, the total space complexity is $(SN) = N^{N+1}N + 2(N^2 + N2^N) = N^{N+2} + 2N^2 + N2^{N+1}$, which is $\mathcal{O}(N^{N+1})$.

The time complexity can be obtained from the three steps in the algorithm.

1. *Initialisation*, line 1–5, where algorithm *Logical-Parameters* is called to get $\hat{K}$. The running time is $\mathcal{O}(N^2 2^N)$.
2. *Inferring* I, line 6–10, which checks the visibility of each edge in $\hat{I}$ and finds out the sign and threshold value for the visible edges.
   - For visibility checking, in the worst case all pairs of resources $\omega \subset V \backslash \{u\}$ and $\omega \cup \{u\}$ need to be checked, thus the cost is $\mathcal{O}(2^{N-1})$.
   - For getting the threshold value of an edge, the algorithm *Activity-Value* is called, which costs $\mathcal{O}(N)$ for a self loop, and $\mathcal{O}(N^2)$ otherwise.

   The sum of all costs in the worst case is: $N^2 \times (\mathcal{O}(2^{N-1}) + \mathcal{O}(N^2))$, which is $\mathcal{O}(N^2 2^N)$.
3. *Assigning* K, line 12–15. In the worst case, for all N components, the symmetric difference has to be calculated for all 2^N resources. Therefore, it costs $\mathcal{O}(N2^N)$.

Altogether, the time complexity of Algorithm 2.2.3 is $\mathcal{O}(N^2 2^N)$.

2.2.4 Algorithm *Observability-Snoussi-Model*

For a given ASTG, the algorithm *Observability-Snoussi-Model* constructs a model $M = (I, K)$ which satisfies the observability condition, and the Snoussi-condition as much as possible. The pseudocode is given in Algorithm 2.4 and illustrated by Example 2.33.

This algorithm includes three steps: *initialisation*, *inferring* I and *assigning* K. *Initialisation* and *assigning* K are exactly the same as in algorithm *Visibility-Model*. The main difference between these two reverse engineering algorithm lies in the second step: *inferring* I.

The basic ideas of the algorithm are explained below. An influence from u to v is *positive* if for some $\omega \subseteq \mathrm{Pre}(u) \backslash \{u\}$, $K(v, \omega) < K(v, \omega \cup \{u\})$, or *negative* if $K(v, \omega) > K(v, \omega \cup \{u\})$. In the pseudocode, lines 6–8 and 18–20 count the positive and negative influences.

1. *Initialisation*, line 1–4. This step is the same as in Pseudocode 2.3. A simple graph $\hat{I}$ is defined as a complete positive graph without any threshold values. A *pseudo* logical parameter function $\hat{K}$ for $\hat{I}$ is constructed by applying algorithm *Logical-Parameters*.
2. *Inferring* I, line 5–26. For each component $v \in V$:
   - *Self loop* (v, v): count the number of positive influences p and the number n of negative ones. If either p or n is larger than 0, then:
     - $E := E \cup \{(v, v)\}$, and $\varepsilon(v, v)$ is assigned $+$ if $p \geq n$, and $-$ otherwise.

Algorithm 2.4: Observability-Snoussi-Model

Input: ASTG $T = (X, S)$, the component set V and the maximal activity levels $\max(\cdot)$. /* S is given by $\delta : V \times X \to \{-1, 0, +1\}$. */

Output: Model $M = (I, K)$ with ASTG$(M) \cong T$, which satisfies the observability condition, and the Snoussi-condition as much as possible. /* $\cong$ means isomorphic. */

$E := \varnothing$;
$\hat{E} := V \times V$; $\hat{\varepsilon} :\equiv +$;
$\hat{I} := (V, \hat{E}, \hat{\varepsilon}, \max)$;
$\hat{K}(\cdot,\cdot) :=$ `Logical-Parameters(`$\hat{I}$, T`)`;
foreach $v \in V$ **do**
  $p := 0$; $n := 0$;
  foreach $\omega \subseteq V \backslash \{v\}$ **do**
    if $\hat{K}(v,\omega) < \hat{K}(v,\omega \cup \{v\})$ **then** $p := p + 1$ **else if** $\hat{K}(v,\omega) > \hat{K}(v,\omega \cup \{v\})$ **then** $n := n + 1$
  if $n + p > 0$ **then**
    $E := E \cup \{(v,v)\}$;
    if $p \geq n$ **then** $\varepsilon(v,v) := +$ **else if** $p < n$ **then** $\varepsilon(v,v) := -$;
    pick $\omega' \subseteq V \backslash \{v\} : \hat{K}(v,\omega') \neq \hat{K}(v,\omega' \cup \{v\})$;
    $\vartheta(v,v) :=$ `Activity-Value(`v, v, ω', $\delta(v,\cdot)$`)`;
    foreach $\hat{\omega} \subseteq V$ **do**
      if $\hat{K}(v,\hat{\omega})$ *does not lie on* $\hat{\omega}$*-side from* v **then**
        $$\hat{K}(v,\hat{\omega}) := \begin{cases} \vartheta(v,v) - 1, & \text{if } \varepsilon(v,v) = +, v \in \hat{\omega} \\ \vartheta(v,v), & \text{if } \varepsilon(v,v) = +, v \notin \hat{\omega} \\ 0, & \text{if } \varepsilon(v,v) = -, v \in \hat{\omega} \\ \max_v, & \text{if } \varepsilon(v,v) = -, v \notin \hat{\omega} \end{cases} ; \text{ /* } \varepsilon \in I, \hat{\omega} \subseteq \mathrm{Pre}_{\hat{I}}(u) \text{.}$$
        */
  foreach $u \in V \backslash \{v\}$ **do**
    $p := 0$; $n := 0$;
    foreach $\omega \subseteq V \backslash \{u\}$ **do**
      if $\hat{K}(v,\omega) < \hat{K}(v,\omega \cup \{u\})$ **then** $p := p + 1$ **else if** $\hat{K}(v,\omega) > \hat{K}(v,\omega \cup \{u\})$ **then** $n := n + 1$
    if $n + p > 0$ **then**
      $E := E \cup \{(u,v)\}$;
      if $p \geq n$ **then** $\varepsilon(u,v) := +$ **else if** $p < n$ **then** $\varepsilon(u,v) := -$;
      pick $\omega' \subseteq V \backslash \{u\} : \hat{K}(v,\omega') \neq \hat{K}(v,\omega' \cup \{u\})$;
      $\vartheta(u,v) :=$ `Activity-Value(`u, v, ω', $\delta(v,\cdot)$`)`;
$I := (V, E, \varepsilon, \vartheta, \max)$;
foreach $v \in V$ **do**
  foreach $\omega \subseteq \mathrm{Pre}_I(v)$ **do**
    $\varsigma := \{u \in \omega \mid \varepsilon(u,v) = +\} \cup \{u \in \mathrm{Pre}_I(v) \backslash \omega \mid \varepsilon(u,v) = -\}$;
    $K(v,\omega) := \hat{K}(v,\varsigma)$;
return $M = (I, K)$

- pick a resource ω' on which $\hat{K}(v,\omega') \neq \hat{K}(v,\omega' \cup \{v\})$, algorithm *Activity-Value* is applied to get $\vartheta(v,v)$.
- *minimise the violation of the Snoussi-condition* in v. According to Lemma 2.26, for each resource, if $\hat{K}(v,\omega)$ does not lie on the ω-side of v, then $\hat{K}(v,\omega)$ will be revised in line 16:

$$\hat{K}(v,\hat{\omega}) := \begin{cases} \vartheta(v,v)-1, & \text{if } \varepsilon(v,v)=+,\ v\in\hat{\omega} \\ \vartheta(v,v), & \text{if } \varepsilon(v,v)=+,\ v\notin\hat{\omega} \\ 0, & \text{if } \varepsilon(v,v)=-,\ v\in\hat{\omega} \\ \max_u, & \text{if } \varepsilon(v,v)=-,\ v\notin\hat{\omega} \end{cases}$$

  Note that because of $\hat{\varepsilon}(v,v)=+$, the 3rd case $\varepsilon(v,v)=-, v\in\hat{\omega}$ here is equivalent to the case $\varepsilon(v,v)=-, v\notin\omega$ in Lemma 2.26.

- *Other incoming interactions*, line 17–25. With the revised $\hat{K}(v,\cdot)$, the existence of other incoming interactions $(u,v)\in\hat{E}$ is checked. If an interaction exists, similarly a resource of v with $\hat{K}(v,\omega') \neq \hat{K}(v,\omega'\cup\{u\})$ is picked. algorithm *Activity-Value* is applied to find $\vartheta(u,v)$.

At the end, the IG $I=(V,E,\varepsilon,\vartheta,\max)$ has been constructed.

3. *Assigning* K, line 27–30. A proper logical parameter function K has to be assigned to I, so that for $M=(I,K)$ we get ASTG$(M)\cong T$. For each component $v\in V$ and each state $x\in X$, let ω be the resource of v under x in I and ς the one in $\hat{I}$. Then ς is the symmetric difference of ω with a set of components for which the outgoing edges to v are negative. Therefore, K can be obtained by rearranging $\hat{K}$: for each $v\in V$ and each ω, $K(v,\omega) := \hat{K}(v,\varsigma)$. $M=(I,K)$ generates the same ASTG as T.

This algorithm is a refinement of *Visibility-Model* in order to achieve a model which satisfies more conditions. The refinements are in the second step *inferring* I and include the following.

1. *Infer interactions based on components*. For each component, this algorithm first detects the self-loop and then other incoming interactions.
2. *Decide the signs of interactions based on counting the positive and negative influences*. For each interaction, this algorithm assigns the sign based on the number of *positive and negative influences*.
3. *Minimal violations of the Snoussi-condition*. According to Lemma 2.26, $\hat{K}(v,\cdot)$ is revised for each $v\in V$. In this way, with the rearranged K from $\hat{K}$, $M=(I,K)$ satisfies the Snoussi-condition as much as possible.

Two more details should be mentioned.

1. The decision on the signs of an interaction. In line 11 and 23, $+$ is preferred to $-$ because $+$ will be chosen even $p=n>0$. However, one can also choose to prefer $-$ by letting $\varepsilon(u,v)=-$ if $p\leq n$. The resulting model will satisfy the observability and the Snoussi-condition as much as the model obtained by preferring $+$.
2. The output model M satisfies the Snoussi-condition "as much as possible" in the following sense:
   (a) M has the fewest violations of the Snoussi-condition *in components in the class of all isomorphic models*. The $\hat{K}$ is revised during the inference of I according to Lemma 2.26. This revision will enable the output M carrying the fewest violations of the Snoussi-condition among all models isomorphic to M.

(b) M has the fewest violations of the Snoussi-condition *in interactions in the class of all equivalent models*. The number of violations of the Snoussi-condition in interactions is defined as:

$$s := \#\{(u, v, \omega) \mid (u, v) \in E,\ \omega \subseteq \mathrm{Pre}(v) \backslash \{u\} \wedge\ K(v, \omega) > K(v, \omega \cup \{u\})\}.$$

Lemma 2.26 ensures that among the class of all isomorphic models, M has the lowest number of violations of the Snoussi-condition in all components, but not necessarily so among the class of all equivalent models. It is still possible that there exists another model (with a different interaction graph, among the class of all equivalent models) that has the same number of Snoussi-condition violations in all interactions, but overall a lower number of violations of the Snoussi-condition in all components.

By applying the algorithm *Observability-Snoussi-Model* in Section 2.2.4 we can get the following result:

Theorem 2.34 [Lorenz, 2011] For a given ASTG $T = (X, S)$, algorithm *Observability-Snoussi-Model* will construct a model M which satisfies the observability condition and

1. among all *isomorphic* models of M, M has the fewest number of *violations of the Snoussi-condition on components*.
2. among all *equivalent* models of M, M has the fewest number of *violations of the Snoussi-condition in interactions*.

Illustration

Example 2.35 We take the same ASTG as in Example 2.33 as input for algorithm *Observability-Snoussi-Model*.

1. *Initialisation.* This step is the same as in algorithm *Visibility-Model*. Like in Figure 2.11, a simple IG $\hat{I}$ and the pseudo logical parameter function $\hat{K}$ are constructed.
2. *Inferring I.*
   (a) For component v_1:
   - (v_1, v_1). One positive influence on resource $\{v_2\}$ is found, $p = 1$; there is no negative influence, $n = 0$. Thus $p \leq n$ and $\varepsilon(v_1, v_1) := +$. Algorithm *Activity-Value* with the resource $\{v_2\}$ gives $\vartheta(v_1, v_1) := 2$. Each resource ω of v_1 is checked whether $\hat{K}(v_1, \omega)$ lies on the ω-side from v_1. If not, $\hat{K}(v_1, \omega)$ will be revised accordingly.

$\hat{\omega}$	$v_1 \in \hat{\omega}$?	$\hat{K}(v_1, \hat{\omega})$ vs 2	on $\hat{\omega}$-side?	revision	new $\hat{K}(v_1, \omega)$
$\emptyset$	$\notin$	$2 < 2$?	no	$\vartheta(v_1, v_1) = 2$	2
$\{v_2\}$	$\notin$	$0 < 2$?	yes	-	0
$\{v_1\}$	$\in$	$2 \geq 2$?	yes	-	2
$\{v_1, v_2\}$	$\in$	$2 \geq 2$?	yes	-	2

   - (v_2, v_1). Only one negative influence is found on resource $\emptyset$, and there is no positive influence. Therefore, $\varepsilon(v_2, v_1) := -$. Applying algorithm *Activity-Value* with resource $\emptyset$ gives $\vartheta(v_2, v_1) := 2$.

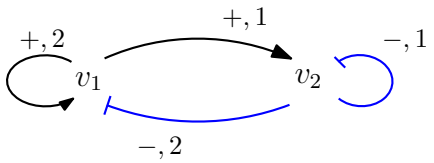
   (b) For component v_2, the same process is repeated as for v_1.

- For (v_2, v_2), only one negative influence is detected on resource $\emptyset$, $n = 1$, and there is no positive influence, $p = 0$. Therefore, $\varepsilon(v_2, v_2) := -$. Using algorithm *Activity-Value* with resource $\emptyset$ gives $\vartheta(v_2, v_2) := 1$.
  Each resource ω of v_2 is checked whether $\hat{K}(v_2, \omega)$ lies on the ω-side from v_2. If not, $\hat{K}(v_2, \omega)$ will be revised accordingly.

$\hat{\omega}$	$v_2 \in \hat{\omega}$?	$\hat{K}(v_2, \hat{\omega})$ vs 1	on $\hat{\omega}$-side?	revise	new $\hat{K}(v_1, \omega)$
$\emptyset$	$\notin$	$2 < 1$?	no	$\max_{v_2} = 2$	2
$\{v_2\}$	$\in$	$0 \geq 1$?	no	0	0
$\{v_1\}$	$\notin$	$1 < 1$?	no	$\max_{v_2} = 2$	2
$\{v_1, v_2\}$	$\in$	$1 \geq 1$?	yes	-	1

- (v_1, v_2). There is one positive influence on resource $\{v_1\}$, $p = 1$, and no negative influence. Thus $\varepsilon(v_1, v_2) := +$. Applying algorithm *Activity-Value* with resource $\{v_1\}$ gives $\vartheta(v_2, v_1) := 1$.

Figure 2.13 shows the output, the IG I and the revised pseudo logical parameter function $\hat{K}$.



ω	$\hat{K}(v_1, \omega)$	ω	$\hat{K}(v_2, \omega)$
$\emptyset$	2	$\emptyset$	2
$\{v_2\}$	0	$\{v_2\}$	0
$\{v_1\}$	2	$\{v_1\}$	2
$\{v_1, v_2\}$	2	$\{v_1, v_2\}$	1

Figure 2.13: The real IG I, and the revised $\hat{K}$.

3. *Assigning* K. Similar as in Example 2.33, for all $v \in V$ and all $\omega \subseteq \mathrm{Pre}_I(v)$, set $K(v, \omega) := \hat{K}(v, \varsigma)$, where $\varsigma = \{u \in \omega \mid \varepsilon(u, v) = +\} \cup \{u \notin \mathrm{Pre}_I(v) \backslash \omega \mid \varepsilon(u, v) = -\}$. K is an rearrangement of $\hat{K}$ relating the resources by the symmetric difference with the set of inhibiting components, see Figure 2.14.

ω	ς	$K(v_1, \omega) := \hat{K}(v_1, \varsigma)$
$\emptyset$	$\{v_2\}$	0
$\{v_2\}$	$\emptyset$	2
$\{v_1\}$	$\{v_1, v_2\}$	2
$\{v_1, v_2\}$	$\{v_1\}$	2
ω	ς	$K(v_2, \omega) := \hat{K}(v_2, \varsigma)$
$\emptyset$	$\{v_2\}$	0
$\{v_2\}$	$\emptyset$	2
$\{v_1\}$	$\{v_1, v_2\}$	1
$\{v_1, v_2\}$	$\{v_1\}$	2

Figure 2.14: K obtained by rearranging $\hat{K}$.

Result: The output model $M = (I, K)$ carries only observable interactions and has no violations of the Snoussi-condition, and $\mathrm{ASTG}(M) \cong T$. △

Complexity The space complexity is similar to algorithm *Visibility-Model*. Both algorithms have the same size of the input and output. Concerning the intermediate variables, algorithm

Observability-Snoussi-Model needs a few more: p, n, ω', which is very small. The space complexity of algorithm *Observability-Snoussi-Model* is again $\mathcal{O}(N^{N+1})$.

The time cost is calculated in three steps.

1. *Initialisation*, line 1–4. Similar to Algorithm 2.2.3, getting $\hat{K}$ for $\hat{I}$ has a maximal running time of $\mathcal{O}(N^2 2^N)$.
2. *Inferring* I, line 5–24. For each component v:
   - Self loop (v, v), line 6–16.
     - line 6–8, counting positive and negative influences. In the worst case, each pair of logical parameters under a resource with and without v has to be compared. The cost is $O(2^{N-1})$;
     - line 9–13, for assigning $\varepsilon(v, v)$ and $\vartheta(v, v)$ the cost is $\mathcal{O}(N)$.
     - minimising the violation of the Snoussi-condition: for each resource checking whether $\hat{K}(v, res)$ lies on the resource side from v, which costs $\mathcal{O}(2^N)$.

     So, for the self loop, the overall cost is $\mathcal{O}(2^N)$.
   - Other $N - 1$ incoming edges, line 17–25:
     - line 18–20 counting positive influences p and negative ones n, cost $\mathcal{O}(2^N)$.
     - line 21–25, assigning $\varepsilon(u, v)$ and $\vartheta(u, v)$ cost $\mathcal{O}(2^N)$.

     Thus for the other $(N - 1)$ other incoming interactions the cost $\mathcal{O}(N2^N)$.

   Overall, the time cost of second step is $N(\mathcal{O}(2^N) + \mathcal{O}(N2^N))$, which is $\mathcal{O}(N2^N)$.
3. *Assigning* K.. The same as in algorithm *Visibility-Model*. This step costs $\mathcal{O}(N2^N)$.

In total, the time cost of the algorithm is $\mathcal{O}(N^2 2^N) + \mathcal{O}(N2^N) + \mathcal{O}(N2^N) = \mathcal{O}(N^2 2^N)$. This algorithm takes more time than algorithm *Visibility-Model* due to the refinement. However, they both have the same level of time complexity.

2.3 Discussion

Let the number of components in the system be again $|V| = N \in \mathbb{N}^+$. All four algorithms take as inputs an ASTG which has at least the size of the state space X. If the system is Boolean, then $|X| = 2^N$. If the system is multi-valued, every component can have an activity level as high as the total number of components in the system, which means $|X| \leq (N + 1)^N$, *i.e.*, $\mathcal{O}(N^N)$. The worst-case space complexity and the time complexity for each algorithm are summarised in Table 2.2.

Algorithm:	$S(N)$	$T(N)$
Logical-Parameters	$\mathcal{O}(N^{N+1})$	$\mathcal{O}(N^2 2^N)$
Activity-Value	$\mathcal{O}(N^{N+1})$	$\mathcal{O}(N)$ or $\mathcal{O}(N^2)$
Visibility-Model	$\mathcal{O}(N^{N+1})$	$\mathcal{O}(N^2 2^N)$
Observability-Snoussi-Model	$\mathcal{O}(N^{N+1})$	$\mathcal{O}(N^2 2^N)$

Table 2.2: The space and time complexity of the four algorithms.

Algorithm *Logical-Parameters* and *Activity-Value* are called from the other two reverse engineering (RE) algorithms. Algorithm *Visibility-Model* and *Observability-Snoussi-Model* have very similar procedures to infer a model for a given ASTG T as input. Assuming the input is really an ASTG, these two algorithms will always generate a model.

1. The first step *initialisation* is the same in both cases. Algorithm *Logical-Parameters* is called to get a pseudo logical parameter function $\hat{K}$ based on a complete positive interaction graph $\hat{I}$. Once the signs of all incoming interactions of a component are known, the resource of it for each extremal state can be determined easily. Information on all state transitions is extracted from the extremal states.

2. In the second step *inferring* I, the existence of an interaction is based on the *visibility condition*. The threshold value of an interaction is assumed to be unique. It is obtained by calling algorithm *Activity-Value* on a resource which makes this interaction visible.

3. The last step *assigning* K is also the same. An IG I is inferred from $\hat{K}$ with no unnecessary interactions. A suitable logical parameter function K is needed for I. For each component, the difference of its resource in I with the one in $\hat{I}$ is a subset of those unnecessary and negative incoming interactions.

The difference of these two RE algorithms lies in the second step *inferring* I.

1. Algorithm *Visibility-Model*: the sign of each interaction is determined based on the value of $\hat{K}$ at a resource which makes the interaction visible. This is similar to choosing the first resource which makes an interaction ($\hat{I}$) visible, and assigning it a "correct" sign in I, which will be finally observable in the output model M.

2. Algorithm *Observability-Snoussi-Model*: the sign of each interaction is determined using an overall comparison of the values of $\hat{K}$ at all resources which make the interaction visible based on the *observability condition*.

3. In algorithm *Observability-Snoussi-Model*, for each component u, $\hat{K}$ is revised to get a minimal number of violations of the Snoussi-condition in u, according to Lemma 2.26.

According to Corollary 2.27, a violation of the Snoussi-condition in an interaction implies also a violation in a component. However, a violation in an component can mean more than one violations in some interactions. Checking the total number of Snoussi-condition violations is expensive, due to the high cost of calculating all pairs of resources $\omega \subseteq \varsigma \subseteq V$.Checking only the number of Snoussi-condition violations in all interactions is easier for many fewer resource pairs are compared (all $\omega \subseteq V \backslash \{u\}$ vs $\omega \cup \{u\}$).

Algorithm *Observability Snoussi-Model* constructs an observable model with minimal Snoussi-condition violations in its isomorphism class. Because the output model has only necessary edges, the number of violations of the Snoussi-condition in all interactions is also minimal in its equivalence class. However, it is still an open question whether this model also has the lowest number of Snoussi-condition violations in all components among all models of its equivalence class. It is possible that there exists another model (with a different IG) that has the same number of Snoussi-condition violations in all interactions, but overall a lower number of Snoussi-condition violations in all components.

The main memory and time consuming part in both RE algorithms is related to operations on resources. For a system of N components, the maximal number of resources for each component is 2^N. Moreover, the comparison of the logical parameters between pairs of resources for the signs of interactions in both RE algorithms is time consuming and executed more than once.

Both RE algorithms can output a model with a minimal number of interactions. Algorithm *Observability-Snoussi-model* spends additional efforts to achieve a logical parameter function for which the output model satisfies more restrictive model conditions.

Without knowing the model in advance, the complete ASTG is hard to obtain by measurements from experiments. However, both RE algorithms were designed to infer a model from a perfect ASTG which leads to a problem in practice. If the input is not a perfect ASTG, then the algorithms do not work as they are supposed to:

1. Algorithm *Logical-Parameters* does not care about the input details beyond the extremal rows.

2. Both RE algorithms are not sensitive to the presence of multiple edges in the input: for example, choosing different resources which can make an interaction visible, different threshold values can be obtained. In this case, the output models are not able to regenerate the input anymore.

However, if the input is not perfect and carries some incorrect transitions, these two RE algorithms can still output a model for an ASTG whose extremal rows agree with the input. These issues will be further explored in Chapter 3.

The size of the state space and the logical parameter function grows exponentially with the number of components. For this reason, the two RE algorithms slow down for larger regulatory networks. The four algorithms have been implemented in Matlab version R2016a (about 500 line of code) on a $x86\_64$ and $4GB$ memory machine under Linux operation system. The software runs very fast on small number of components (within 10), within a few seconds. If the number of components gets larger, it can be very slow. For example, given an ASTG with $32,739$ transitions from a Boolean GRN of 15 components (mTOR-MAPK signalling transduction network), both algorithms need more than 4 days and even the verification of the output model takes 3 hours. The Matlab package is available at https://github.com/riplotus/RE_LorenzAlgorithms.

Chapter 3

Asynchronous state transition graphs and generalised Lorenz algorithms

When designing Lorenz algorithms as presented in Chapter 2 it was assumed that the input graph is always a valid ASTG. If this is not the case, the algorithms may still work, but they will compute a model that does not exactly reproduce the input. In Section 3.1.1 we propose three conditions that are necessary and sufficient for a graph based on the state space to be a valid ASTG. These conditions characterise the correct inputs for Lorenz algorithms. Inspired by these ASTG conditions, a fourth condition for an ASTG to admit a model satisfying the observability and the Snoussi-conditions is proposed in Section 3.1.2. These four conditions are then embedded into Lorenz algorithms, as *generalised* Lorenz algorithms in Section 3.2. The generalised Lorenz algorithms can tell whether the input is a valid ASTG and if so construct a model with the required properties.

3.1 Characterising asynchronous state transition graphs

Lorenz algorithms as presented in Chapter 2 will always construct a model, even if the input is not an ASTG. While this allow the user getting a model even if there are some errors in the input graph, the output model will not be able to regenerate the input. In general, it is possible one gives an input which is a directed graph on the state space and is not a valid ASTG. Such non-ASTG graphs may have incomplete information in order to be ASTGs, unexpected transitions among some states, influences potentially caused by multiple interactions, etc.

When looking at an ASTG as a directed graph on the state space, graph-theoretical properties characterising ASTGs are of great interest. In this chapter, three ASTG conditions will be derived and proven to be necessary and sufficient for a graph based on the state space (see Definition 3.1) to be an ASTG. These conditions guarantee a valid input for Lorenz algorithms, such that the output will be a model which regenerates the input. Furthermore, these conditions can be embedded into an extended version of Lorenz algorithms that is able to handle general inputs.

3.1.1 Three ASTG conditions

Definition 3.1 (Graph based on a state space, $\mathcal{G}_X$) A directed graph $\mathcal{G}_X = (X, \mathcal{E})$, where $X = \Pi_{u \in V} X_u$ is a state space with $X_u = \{0, \ldots, \max_u\}$, for all $u \in V$, and $\mathcal{E}$ represents

directed edges between the states, is called a *graph based on the state space* X.

Let $\mathcal{G}_X$ be a graph based on a state space X. In the following, we formulate three conditions, ASTG condition 1, 2 and 3 to determine whether or not $\mathcal{G}_X$ is an ASTG.

ASTG condition 1 *(Asynchronicity)* In $\mathcal{G}_X$, any two states connected by a directed edge have Hamming distance one, which means that they differ in only one component.

ASTG condition 2 *(Unitarity)* In $\mathcal{G}_X$, every state $x \in X$ has at most one outgoing edge in each direction $u \in V$, which has length one.

Based on ASTG conditions 1 and 2, we define the notion of *destination value*.

Definition 3.2 (Destination value) Let $\mathcal{G}_X$ be a graph based on X satisfying the ASTG conditions 1 and 2. Given a u-row $\tau^u = (x^0, \ldots, x^{\max_u})$, let $\delta(u, x^i) \in \{-1, 0, +1\}$ describe the transition from state $x^i \in \tau^u$ in direction u, for all $i \in \{0, \ldots, \max_u\}$. The destination value of x^i in direction of u is defined as:

$$\mathrm{dest}(u, x^i) = \begin{cases} i & \text{if } \delta(u, x^i) = 0 & (a) \\ i+1 & \text{if } \delta(u, x^i) = 1 \text{ and } \delta(u, x^{i+1}) \neq 1 & (b) \\ i-1 & \text{if } \delta(u, x^i) = -1 \text{ and } \delta(u, x^{i-1}) \neq -1 & (c) \\ \mathrm{dest}(u, x^{i+1}) & \text{if } \delta(u, x^i) = \delta(u, x^{i+1}) = 1 & (d) \\ \mathrm{dest}(u, x^{i-1}) & \text{if } \delta(u, x^i) = \delta(u, x^{i-1}) = -1 & (e) \end{cases} \tag{3.1}$$

for $i \in \{0, \ldots, \max_u\}$. The destination values of the u-row τ^u in direction of u are given by the vector

$$\mathrm{dest}(u, \tau^u) = (\mathrm{dest}(u, x^0), \ldots, \mathrm{dest}(u, x^{\max_u})).$$

The concept of destination value is very similar to the notion of *monotone target value* (MTV) defined by [Streck et al., 2015]. MTV means the target value that a state can transit to in one direction in a *transition system (TS)* (which can be understood as a state transition graph). In an ASTG, the destination value is the same as the MTV. However, the *destination value* is defined on a graph based on a state space $\mathcal{G}_X$, which does not have to be a valid ASTG.

The following Lemma 3.3 describes the destination values for the three u-row types in an ASTG.

Lemma 3.3 In an ASTG T_M, every u-row τ^u has at most two different destination values in direction of u. Given $a := \mathrm{dest}(u, x^0)$ and $b := \mathrm{dest}(u, x^{\max_u})$, there are the following three cases:

1. If τ^u belongs to *open*-type, then for all $x \in \tau^u$, $\mathrm{dest}(u, x) = a = b$.
2. If τ^u belongs to *pos*-type with a threshold t, then it holds that $a < t \leq b$, and $\mathrm{dest}(u, x^0) = \cdots = \mathrm{dest}(u, x^{t-1}) = a$ and $\mathrm{dest}(u, x^t) = \cdots = \mathrm{dest}(u, x^{\max_u}) = b$.
3. If τ^u belongs to *neg*-type with a threshold t, then $a \geq t > b$, and $\mathrm{dest}(u, x^0) = \cdots = \mathrm{dest}(u, x^{t-1}) = a$ and $\mathrm{dest}(u, x^t) = \cdots = \mathrm{dest}(u, x^{\max_u}) = b$.
   In addition, $a = t$ and $b = t - 1$.

Proof. From Definition 3.2 and Proposition 2.10, we get the following cases:

1. If τ^u belongs to the *open* type, then there exist only one state x^a with $\delta(u, x^a) = 0$, $\delta(u, x^0) = \cdots = \delta(u, x^{a-1}) = 1$ and $\delta(u, x^{a+1}) = \cdots = \delta(u, x^{\max_u}) = -1$. Thus, $\mathrm{dest}(u, x^0) = \cdots = \mathrm{dest}(u, x^{\max_u}) = a$, see Figure 3.1.

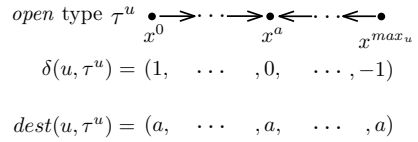


Figure 3.1: A *open* type u-row τ^u, the state transitions and the destination values.

2. If τ^u belongs to the *pos* type with threshold t, then there exists $a < b \in \{0, \ldots, \max_u\}$, so that $\delta(u, x^a) = \delta(u, x^b) = 0$. It holds that $\delta(u, x^{t-1}) \leq 0$ and $\delta(u, x^t) \geq 0$. $\delta(u, \tau^u) = (\delta(u, x^0), \ldots, \delta(u, x^a), \ldots, \delta(u, x^{t-1}),\ \delta(u, x^t), \ldots, \delta(u, x^b), \ldots, \delta(u, x^{\max_u})) = (1, \ldots, 0, \ldots, -1, 1, \ldots, 1, 0, -1, \ldots, -1)$. From the definition of the destination value, $\mathrm{dest}(u, x^0) = \cdots = \mathrm{dest}(u, x^{t-1}) = a$ and $\mathrm{dest}(u, x^t) = \cdots = \mathrm{dest}(u, x^{\max_u}) = b$, see Figure 3.2.

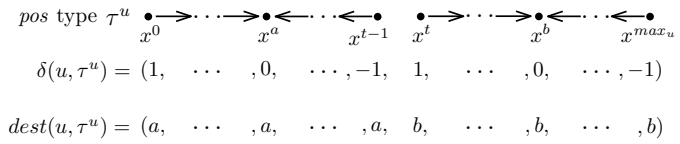


Figure 3.2: A *pos* type u-row τ^u, the state transitions and the destination values.

3. If τ^u belongs to the *neg* type with threshold t, then $\delta(u, x^i) = 1$ for $i \in \{0, \ldots, t-1\}$, and $\delta(u, x^i) = -1$ for $i \in \{t, \ldots, \max_u\}$. From the definition of destination values, $\mathrm{dest}(u, x^i) = t$ for $i \in \{0, \ldots, t-1\}$, and $\mathrm{dest}(u, x^i) = t-1$ for $i \in \{t, \ldots, \max_u\}$. Moreover, $a = t$ and $b = t-1$, see Figure 3.3.

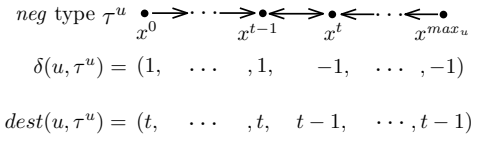


Figure 3.3: A *neg* type u-row τ^u, the state transitions and the destination values.

Therefore, Lemma 3.3 is proved. □

According to Proposition 2.10 on the three types u-rows, we can enumerate all possible kinds of u-rows of a certain length (*i.e.*, their number of states). For example, for all u-rows of length 4, if the threshold of those u-rows of *pos* and *neg* type is set to 2, we can find in total 9 different u-rows: 4 of *pos* type, 1 of *neg* type and 4 of *open* type. Figure 3.4 shows these 9 u-rows and their destination values.

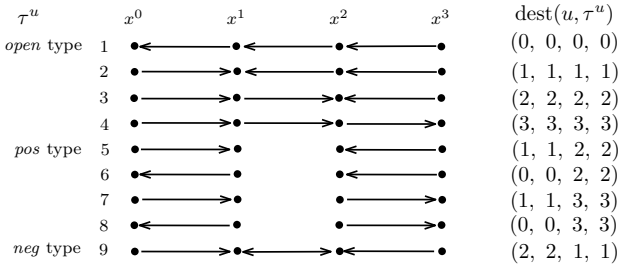


Figure 3.4: All 9 u-rows of length 4 with threshold 2 for the *pos* and *neg* types and their destination values.

We are now able to formulate the third ASTG condition. Note that by ASTG condition 2, for each state and in each direction, there exists only one destination value.

ASTG condition 3 *(u-hypercube)* Let $\mathcal{G}_X$ be a graph based on X satisfying the ASTG conditions 1 and 2. For all components $u, i \in V$, there exists $t_{ui} \in \{0, \ldots, \max_i\}$ such that for all *u-hypercubes* of the form $\mathscr{X}_u := \Pi_{i=1}^{n} \mathfrak{X}_i \subseteq X$ with either $\mathfrak{X}_i = \{0, \ldots, t_{ui} - 1\}$ or $\mathfrak{X}_i = \{t_{ui}, \ldots, \max_i\}$, we have

$$\forall x, y \in \mathscr{X}_u : \mathrm{dest}(u, x) = \mathrm{dest}(u, y).$$

The destination value of the u-hypercube in direction of u is denoted as $\mathrm{dest}(u, \mathscr{X}_u)$ which equals to $\mathrm{dest}(u, x)$ for all $x \in \mathscr{X}_u$. Moreover, the set of all these u-hypercubes $\mathscr{X}_u$ is denoted by $H(u)$.

The motivation behind the u-hypercubes is the following. In an IG, if there exists a regulation from v to u, then $t_{uv} \in \{1, \ldots, \max_v\}$ is the position where this influence can be observed. If there is no regulation from component v to u, then t_{uv} can be any value in $\{0, \max_v\}$, so that in the v-th dimension of $\mathscr{X}_u$ we have $\mathfrak{X}_v$ is $\{0, \ldots, \max_v\}$.

Remark 3.4 Given an ASTG $\mathcal{G}_X$ satisfying the ASTG condition 3 for the set of u-hypercubes $H(u)$, there always exists a set of u-hypercubes $\tilde{H}(u)$ with $\tilde{t}_{ui} > 0$ for all $u, i \in V$, such that $\mathcal{G}_X$ still satisfies the u-hypercube condition on $\tilde{H}(u)$.

Proof. For all $u, i \in V$, if $t_{ui} = 0$ and $\mathfrak{X}_i = \{0, \ldots, \max_i\}$, let $\tilde{t}_{ui}$ be an arbitrary value from $\{1, \ldots, \max_i\}$. Then $\tilde{\mathfrak{X}}_i^1 = \{0, \ldots, \tilde{t}_{ui} - 1\}$, $\tilde{\mathfrak{X}}_i^2 = \{\tilde{t}_{ui}, \ldots, \max_i\}$ and every u-hypercube $\mathscr{X}_u \in H(u)$ can be divided into two u-hypercubes, $\mathscr{X}_u^1$ with $\tilde{\mathfrak{X}}_i^1$ and $\mathscr{X}_u^2$ with $\tilde{\mathfrak{X}}_i^2$. Let $\tilde{H}(u)$ denote the new set of u-hypercubes. Since $\mathscr{X}_u^1$ and $\mathscr{X}_u^2$ are subsets of $\mathscr{X}_u$, the destination values of $\mathscr{X}_u^1$ and $\mathscr{X}_u^2$ are the same as for $\mathscr{X}_u$. Thus $\mathcal{G}_X$ satisfies the u-hypercube condition on $\tilde{H}(u)$. □

Note that for all other dimensions different from u, *i.e.*, for all $i \neq u \in V$ and $x, y \in \mathscr{X}_u$, $\mathrm{dest}(i, x)$ does not need to be the same as $\mathrm{dest}(i, y)$. Before proving necessity of ASTG condition 3, we first illustrate it on a 2-dimensional and a 3-dimensional example.

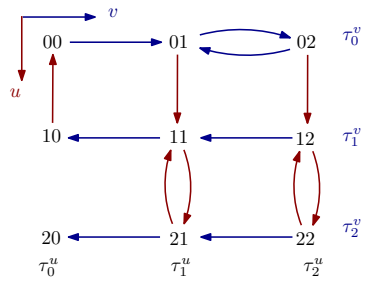
Example 3.5 This example is to illustrate ASTG condition 3 on a 2-D ASTG T and a corresponding model M. Note that the model of T does not need to satisfy the Snoussi-condition. Figure 3.5 shows an ASTG T which satisfies ASTG condition 3, and two models of T. The hypercubes of T are shown in Figure 3.6 and 3.7.

In particular, Figure 3.6a shows the state transitions in direction of u in T, and Figure 3.6b shows all 4 u-hypercubes with their destination values. Take for example $\mathscr{X}_u^1 = \{00, 10\}$ in the top-left of Figure 3.6b. All states have the same destination value, $\mathrm{dest}(u, \mathscr{X}_u^1) = 0$. Table 3.1 shows the set of u-hypercubes.

$\mathscr{X}_u^j =$	$\mathfrak{X}_u$	$\times$	$\mathfrak{X}_v$	$\mathrm{dest}(u, \mathscr{X}_u^j)$
$\mathscr{X}_u^1 =$	$\{0, 1\}$	$\times$	$\{0\}$	0
$\mathscr{X}_u^2 =$	$\{0, 1\}$	$\times$	$\{1, 2\}$	2
$\mathscr{X}_u^3 =$	$\{2\}$	$\times$	$\{0\}$	2
$\mathscr{X}_u^4 =$	$\{2\}$	$\times$	$\{1, 2\}$	1

Table 3.1: 4 u-hypercubes and their destination values in direction of u, in Figure 3.6.

Similarly, the set of v-hypercubes is shown in Figure 3.7. △

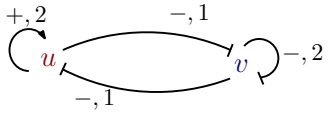


(a) ASTG T.

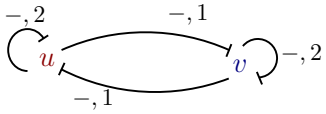
ω	$K(u,\omega)$	$K(v,\omega)$
$\emptyset$	2	0
$\{v\}$	0	0
$\{u\}$	1	0
$\{u,v\}$	2	2

ω	$K'(u,\omega)$	$K'(v,\omega)$
$\emptyset$	1	0
$\{v\}$	2	0
$\{u\}$	2	0
$\{u,v\}$	0	2

(b) K.

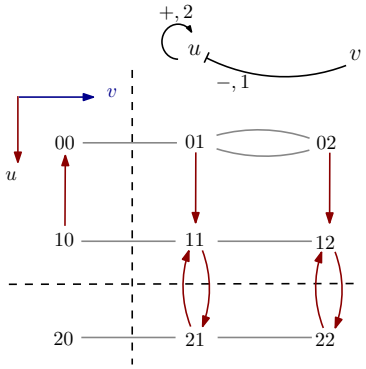


(c) IG I.

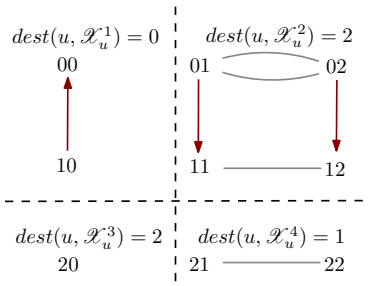


(d) IG I'.

Figure 3.5: (a) Common ASTG T of the models $M = (I, K)$ and $M' = (I', K')$. (b) Logical parameter functions K and K'. (c) IG I. (d) IG I'. Neither M nor M' satisfies the Snoussi-condition.

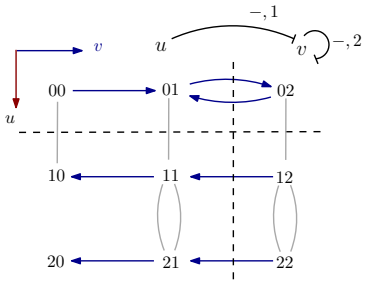


(a) State transitions of u.

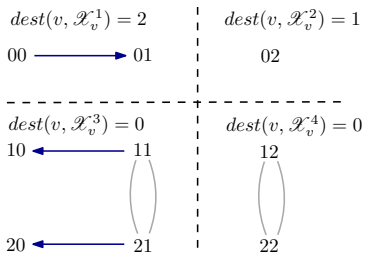


(b) $H(u)$.

Figure 3.6: (a) State transitions in direction of u. For I, the incoming interactions of u are shown on the top. (b) All $\mathscr{X}_u$, $H(u)$, and their destination values. For these $\mathscr{X}_u$, $t_{uu} = 2$ and $t_{uv} = 1$, which are equal to the thresholds $\vartheta(u,u)$ and $\vartheta(v,u)$, respectively.



(a) State transitions of v.



(b) $H(v)$.

Figure 3.7: (a) State transitions in direction of v. For I, the incoming interactions of v are shown on the top. (b) All $\mathscr{X}_v$, $H(v)$, and their destination values. For these $\mathscr{X}_v$, $t_{vu} = 1$ and $t_{vv} = 2$, which are equal to the thresholds $\vartheta(u,v)$ and $\vartheta(v,v)$, respectively.

Example 3.6 This example is to illustrate ASTG condition 3 using a 3-D ASTG T_M and a corresponding model M, as shown in Figure 3.8.

Figure 3.9 shows all the transitions in direction of v_1 (Figure 3.9a), and 8 v_1-hypercubes with their destination values (Figure 3.9b). Let $\mathscr{X}^1_{v_1} = \{000, 010, 100, 110\}$ be the v_1-hypercube on the top-left of Figure 3.9b. From the transitions of all states in $\mathscr{X}^1_{v_1}$ in direction of v_1, it is obvious that $\mathrm{dest}(v_1, \mathscr{X}^1_{v_1}) = 0$. Table 3.2 shows all the 8 v_1-hypercubes of v_1.

$\mathscr{X}^j_{v_1} =$	$\mathfrak{X}_{v_1}$	$\times$	$\mathfrak{X}_{v_2}$	$\times$	$\mathfrak{X}_{v_3}$	$\mathrm{dest}(v_1, \mathscr{X}^j_{v_1})$
$\mathscr{X}^1_{v_1} =$	$\{0,1\}$	$\times$	$\{0,1\}$	$\times$	$\{0\}$	0
$\mathscr{X}^2_{v_1} =$	$\{0,1\}$	$\times$	$\{0,1\}$	$\times$	$\{1,2\}$	0
$\mathscr{X}^3_{v_1} =$	$\{0,1\}$	$\times$	$\{2\}$	$\times$	$\{0\}$	0
$\mathscr{X}^4_{v_1} =$	$\{0,1\}$	$\times$	$\{2\}$	$\times$	$\{1,2\}$	3
$\mathscr{X}^5_{v_1} =$	$\{2,3\}$	$\times$	$\{0,1\}$	$\times$	$\{0\}$	0
$\mathscr{X}^6_{v_1} =$	$\{2,3\}$	$\times$	$\{0,1\}$	$\times$	$\{1,2\}$	3
$\mathscr{X}^7_{v_1} =$	$\{2,3\}$	$\times$	$\{2\}$	$\times$	$\{0\}$	3
$\mathscr{X}^8_{v_1} =$	$\{2,3\}$	$\times$	$\{2\}$	$\times$	$\{1,2\}$	3

Table 3.2: 8 v_1-hypercubes and their destination values, in Figure 3.9.

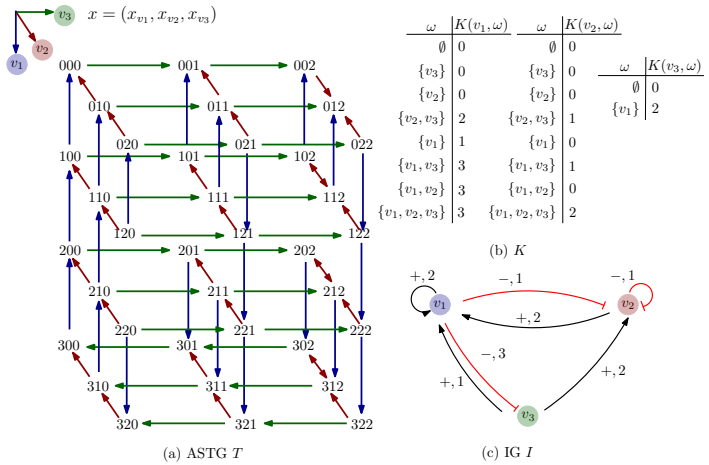
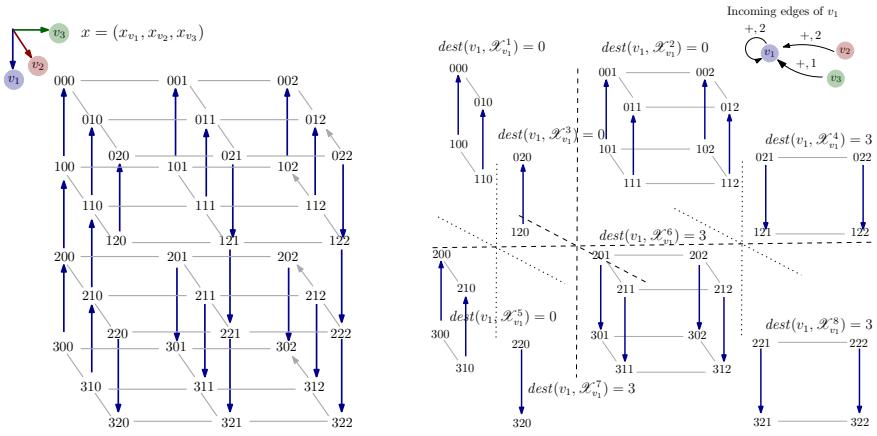
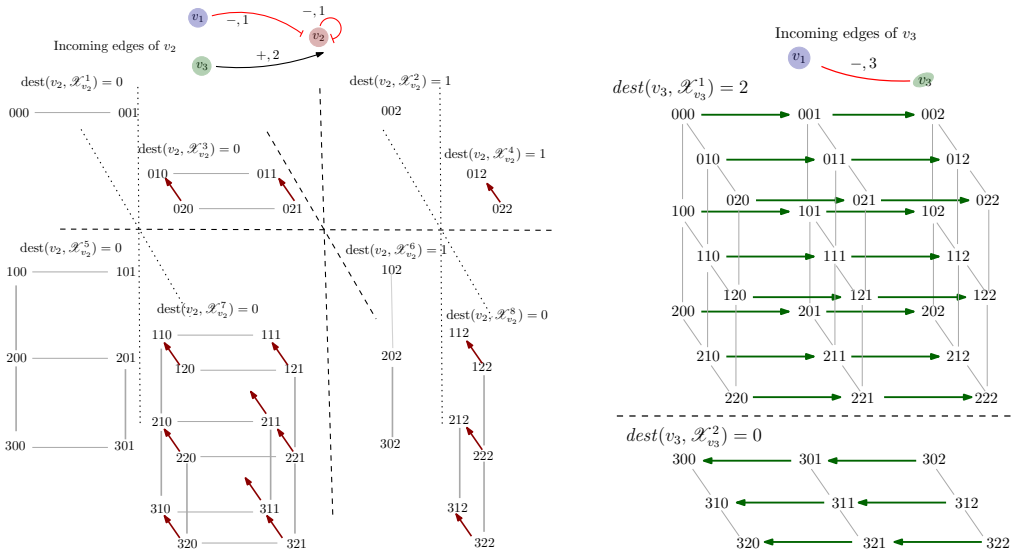


Figure 3.8: (a) An ASTG T. (b), (c) A model of T, $M = (I, K)$.



(a) State transitions of v_1. (b) $H(v_1)$.

Figure 3.9: (a) State transitions in direction of v_1. All incoming edges of v_1 are shown on the top-right. (b) All v_1-hypercubes, $H(v_1)$, and the destination values of all v_1-hypercubes. $t_{v_1v_1} = 2$, $t_{v_1v_2} = 2$ and $t_{v_1v_3} = 1$, which equal to the thresholds of the incoming edges.



(a) $H(v_2)$. (b) $H(v_3)$.

Figure 3.10: (a) All v_2-hypercubes, $H(v_2)$ and their destination values. $t_{v_2v_1} = 1$, $t_{v_2v_2} = 1$ and $t_{v_2v_3} = 2$. (b) v_3-hypercubes, $H(v_3)$ and their destination values. $t_{v_3v_1} = 3$, $t_{v_1v_2} = t_{v_1v_3} = 0$.

△

We next show that the destination value is the same for all states in the set $X^{u,\omega}$ introduced in the following definition.

Definition 3.7 (Set of states for a component under a resource) Let $I = (V, E, \vartheta, \varsigma, \max)$ be an IG. For a component $u \in V$ and one of its resources $\omega \subseteq \mathrm{Pre}(u)$, the *set of states for u under ω* is defined as $X^{u,\omega} := \{x \in X \mid \mathrm{Res}_u(x) = \omega\}$.

Lemma 3.8 Consider a model $M = (I, K)$ and its ASTG T_M. For each $u \in V$ and each resource $\omega \subseteq \mathrm{Pre}(u)$, for all states $x, y \in X^{u,\omega}$, $\mathrm{dest}(u, x) = \mathrm{dest}(u, y)$.

Proof. Firstly, (1) will show that for any u-row, those states in $X^{u,\omega}$ have the same destination values in direction of u. Then, (2) will show that the structures of the u-rows which contain states in $X^{u,\omega}$ are isomorphic. Lastly, (3) will show for all states in $X^{u,\omega}$, they have the same destination values in direction of u.

(1) According to Lemma 3.3, in an *open* type u-row, all states have the same destination value; and in a u-row of *pos* or *neg* type, the states where u has resource ω share the same destination value a, and those where u has the other resource $\omega\Delta\{u\}$ share the same destination value b. Therefore, for any u-rows those states in $X^{u,\omega}$ have the same destination values in direction of u.

(2) According to Lemma 2.19, for two states x, $y \in X$ such that there exists a component $u \in V$ with $Res_u(x)\backslash u = Res_u(y)\backslash u$, the u-row containing x is isomorphic to the u-row containing y. Therefore, for any states x, $y \in X^{u,\omega}$, the u-row containing x is isomorphic to the u-row containing y.

(3) For any states $x, y \in X^{u,\omega}$, one of the following two cases holds:

  (a) x and y lie on the same u-row. From (1), $\mathrm{dest}(u, x) = \mathrm{dest}(u, y)$.

  (b) x and y lies on different u-rows. From (2), the u-row $\tau^u_1 = (x^0, \cdots, x^{\max_u})$ containing x is isomorphic to the u-row $\tau^u_2 = (y^0, \cdots, y^{\max_u})$ containing y. By Defini-

tion 3.2 this implies that $\mathrm{dest}(u, x^i) = \mathrm{dest}(u, y^i)$ for all $i \in \{0, \ldots, \max_u\}$.

Together with (1), it holds that $\mathrm{dest}(u, x) = \mathrm{dest}(u, y)$.

Therefore, for all states $x, y \in X^{u,\omega}$, $\mathrm{dest}(u, x) = \mathrm{dest}(u, y)$. □

Theorem 3.9 (Necessity) Any ASTG $T = (X, S)$ satisfies the three ASTG conditions 1, 2 and 3.

Proof. According to the definition of an ASTG, there is at least one model M such that $ASTG(M) \cong T$. From how ASTG is defined, we know that T is also a graph based on X. Since the state transitions in an ASTG are asynchronous and unitary, T satisfies the ASTG condition 1 and 2. Now we want to prove that T satisfies the u-hypercube ASTG condition 3.

1. Firstly, for each component $u \in V$, we construct a set of u-hypercubes $H(u)$ in the following way:

   (a) For all $v \in V$, if $(v, u) \in E$, then let $t_{uv} := \vartheta(v, u)$; otherwise, let t_{uv} be any value from $\{0, \ldots, \max_v\}$.

   (b) Let $H(u) = \cup\mathscr{X}_u$, and each u-hypercube $\mathscr{X}_u := \Pi_{i \in V}\mathfrak{X}_i$, with $\mathfrak{X}_i$ either equals to $\{0, \ldots, t_{ui}-1\}$ or $\{t_{ui}, \ldots, \max_i\}$. Note that, if $t_{ui} = 0$, then $\mathfrak{X}_i = \{0, \ldots, \max_i\}$.

2. Next we show that each u-hypercube $\mathscr{X}_u \in H(u)$ is included in a set of states with the same resource.

   (a) Let $X^{u,\omega} = \{x \in X \mid \mathrm{Res}_u(x) = \omega\}$ the set of states for u under ω. For all $i \in V$,

   - if $i \in \mathrm{Pre}(u)$, then t_{ui} equals to $\vartheta(i, u)$. This means, $\mathfrak{X}_i$ either equals to $\{0, \ldots, \vartheta(i, u) - 1\}$ or $\{\vartheta(i, u), \ldots, \max_i\}$. Since the states in $X^{u,\omega}$ have the same resource, we have for all $x \in X^{u,\omega}$, $x_i \in \mathfrak{X}_i$.

   - if $i \notin \mathrm{Pre}(u)$, then t_{ui} can be any value from $\{0, \ldots, \max_i\}$. Since $X_i^{u,\omega}$ equals to $\{0, \ldots, \max_i\}$, and $\mathfrak{X}_i$ either equals to $\{0, \ldots, t_{ui} - 1\}$ or $\{t_{ui}, \ldots, \max_i\}$, we have $\mathfrak{X}_i \subseteq X_i^{u,\omega}$.

   From above, $\mathscr{X}_u \subseteq X^{u,\omega}$ and each $\mathscr{X}_u \in H(u)$ has a unique resource ω for u. Thus there exists a set of states under the same resource $X^{u,\omega}$ with $\mathscr{X}_u \subseteq X^{u,\omega}$.

   (b) Let $\omega := \mathrm{Res}_u(x)$, for all $x \in \mathscr{X}_u$. According to Lemma 3.8, all states of $X^{u,\omega}$ have the same destination value in direction of u. Since $\mathscr{X}_u \subseteq X^{u,\omega}$, all states in a u-hypercube have the same destination value.

From the above cases, we proved that, for an ASTG T, for each component u, we can always construct a set of u-hypercubes $H(u)$, such that all states in each u-hypercube $\mathscr{X}_u$ have the same destination value in direction of u, *i.e.*, for all $x, y \in \mathscr{X}_u$, $\mathrm{dest}(u, x) = \mathrm{dest}(u, y) = \mathrm{dest}(u, \mathscr{X}_u)$. Therefore, T satisfies ASTG condition 3. □

Theorem 3.10 (Sufficiency) If a graph $\mathcal{G}_X$ based on a state space X satisfies the three ASTG conditions 1, 2 and 3, then $\mathcal{G}_X$ is an ASTG.

Proof. Suppose that $\mathcal{G}_X$ satisfies the three ASTG conditions. Without loss of generality, for all u-hypercubes of $\mathcal{G}_X$, $t_{uv} > 0$ for all $v \in V$, according to Remark 3.4. To prove that G_X is an ASTG, we first construct a model $M = (I, K)$ with an ASTG $T_M = (X, S_M)$ on the same state space X, and then show that $T_M \cong G_X$.

1. The model $M = (I, K)$ is constructed in the following way.

(a) We choose an IG $I = (V, E, \vartheta, \varepsilon, \max)$ with a complete set of positive interactions, *i.e.*, for all $v, u \in V$, $(v, u) \in E$ with $\varepsilon(v, u) = +$. The function $\max$ can be extracted from the state space X. Furthermore, for all $u, v \in V$, let $\vartheta(v, u) := t_{uv}$.

(b) For each $\omega \in \mathrm{Pre}(u) = V$, we want to show that, the set of states for u under resource ω, $X^{u,\omega} \in H(u)$.
All interactions in I are positive, thus $X^{u,\omega} = \{x \in X \mid \begin{array}{l} \forall v \notin \omega : x_v < \vartheta(v, u) \\ \forall v \in \omega : x_v \geq \vartheta(v, u) \end{array}\}$.
From $\mathcal{G}_X$ and $H(u)$, we can find a u-hypercube, $\mathscr{X}_u = \Pi_{v \in V} \mathfrak{X}_v$, where for all $v \in V$, $t_{uv} \in \{1, \ldots, \max_v\}$. If $v \notin \omega$, let $\mathfrak{X}_v = \{0, \ldots, t_{uv} - 1\}$, and if $v \in \omega$, let $\mathfrak{X}_v = \{t_{uv}, \ldots, \max_v\}$. Then $\mathscr{X}_u$ has the same set of states as $X^{u,\omega}$ and $\mathrm{dest}(u, X^{u,\omega}) = \mathrm{dest}(u, \mathscr{X}_u)$. In this case, $\mathscr{X}_u$ is termed as $\mathscr{X}_u^\omega$.

(c) Now we can assign a logical parameter function K proper values. For all $x \in \mathscr{X}_u^\omega$, let $K(u, \omega) := \mathrm{dest}(u, \mathscr{X}_u^\omega)$.

2. Now we want to show that, for all $x \in X$, for all $u \in V$, if $(x, x + \mathbf{e}_u) \in T_M$, then $(x, x + \mathbf{e}_u) \in \mathcal{G}_X$ and vice versa. In the constructed model $M = (I, K)$, for any $x \in X$ with $\omega = \mathrm{Res}_u(x)$, the following cases hold.

(a) If $(x, x + \mathbf{e}_u) \in T_M$, then:

i. $K(u, \omega) > x_u$;

ii. $X^{u,\omega} = \mathscr{X}_u^\omega$. For all $x \in \mathscr{X}_u^\omega$, $K(u, \omega) = \mathrm{dest}(u, x)$, thus $\mathrm{dest}(u, x) > x_u$.

This implies that $(x, x + \mathbf{e}_u) \in \mathcal{G}_X$.

(b) Similarly, if $(x, x + \mathbf{e}_u) \in \mathcal{G}_X$, with $x \in \mathscr{X}_u^\omega$, then:

i. $\mathrm{dest}(u, x) > x_u$;

ii. $K(u, \omega) = \mathrm{dest}(u, x)$, thus we have $K(u, \omega) > x_u$.

This implies that $(x, x + \mathbf{e}_u) \in T_M$.

Analogously, for $(x, x - \mathbf{e}_u) \in T_M$, it holds also $(x, x - \mathbf{e}_u) \in \mathcal{G}_X$, and vice versa.

Therefore, the ASTG T_M from the constructed model M is equivalent with $\mathcal{G}_X$, *i.e.*, $\mathcal{G}_X$ is an ASTG.

□

3.1.2 Compatible model condition

The three ASTG-conditions guarantee $\mathcal{G}_X$ to be an ASTG which admits a model $M = (I, K)$. However, they do not guarantee that the ASTG will admit models which satisfy specific model conditions. For example, an interaction in I can be assigned signs opposite to its K.

Definition 3.11 (Compatible model M) A model $M = (I, K)$ is *compatible* if it satisfies both the observability condition and the Snoussi-condition.

Definition 3.12 (Compatible logical parameter function K for an IG I) A logical parameter function K is called *compatible* with an IG I if the model $M = (I, K)$ is compatible.

In this section, we present an ASTG condition 4, which provides a way to judge whether or not an ASTG can admit a compatible model (see Definition 3.11). Before presenting this condition, a few lemmas are needed. The first Lemma 3.13 shows that a model which satisfies

the Snoussi-condition has an equivalent model which satisfies both the observability and the Snoussi-condition.

Lemma 3.13 For a model $M = (I, K)$ which satisfies the Snoussi-condition, there exists an equivalent model $\tilde{M} = (\tilde{I}, \tilde{K})$ which contains only the observable interactions of I and satisfies both the observability and the Snoussi-condition.

Proof. Let $I = (V, E, \varepsilon, \vartheta, \max)$. Assume that $M = (I, K)$ satisfies the Snoussi-condition. Thus, every interaction of I satisfies the Snoussi-condition. Therefore, for all $(v, u) \in E$, for all $\omega \in \mathrm{Pre}(u)\backslash\{v\}$, $K(u, \omega) \leq K(u, \omega \cup \{v\})$. A model $\tilde{M} = (\tilde{I}, \tilde{K})$ is constructed based on M. Firstly, $\tilde{I} = (V, \tilde{E}, \tilde{\varepsilon}, \tilde{\vartheta}, \max)$ is constructed to include all observable interactions of I. Start with $\tilde{E} = \emptyset$. Then, for all $(v, u) \in E$, repeat the following operations.

1. If there exists $\omega \in \mathrm{Pre}(u)\backslash\{v\}$, $K(u, \omega) < K(u, \omega \cup \{v\})$, which means (v, u) is observable in M, then let $\tilde{E} := \tilde{E} \cup \{(v, u)\}$, $\tilde{\varepsilon}(v, u) := \varepsilon(v, u)$ and $\tilde{\vartheta}(v, u) := \vartheta(v, u)$.

2. If for all $\omega \in \mathrm{Pre}(u)\backslash\{v\}$, $K(u, \omega) = K(u, \omega \cup \{v\})$, thus (v, u) is not observable in M, then $(v, u) \notin \tilde{E}$.

Until now, all observable interactions of I are added into $\tilde{I}$. Since $\tilde{E} \subseteq E$, $\tilde{K}$ can be also constructed from K by setting $\tilde{K}(u, \omega) := K(u, \omega)$, for all $u \in V$ and $\omega \subseteq \mathrm{Pre}(u)|_{\tilde{I}}$. So that, $\tilde{M} = (\tilde{I}, \tilde{K})$ is equivalent to M. The reason why $\tilde{K}$ can be constructed like this is given next.

a) For all $i \in \mathrm{Pre}(u)|_I \backslash\, \mathrm{Pre}(u)|_{\tilde{I}}$, for all $\omega \in \mathrm{Pre}_I(u)$, $K(u, \omega) = K(u, \omega \Delta \{i\})$.

b) For all $x \in X$, because $\tilde{E} \subseteq E$, for all $u \in V$, the resource $\mathrm{Res}_u(x)$ in $\tilde{I}$ is a subset of that in I. Some resources of u which only exist in I are subsets of the components from the unobservable interactions, *i.e.*, $\mathrm{Res}_u(x)|_{\tilde{I}} \subseteq \mathrm{Res}_u(x)|_I$ and $\mathrm{Res}_u(x)|_I \backslash\, \mathrm{Res}_u(x)|_{\tilde{I}} \subseteq \mathrm{Pre}(u)|_I \backslash\, \mathrm{Pre}(u)|_{\tilde{I}}$.

From how $\tilde{K}$ is constructed, for all $x \in X$, for all $u \in V$, it holds that $\tilde{K}(u, \mathrm{Res}_u(x)|_{\tilde{I}}) = K(u, \mathrm{Res}_u(x)|_{\tilde{I}}) = K(u, \mathrm{Res}_u(x)|_I)$. The two models generate the same ASTG, $\tilde{M}$ is equivalent with M.

Moreover, for all $v \in \mathrm{Pre}(u)$, for all $\zeta \subseteq \mathrm{Pre}(u)|_{\tilde{I}}\backslash\{v\}$, we have $\tilde{K}(u, \zeta) \leq \tilde{K}(u, \zeta \cup \{v\})$, which means $\tilde{M}$ also satisfies the Snoussi-condition.

In summary, $\tilde{I}$ keeps only those observable interactions from I. $\tilde{M}$ is equivalent with M and satisfies both the observability and Snoussi-condition. The lemma is proved. □

A u-hypercube has a unique destination value. For $v \in V$, the v-th dimension of a u-hypercube $\mathscr{X}_u$ is either $\{0, \ldots, t_{uv} - 1\}$ or $\{t_{uv}, \ldots, \max_v\}$. If $t_{uu} \neq 0$, a u-hypercube contains only part of the u-row. In order to determine the destination values of a full u-row, the u-hypercube that contains the other part of the u-row has to be found. To be general, a definition of *the complement of $\mathscr{X}_u$ in direction of v* is given, so that a complete v-row $\tau^v = (x^0, \ldots, x^{\max_v})$ is contained in these two u-hypercubes.

Definition 3.14 (Complement of $\mathscr{X}_u$ in the v-th dimension) Consider a u-hypercube $\mathscr{X}_u$ in the state space X, where for all $i \in V$, $\mathfrak{X}_i = \{0, \ldots, t_{ui} - 1\}$ or $\mathfrak{X}_i = \{t_{ui}, \ldots, \max_i\}$. The *complement of $\mathscr{X}_u$ in the v-th dimension* is a u-hypercube $\mathscr{X}_u^{c_v}$, where for all $i \neq v \in V$, $\mathfrak{X}_i^{c_v} = \mathfrak{X}_i$, and $\mathfrak{X}_v^{c_v} = X_v \backslash \mathfrak{X}_v$ with $X_v = \{0, \ldots, \max_v\}$. If $\mathfrak{X}_v = X_v$ in $\mathscr{X}_u$, then $\mathscr{X}_u^{c_v} = \emptyset$. The union of $\mathscr{X}_u$ and $\mathscr{X}_u^{c_v}$ is again u-hypercube, which is denoted by ${}^v\mathscr{Y}_u = \mathscr{X}_u \cup \mathscr{X}_u^{c_v}$ or simply $\mathscr{Y}_u$ if $v = u$.

Figure 3.11 illustrates the complement of a u-hypercube in the v-th dimension.

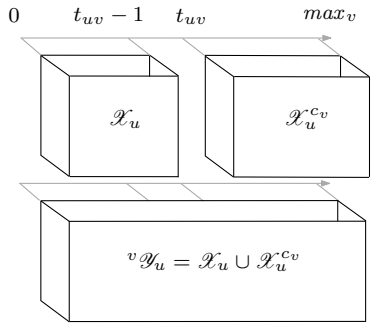


Figure 3.11: A u-hypercube $\mathscr{X}_u$ with $\mathfrak{X}_v = \{0, \ldots, t_{uv} - 1\}$; the complement of it in the v-th dimension, $\mathscr{X}_u^{c_v}$. The hypercube below is the union of $\mathscr{X}_u$ and $\mathscr{X}_u^{c_v}$, ${}^v\mathscr{Y}_u = \mathscr{X}_u \cup \mathscr{X}_u^{c_v}$.

For two u-hypercubes $\mathscr{X}_u$ and $\mathscr{X}_u^{c_u}$, the destination values in direction of u can be discussed via the u-rows. In an ASTG, if the destination value in direction u of $\mathscr{X}_u$ is not equal to that of $\mathscr{X}_u^{c_u}$, *i.e.*, $\mathrm{dest}(u, \mathscr{X}_u) \neq \mathrm{dest}(u, \mathscr{X}_u^{c_u})$, then there are either u-rows of type *pos* or u-rows of type *neg* in $\mathscr{Y}_u$. Lemma 3.15 discusses the three possible cases.

Lemma 3.15 Given an ASTG, consider a u-hypercube $\mathscr{X}_u$ with $\mathfrak{X}_u = \{0, \ldots, t_{uu} - 1\}$ and its complement in the u-th dimension $\mathscr{X}_u^{c_u}$. Then all u-rows in $\mathscr{Y}_u = \mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ are isomorphic to each other and there are the following three cases:

1. If $\mathrm{dest}(u, \mathscr{X}_u) < \mathrm{dest}(u, \mathscr{X}_u^{c_u})$, then the u-rows of $\mathscr{Y}_u$ are of *pos* type.
2. If $\mathrm{dest}(u, \mathscr{X}_u) > \mathrm{dest}(u, \mathscr{X}_u^{c_u})$, then the u-rows of $\mathscr{Y}_u$ are of *neg* type.
3. If $\mathrm{dest}(u, \mathscr{X}_u) = \mathrm{dest}(u, \mathscr{X}_u^{c_u})$, then the u-rows of $\mathscr{Y}_u$ are of *open* type.

Proof. Because $\mathscr{X}_u^{c_u}$ is the complement of $\mathscr{X}_u$ in the u-th dimension, $\mathfrak{X}_u^{c_u} = \{t_{uu}, \ldots, \max_u\}$. Let $\tau^u = \{x^0, \ldots, x^{\max_u}\}$ be a u-row in $\mathscr{Y}_u$ with $\{x^0, \ldots, x^{t_{uu}-1}\} \subseteq \mathscr{X}_u$ and $\{x^{t_{uu}}, \ldots, x^{\max_u}\} \subseteq \mathscr{X}_u^{c_u}$. Suppose $\tau_y^u = \{y^0, \ldots, y^{\max_u}\}$ is another u-row in $\mathscr{Y}_u$. For all $i \in \{0, \ldots, \max_u\}$, according to the ASTG condition 3, the following holds.

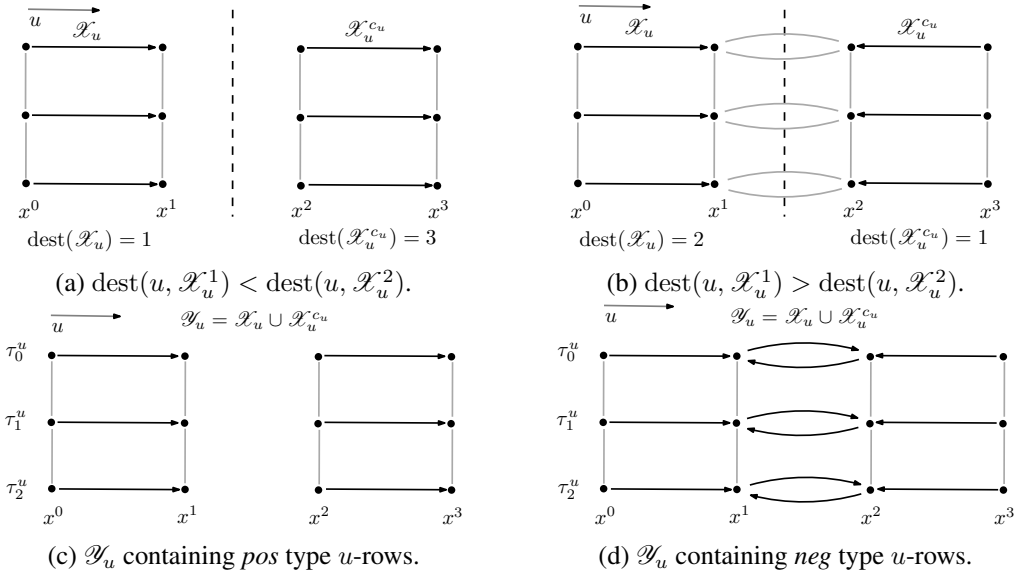
- if $y^i \in \mathscr{X}_u$, then $\mathrm{dest}(u, y^i) = \mathrm{dest}(u, \mathscr{X}_u) = \mathrm{dest}(u, x^i)$.
- if $y^i \in \mathscr{X}_u^{c_u}$, then $\mathrm{dest}(u, y^i) = \mathrm{dest}(u, \mathscr{X}_u^{c_u}) = \mathrm{dest}(u, x^i)$.

So τ_y^u is isomorphic to τ^u. Now there are the following cases:

1. If $\mathrm{dest}(u, \mathscr{X}_u) < \mathrm{dest}(u, \mathscr{X}_u^{c_u})$, then $\mathrm{dest}(u, \{x^0, \ldots, x^{t_{uu}} - 1\}) = \mathrm{dest}(u, \mathscr{X}_u) < \mathrm{dest}(u, \mathscr{X}_u^{c_u}) = \mathrm{dest}(u, \{x^{t_{uu}}, \ldots, x^{\max_u}\})$. According to Lemma 3.3, τ^u belongs to *pos* type.
2. If $\mathrm{dest}(u, \mathscr{X}_u) > \mathrm{dest}(u, \mathscr{X}_u^{c_u})$, then $\mathrm{dest}(u, \{x^0, \ldots, x^{t_{uu}} - 1\}) = \mathrm{dest}(u, \mathscr{X}_u) > \mathrm{dest}(u, \mathscr{X}_u^{c_u}) = \mathrm{dest}(u, \{x^{t_{uu}}, \ldots, x^{\max_u}\})$. According to Lemma 3.3, τ^u belongs to *neg* type.
3. If $\mathrm{dest}(u, \mathscr{X}_u) = \mathrm{dest}(u, \mathscr{X}_u^{c_u})$, then for each u-row in $\mathscr{Y}_u$, $\mathrm{dest}(u, \{x^0, \ldots, x^{t_{uu}} - 1\}) = \mathrm{dest}(u, \{x^{t_{uu}}, \ldots, x^{\max_u}\})$. According to Lemma 3.3, τ^u belongs to *open* type.

□

Figure 3.12 illustrates the first two cases of Lemma 3.15.



(a) $\mathrm{dest}(u, \mathscr{X}_u^1) < \mathrm{dest}(u, \mathscr{X}_u^2)$.

(b) $\mathrm{dest}(u, \mathscr{X}_u^1) > \mathrm{dest}(u, \mathscr{X}_u^2)$.

(c) $\mathscr{Y}_u$ containing *pos* type u-rows.

(d) $\mathscr{Y}_u$ containing *neg* type u-rows.

Figure 3.12: (a) and (c), $\mathscr{Y}_u = \mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ contains u-rows of type *pos*. (b) and (d), $\mathscr{Y}_u = \mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ contains u-rows of type *neg*.

According to Lemma 3.15, if $\mathrm{dest}(u, \mathscr{X}_u) \neq \mathrm{dest}(u, \mathscr{X}_u^{c_u})$ in an ASTG, then the u-rows contained in $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ are of type *pos* or *neg*. From Proposition 2.10, if an ASTG contains a u-row of *pos* or *neg* type, then in all models of the ASTG, there exists a self-loop on u.

In an ASTG $\mathcal{G}_X$, if $\mathrm{dest}(u, \mathscr{X}_u) \neq \mathrm{dest}(u, \mathscr{X}_u^{c_v})$, for some $v \neq u \in V$, then there exists an influence from v to u in all possible models of $\mathcal{G}_X$. Lemma 3.16 will show that the destination values of this pair of hypercubes $\mathscr{X}_u$ and $\mathscr{X}_u^{c_v}$, are closely related to the logical parameters of u under a pair of resources differing in v, for all possible models of the ASTG.

Lemma 3.16 Given an ASTG $\mathcal{G}_X$, consider a u-hypercube $\mathscr{X}_u$ with $\mathfrak{X}_v = \{0, \ldots, t_{uv} - 1\}$, and its complement in the v-th dimension $\mathscr{X}_u^{c_v}$, $v \neq u$. Let $\mathscr{Y}_u = \mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ and $\mathscr{Y}_u^{c_v} = (\mathscr{X}_u \cup \mathscr{X}_u^{c_u})^{c_v} = \mathscr{X}_u^{c_v} \cup \mathscr{X}_u^{c_u c_v}$.

(a) If $\mathrm{dest}(u, \mathscr{X}_u) < \mathrm{dest}(u, \mathscr{X}_u^{c_v})$, then there exist two states $x \in \mathscr{Y}_u$ and $\tilde{x} \in \mathscr{Y}_u^{c_v}$, where $\tilde{x}_k = x_k$, for all $k \neq v \in V$, such that in all possible models $M = (I, K)$ of $\mathcal{G}_X$, $K(u, \mathrm{Res}_u(x)) < K(u, \mathrm{Res}_u(\tilde{x}))$ with $\mathrm{Res}_u(x) = \mathrm{Res}_u(\tilde{x})\Delta\{v\}$.

(b) If $\mathrm{dest}(u, \mathscr{X}_u) > \mathrm{dest}(u, \mathscr{X}_u^{c_v})$, then there exist two states $x \in \mathscr{Y}_u$ and $\tilde{x} \in \mathscr{Y}_u^{c_v}$, where $\tilde{x}_k = x_k$, for all $k \neq v \in V$, such that in all possible models $M = (I, K)$ of $\mathcal{G}_X$, $K(u, \mathrm{Res}_u(x)) > K(u, \mathrm{Res}_u(\tilde{x}))$ with $\mathrm{Res}_u(x) = \mathrm{Res}_u(\tilde{x})\Delta\{v\}$.

The positions of all mentioned u-hypercubes are shown in Figure 3.13.

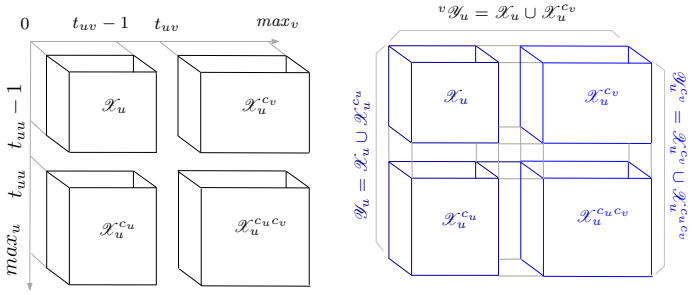


Figure 3.13: u-hypercube $\mathscr{X}_u$, its complements in the u-th and v-th dimension, together with $\mathscr{Y}_u$, ${}^v\mathscr{Y}_u$ and $\mathscr{Y}_u^{c_v}$.

Proof. (a) and (b) are two opposite cases on $\mathrm{dest}(u, \mathscr{X}_u)$ and $\mathrm{dest}(u, \mathscr{X}_u^{c_v})$. We will only give the proof of case (a), because case (b) can be proved analogously. Depending on $\mathfrak{X}_u$, there are two ways for finding states $x \in \mathscr{Y}_u$ and $\tilde{x} \in \mathscr{Y}_u^{c_v}$, such that $\delta(u, x) < \delta(u, \tilde{x})$. Given a u-row τ^u from $\mathscr{Y}_u$, there always exists a u-row $\tilde{\tau}^u \subseteq \mathscr{Y}_u^{c_v}$, such that the states of τ^u differ from the states of $\tilde{\tau}^u$ only in the v-th component.

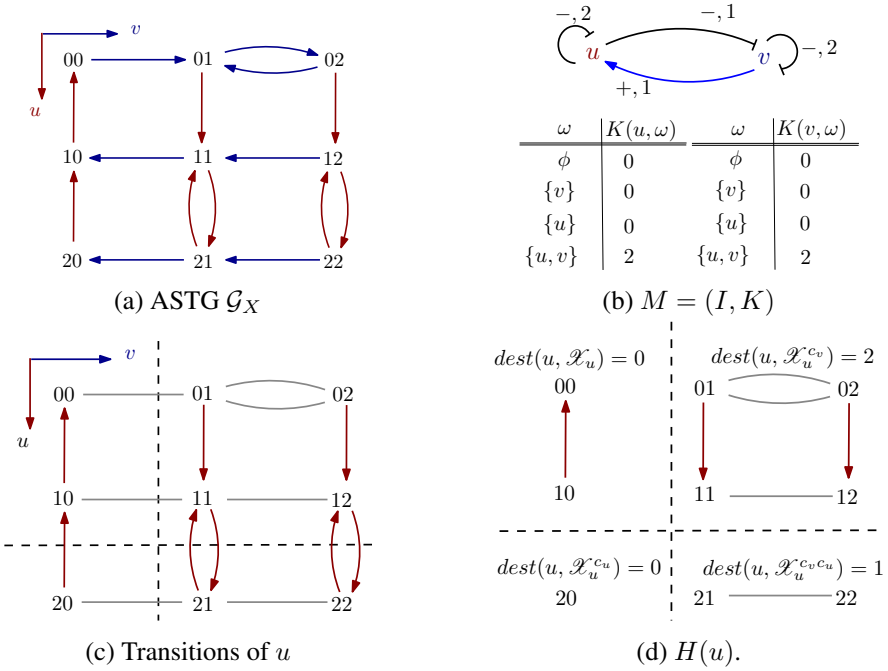
1. If $\mathfrak{X}_u = \mathfrak{X}_u^{c_v} = \{0, \ldots, t_{uu} - 1\}$, we take the states $x \in \tau^u$, $\tilde{x} \in \tilde{\tau}^u$ with $x_u = \tilde{x}_u = \mathrm{dest}(u, \mathscr{X}_u)$, and for all $k \neq v \in V$, $\tilde{x}_k = x_k$. According to the definition of the destination value, $\mathrm{dest}(u, \mathscr{X}_u)$ is the smallest value on τ^u where $\delta(u, x) \neq 1$, *i.e.*, $\delta(u, x) < 1$. Since $\mathrm{dest}(u, \mathscr{X}_u^{c_v}) > \mathrm{dest}(u, \mathscr{X}_u)$, we have $\delta(u, \tilde{x}) = 1$. This gives $\delta(u, x) < \delta(u, \tilde{x})$.

2. If $\mathfrak{X}_u = \mathfrak{X}_u^{c_v} = \{t_{uu}, \ldots, \max_u\}$, we take two states $x \in \tau^u$, $\tilde{x} \in \tilde{\tau}^u$ with $x_u = \tilde{x}_u = \mathrm{dest}(u, \mathscr{X}_u^{c_v})$, and for all $k \neq v \in V$, $\tilde{x}_k = x_k$. According to the definition of the destination value, $\mathrm{dest}(u, \mathscr{X}_u^{c_v})$ is the biggest value on $\tilde{\tau}^u$ with $\delta(u, \tilde{x}) \neq -1$, *i.e.*, $\delta(u, \tilde{x}) \geq 0$. Since $\mathrm{dest}(u, \mathscr{X}_u) < \mathrm{dest}(u, \mathscr{X}_u^{c_v})$, we have $\delta(u, x) = -1$. This also gives $\delta(u, x) < \delta(u, \tilde{x})$.

Therefore, for all possible models $M = (I, K)$ for $\mathcal{G}_X$, on these two states x and $\tilde{x}$, $\mathrm{sgn}(K(u, \mathrm{Res}_u(x)) - x_u) = \delta(u, x) < \delta(u, \tilde{x}) = \mathrm{sgn}(K(u, \mathrm{Res}_u(\tilde{x})) - \tilde{x}_u)$, since $x_u = \tilde{x}_u$, $K(u, \mathrm{Res}_u(x)) < K(u, \mathrm{Res}_u(\tilde{x}))$. Moreover, because $x \in \mathscr{Y}_u$ and $\tilde{x} \in \mathscr{Y}_u^{c_v}$, $\mathrm{Res}_u(x) = \mathrm{Res}_u(\tilde{x}) \Delta \{v\}$. Thus, the lemma is proved. □

According to Lemma 3.16, if $\mathrm{dest}(u, \mathscr{X}_u) \neq \mathrm{dest}(u, \mathscr{X}_u^{c_v})$ in an ASTG $\mathcal{G}_X$, then $(v, u) \in E$ in all models of $\mathcal{G}_X$.

Figure 3.14 illustrates Lemma 3.16. In Figure 3.14d, $\mathrm{dest}(u, \mathscr{X}_u) = 0 < \mathrm{dest}(u, \mathscr{X}_u^{c_v}) = 2$. Consider the states $x = 00$ and $\tilde{x} = 02$, then for all models of $\mathcal{G}_X$, $K(u, \mathrm{Res}_u(00)) < K(u, \mathrm{Res}_u(02))$.



ω	$K(u,\omega)$	ω	$K(v,\omega)$
ϕ	0	ϕ	0
$\{v\}$	0	$\{v\}$	0
$\{u\}$	0	$\{u\}$	0
$\{u,v\}$	2	$\{u,v\}$	2

(a) ASTG $\mathcal{G}_X$

(b) $M = (I, K)$

(c) Transitions of u

(d) $H(u)$.

Figure 3.14: (a) ASTG $\mathcal{G}_X$. (b) A model $M = (I, K)$ of $\mathcal{G}_X$. (c) State transitions in direction of u. (d) All u-hypercubes $H(u)$, and the destination values of each u-hypercube.

Now we are ready to introduce ASTG condition 4 for characterising ASTGs that admit a compatible model.

ASTG condition 4 (Compatible model condition) Let $\mathcal{G}_X$ be a graph based on X satisfying the ASTG conditions 1, 2 and 3. $\mathcal{G}_X$ admits a compatible model if and only if, for **no** components $u, v \in V$ (u can be the same as v), there exist 4 u-hypercubes $\mathscr{X}_u^1, \mathscr{X}_u^{1c_v}, \mathscr{X}_u^2, \mathscr{X}_u^{2c_v}$ with $\mathfrak{X}_v^1 = \mathfrak{X}_v^2 = \{0, \ldots, t_{uv} - 1\}$, such that both (1) and (2) are true.

(1) $\mathrm{dest}(u, \mathscr{X}_u^1) < \mathrm{dest}(u, \mathscr{X}_u^{1c_v})$.

(2) $\mathrm{dest}(u, \mathscr{X}_u^2) > \mathrm{dest}(u, \mathscr{X}_u^{2c_v})$.

Theorem 3.17 (Necessity) If an ASTG $T = (X, S)$ satisfies the ASTG condition 4, the it admits a compatible model.

Proof. To prove necessity, an arbitrary ASTG $\mathscr{G}_X$ which violates the compatible model condition is given. Then an incompatibility position in $\mathcal{G}_X$ is found, for all models that $\mathcal{G}_X$ can have.

Let $\mathcal{G}_X$ be an ASTG which violates the compatible model condition, then there exists such two u-hypercubes $\mathscr{X}_u^1$ and $\mathscr{X}_u^2$ in $\mathcal{G}_X$, where $\mathfrak{X}_v^1 = \mathfrak{X}_v^2$, such that

$$\mathrm{dest}(u, \mathscr{X}_u^1) < \mathrm{dest}(u, \mathscr{X}_u^{1c_v}) \text{ and } \mathrm{dest}(u, \mathscr{X}_u^2) > \mathrm{dest}(u, \mathscr{X}_u^{2c_v}).$$

Figure 3.15 illustrates how these u-hypercubes look like. Now we want to find an incompatibility position in $\mathcal{G}_X$.

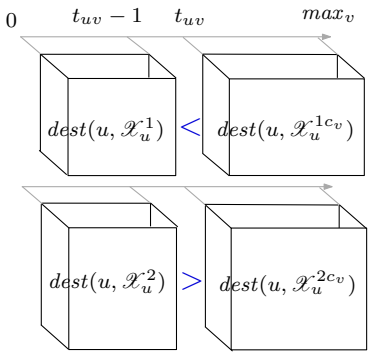


Figure 3.15: Two pairs of u-hypercubes in $\mathcal{G}_X$, $\mathscr{X}_u^1$ and $\mathscr{X}_u^{1c_v}$, $\mathscr{X}_u^2$ and $\mathscr{X}_u^{2c_v}$, where $\mathfrak{X}_v^1 = \mathfrak{X}_v^2 = \{0, \ldots, t_{uv} - 1\}$, $\mathfrak{X}_v^{1c_v} = \mathfrak{X}_v^{2c_v} = \{t_{uv}, \ldots, \max_v\}$.

Depending on whether v equals to u, there are two cases.

1. $v = u$. According to Lemma 3.15, for all the models of the ASTG which satisfy the Snoussi-condition, $\mathrm{dest}(u, \mathscr{X}_u^1) < \mathrm{dest}(u, \mathscr{X}_u^{1c_v})$ implies the existence of a positive self-loop (u, u), and $\mathrm{dest}(u, \mathscr{X}_u^2) > \mathrm{dest}(u, \mathscr{X}_u^{2c_v})$ the existence of a negative self-loop.

   From Remark 2.13, in an ASTG, for a component $u \in V$, if there are both *pos* and *neg* u-rows, then no model of the ASTG satisfies the Snoussi-condition. Therefore, in all the models of $\mathcal{G}_X$ the Snoussi-condition is violated in the interaction (u, u).

2. $v \neq u$. According to Lemma 3.16, one can find two pairs of states $x \in \mathscr{Y}_u^1, \tilde{x} \in \mathscr{Y}_u^{1c_v}$ and $y \in \mathscr{Y}_u^2, \tilde{y} \in \mathscr{Y}_u^{2c_v}$, such that for all models of $\mathcal{G}_X$, the following condition holds for the logical parameters:

   (a) $K(u, \mathrm{Res}_u(x)) < K(u, \mathrm{Res}_u(\tilde{x}))$ where $\mathrm{Res}_u(x) = \mathrm{Res}_u(\tilde{x}) \Delta \{v\}$;

(b) $K(u, \mathrm{Res}_u(y)) > K(u, \mathrm{Res}_u(\tilde{y}))$ where $\mathrm{Res}_u(y) = \mathrm{Res}_u(\tilde{y})\Delta\{v\}$.

The range in dimension v of $\mathscr{Y}_u^1$ and $\mathscr{Y}_u^2$ is $\{0, \ldots, t_{uv} - 1\}$, and the range of $\mathscr{Y}_u^{1c_v}$ and $\mathscr{Y}_u^{2c_v}$ is $\{t_{uv}, \ldots, \max_v\}$. Therefore, if $\varepsilon(v, u) = +$, then $v \in \mathrm{Res}_u(\tilde{x}) \cap \mathrm{Res}_u(\tilde{y})$, and Case 2b violates the Snoussi-condition. If $\varepsilon(v, u) = -$, then $v \in \mathrm{Res}_u(x) \cap \mathrm{Res}_u(y)$, then Case 2a violates the Snoussi-condition. Thus, both Case 2a and 2b lead to a violation of the Snoussi-condition in the interaction (v, u).

Therefore, if $\mathcal{G}_X$ violates the compatible model condition, then $\mathcal{G}_X$ involves an inherent violation of the Snoussi-condition in all its models. Analogously, for the other case $\mathrm{dest}(u, \mathscr{X}_u^1) > \mathrm{dest}(u, \mathscr{X}_u^{1c_v})$ and $\mathrm{dest}(u, \mathscr{X}_u^2) < \mathrm{dest}(u, \mathscr{X}_u^{2c_v})$, the same incompatibility can be found in all models of G_X. In conclusion, the compatible model condition has to be satisfied for an ASTG in order to admit a compatible model. □

Theorem 3.18 (Sufficiency) If an ASTG $T = (X, S)$ admits a compatible model, then it satisfies the ASTG condition 4.

Proof. To prove sufficiency, we show that for any ASTG $\mathcal{G}_X$ satisfying condition 4, a compatible model can be found.

Let $\mathcal{G}_X$ be an ASTG which satisfies the compatible model condition. According to Remark 3.4, for all $u, i \in V$, one can find such $t_{ui} > 0$, such that for all u-hypercubes in $H(u)$, $\mathcal{G}_X$ still satisfies the ASTG condition 3. Now we want to find a compatible model for $\mathcal{G}_X$.

1. First we construct a model $M = (I, K)$ for $\mathcal{G}_X$ like in the proof for Theorem 3.10, such that $I = (V, E, \varepsilon, \vartheta, \max)$ has a complete set of positive interactions, *i.e.*, for all $v, u \in V$, $(v, u) \in E$ with $\varepsilon(v, u) = +$, and the logical parameter function K takes the destination values of all hypercubes of $\mathcal{G}_X$. In I, for each interaction $(v, u) \in E$, $\vartheta(v, u) > 0$. For all $u \in V$, each u-hypercube $\mathscr{X}_u \in H(u)$ is related to a resource $\omega \subseteq \mathrm{Pre}(u)$ by $\mathrm{Res}_u(\mathscr{X}_u) = \omega$, denoted by $\mathscr{X}_u^\omega$ where $K(u, \omega) = \mathrm{dest}(u, \mathscr{X}_u^\omega)$, and $\mathrm{ASTG}(M) \cong \mathcal{G}_X$.

2. Next, based on M, an equivalent model $\tilde{M} = (\tilde{I}, \tilde{K})$ is constructed.

   (a) Let $\tilde{I} = (V, E, \tilde{\varepsilon}, \vartheta, \max)$.

   For all u, $v \in V$, for all pairs of u-hypercubes $\mathscr{X}_u^1$ and $\mathscr{X}_u^2$ which differ only in v, *i.e.*, $\mathfrak{X}_v^1 = \{0, \ldots, t_{uv} - 1\}$, $\mathfrak{X}_v^2 = \{t_{uv}, \ldots, \max_v\}$ and for all $i \neq v \in V$, $\mathfrak{X}_i^1 = \mathfrak{X}_i^2$, there are the following two cases according to the compatible model condition.

   i. If for all pairs $\mathscr{X}_u^1, \mathscr{X}_u^2$ we have $\mathrm{dest}(u, \mathscr{X}_u^1) \leq \mathrm{dest}(u, \mathscr{X}_u^2)$, then let $\tilde{\varepsilon}(v, u) := +$.

   ii. Otherwise, if for all pairs $\mathscr{X}_u^1, \mathscr{X}_u^2$ we have $\mathrm{dest}(u, \mathscr{X}_u^1) \geq \mathrm{dest}(u, \mathscr{X}_u^2)$, and there exists at least a pair of u-hypercubes $\tilde{\mathscr{X}}_u^1, \tilde{\mathscr{X}}_u^2$ in $\mathcal{G}_X$ such that $\mathrm{dest}(u, \tilde{\mathscr{X}}_u^1) > \mathrm{dest}(u, \tilde{\mathscr{X}}_u^2)$, then let $\tilde{\varepsilon}(v, u) := -$.

   (b) Once $\tilde{I}$ has been obtained, $\tilde{K}$ can be constructed from K. For each $u \in V$, $\mathrm{Pre}(u)|_{\tilde{I}} = \mathrm{Pre}(u)|_I$. Let $\mathrm{Pre}(u)^- := \{v \in V \mid \tilde{\varepsilon}(v, u) = -\}$.

   For each $x \in X$, it holds that $\mathrm{Res}_u(x)|_{\tilde{I}} = \mathrm{Res}_u(x)|_I \Delta \mathrm{Pre}(u)^-$. For all $\zeta \subseteq \mathrm{Pre}(u)$, let $\tilde{K}(u, \zeta) := K(u, \zeta \Delta \mathrm{Pre}(u)^-)$.

   Since $\tilde{K}$ is constructed based on K and it still keeps the destination values, *i.e.*, for all $u \in V$, for all $x \in X$, $\tilde{K}(u, \mathrm{Res}_u(x)\ |_{\tilde{I}}) = \mathrm{dest}(u, x) = K(u, \mathrm{Res}_u(x)|_I)$, we have $\mathrm{ASTG}(\tilde{M}) = \mathcal{G}_X$. $\tilde{M}$ is an equivalent model to M.

3. Now we will show $\tilde{M}$ satisfies the Snoussi-condition. From how $\tilde{M}$ is constructed, for all v, $u \in V$, for all pairs of u-hypercubes $\mathscr{X}_u^1$ and $\mathscr{X}_u^2$ which differ only in v, *i.e.*, $\mathfrak{X}_v^1 = \{0, \ldots, t_{uv} - 1\}$, $\mathfrak{X}_v^2 = \{t_{uv}, \ldots, \max_v\}$ and for all $i \neq v \in V$, $\mathfrak{X}_i^1 = \mathfrak{X}_i^2$, the following cases hold.

   (a) If $\varepsilon(v, u) = +$, then $\mathrm{Res}_u(\mathscr{X}_u^2)|_{\tilde{I}} = \mathrm{Res}_u(\mathscr{X}_u^1)|_{\tilde{I}} \cup \{v\}$. In this case, it holds always $\mathrm{dest}(u, \mathscr{X}_u^1) \leq \mathrm{dest}(u, \mathscr{X}_u^2)$, which means $\tilde{K}(u, \mathrm{Res}_u(\mathscr{X}_u^1)|_{\tilde{I}}) \leq \tilde{K}(u, \mathrm{Res}_u(\mathscr{X}_u^1)|_{\tilde{I}} \cup \{v\})$.

   (b) If $\varepsilon(v, u) = -$, then $\mathrm{Res}_u(\mathscr{X}_u^1)|_{\tilde{I}} = \mathrm{Res}_u(\mathscr{X}_u^2)|_{\tilde{I}} \cup \{v\}$. In this case, it holds always $\mathrm{dest}(u, \mathscr{X}_u^1) \geq \mathrm{dest}(u, \mathscr{X}_u^2)$, which means $\tilde{K}(u, \mathrm{Res}_u(\mathscr{X}_u^2)|_{\tilde{I}} \cup \{v\}) \geq \tilde{K}(u, \mathrm{Res}_u(\mathscr{X}_u^2)|_{\tilde{I}})$.

   For all $u \in V$, each u-hypercube $\mathscr{X}_u \in H(u)$ is also related to a resource $\varsigma \subseteq \mathrm{Pre}(u)|_{\tilde{I}}$ by $\mathrm{Res}_u|_{\tilde{I}}(\mathscr{X}_u) = \varsigma$ where $\tilde{K}(u, \varsigma) = \mathrm{dest}(u, \mathscr{X}_u)$.

   From (a) and (b), it follows that $\tilde{K}(u, \omega) \leq \tilde{K}(u, \omega \cup \{v\})$. Every interaction of M satisfies the Snoussi-condition. According to Remark 2.5, $\tilde{M}$ satisfies the Snoussi-condition.

From above, $\tilde{M}$ is a Snoussi-model for $\mathcal{G}_X$. According to Lemma 3.13, for a Snoussi-model, one can always find an equivalent model which satisfies both the observability and Snoussi-conditions, thus the sufficiency is proved. □

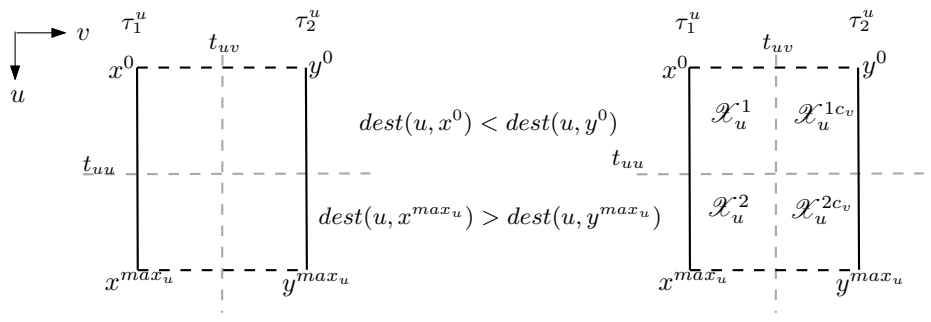
The following Corollary 3.19 is a direct application of ASTG condition 4 in terms of the extremal states.

Corollary 3.19 Consider an ASTG T and two extremal u-rows $\tau_1^u = (x^0, \ldots, x^{\max_u})$ and $\tau_2^u = (y^0, \ldots, y^{\max_u})$ where for all $i \in \{0, \ldots, \max_u\}$, $x_v^i = 0$ and $y^i = x^i + \max_v \mathbf{e}_v$. Assume that $\mathrm{dest}(u, x^0) \neq \mathrm{dest}(u, y^0)$ and $\mathrm{dest}(u, x^{\max_u}) \neq \mathrm{dest}(u, y^{\max_u})$. If $\mathrm{sgn}(\mathrm{dest}(u, x^0) - \mathrm{dest}(u, y^0)) \neq \mathrm{sgn}(\mathrm{dest}(u, x^{\max_u}) - \mathrm{dest}(u, y^{\max_u}))$, then T does not satisfy ASTG condition 4.

Proof. T satisfies u-hypercube ASTG condition 3, according to Theorem 3.9. For some $u, v \in V$, for the four extremal states x^0 and $x^{\max_u}$ from τ_1^u, and y^0 and $y^{\max_u}$ from τ_2^u, assume without loss of generality $\mathrm{dest}(u, x^0) < \mathrm{dest}(u, y^0)$ and $\mathrm{dest}(u, x^{\max_u}) > \mathrm{dest}(u, y^{\max_u})$.

1. $\mathrm{dest}(u, x^0) < \mathrm{dest}(u, y^0)$ implies $t_{uv} > 0$ for all u-hypercubes. Therefore, x^0, y^0 are in two different u-hypercubes, and also $x^{\max_u}$ and $y^{\max_u}$ are in two different u-hypercubes.

2. $\mathrm{dest}(u, x^0) < \mathrm{dest}(u, y^0)$ together with $\mathrm{dest}(u, x^{\max_u}) > \mathrm{dest}(u, y^{\max_u})$ implies that at least one of $\mathrm{dest}(u, x^0) \neq \mathrm{dest}(u, x^{\max_u})$ and $\mathrm{dest}(u, y^0) \neq \mathrm{dest}(u, y^{\max_u})$ holds. This means, one of τ_1^u and τ_2^u is not of *open* type. Therefore, $t_{uu} > 0$ for all u-hypercubes.



3. Thus, these 4 extremal states are located in four u-hypercubes. Let these 4 u-hypercubes

be $\mathscr{X}_u^1 \ni x^0$, $\mathscr{X}_u^{1c_v} \ni y^0$, $\mathscr{X}_u^2 \ni x^{\max_u}$, and $\mathscr{X}_u^{2c_v} \ni y^{\max_u}$. From the u-hypercube condition, the destination value of all states in a u-hypercube is the same. This gives the following.

- $\mathrm{dest}(u, \mathscr{X}_u^1) = \mathrm{dest}(u, x^0) < \mathrm{dest}(u, y^0) = \mathrm{dest}(u, \mathscr{X}_u^{1c_v})$
- $\mathrm{dest}(u, \mathscr{X}_u^2) = \mathrm{dest}(u, x^{\max_u}) > \mathrm{dest}(u, y^{\max_u}) = \mathrm{dest}(u, \mathscr{X}_u^{2c_v})$

By the assumption of the lemma, it holds that $\mathrm{dest}(u, \mathscr{X}_u^1) < \mathrm{dest}(u, \mathscr{X}_u^{1c_v})$ and $\mathrm{dest}(u, \mathscr{X}_u^2) > \mathrm{dest}(u, \mathscr{X}_u^{2c_v})$, which contradicts the ASTG condition 4.

□

3.1.3 Enumerating asynchronous state transition graphs

Based on the three necessary and sufficient ASTG conditions, we now derive rules for constructing all ASTGs on a given state space. Proposition 3.25 explains how to construct ASTGs on a multi-valued state space. As a special case, Proposition 3.26 considers the ASTGs on a Boolean state space. Proposition 3.30 shows that all possible ASTGs can be obtained in this way.

A u-row of *pos* or *neg* type indicating $\vartheta(u, u) = t$ is also called a u-row of threshold t (see Definition 2.11). In order to give more information on the u-rows than just their row type, we define the *set of eligible u-row structures of threshold value t*.

Definition 3.20 (Set of eligible u-row structures of threshold value t) Given $t \in \{0, \ldots, \max_u\}$ the *set $\{\tau^u\}_{elg}^t$ of eligible u-row structures of threshold value t* contains:

1. all *pos* type of u-rows of threshold t if $t > 0$;
2. all *neg* type of u-rows of threshold t if $t > 0$;
3. all *open* type of u-rows.

For a Boolean component u, Figure 3.16 shows the eligible set of u-row structures of threshold value 1. According to Proposition 2.10, there exist only four possible u-row structures in an ASTG on a Boolean state space, as shown in Figure 3.16.

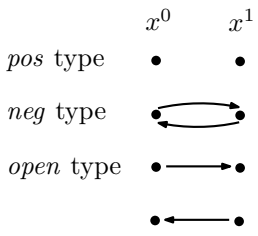


Figure 3.16: Set $\{\tau^u\}_{elg}^1$ of eligible u-row structures of threshold value 1 in the Boolean case.

Another example is given by the 9 u-rows in Figure 3.4 of Section 3.1.1, which shows the set of eligible u-row structures of threshold value 2, $\{\tau^u\}_{elg}^2$.

Lemma 3.21 For a given $t \in \{0, \ldots, \max_u\}$, the number of u-row structures in the set $\{\tau^u\}_{elg}^t$ can be calculated according to the row types. We have:

- $t(\max_u - t + 1)$ u-row structures of *pos* type, if $t > 0$.
- 1 u-row structure of *neg* type, if $t > 0$.

- $(\max_u + 1)$ u-row structures of *open* type.

Proof. From Proposition 2.10, given an ASTG, each u-row τ^u belongs to exactly one of the three row types, *pos*, *neg* and *open* type. Let the length of τ^u be the number of states, which is $(1 + \max_u)$.

1. If τ^u is of *pos* type with threshold $t \in \{1, \ldots, \max_u\}$, let $a \in \{0, \ldots, t-1\}$ and $b \in \{t, \ldots, \max_u\}$ be the destination values of states x^i with $i \in \{0, \ldots, t-1\}$, and states x^j with $j \in \{t, \ldots, \max_u\}$, respectively. Thus, there are t different choices for a, and $\max_u - t + 1$ choice for b. For every $t \in \{1, \ldots, \max_u\}$, the number of possible structures of τ^u is therefore $t(\max_u - t + 1)$.

2. If τ^u is of *neg* type with threshold $t \in \{1, \ldots, \max_u\}$, then the destination value of states x^i with $i \in \{0, \ldots, t-1\}$ is t, and of states x^j with $j \in \{t, \ldots, \max_u\}$ is $t - 1$. Thus, the row structure of τ^u is uniquely determined.

3. If τ^u is of *open* type, let $a \in \{0, \ldots, \max_u\}$ be the destination values of all states in τ^u. Thus, there are in total $\max_u + 1$ different row structures for τ^u.

□

The following proposition provides a combination of the three ASTG conditions:

Proposition 3.22 Given a graph $\mathscr{G}_X$ based on X, consider for each $u \in V$ a set of u-hypercubes $H(u)$ as defined in ASTG condition 3. If for each $u \in V$, all u-rows in each pair of u-hypercubes $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ are isomorphic and each u-row belongs to the set of eligible u-row structures of threshold value t_{uu}, then $\mathscr{G}_X$ is an ASTG.

Proof. This proposition can be proved using the three ASTG conditions. Let $H(u)$ be the set of u-hypercubes in $\mathscr{G}_X$ with $t_{ui} \in \{0, \ldots, \max_i\}$, for each $i \in V$ and for each $u \in V$.

1. By hypothesis, in $\mathscr{G}_X$, for all $u \in V$, each u-row belongs to the set of eligible u-row structures of threshold value t_{uu}. From Definition 3.20, $\{\tau^u\}_{elg}^{t_{uu}}$ contains all u-rows of *pos* and *neg* types at threshold value t_{uu}, and all u-rows of *open* type.

   (a) In the three types of u-rows (Proposition 2.10), a state can only transit to the direct neighbours of Hamming distance 1. Thus the ASTG condition 1 (*asynchronicity*) is satisfied.

   (b) In the three types of u-rows, each state x can be updated only by distance 1 in one direction. Thus, the ASTG condition 2 (*unitarity*) is satisfied.

2. Now we want to prove that the ASTG condition 3 (*u-hypercube*) also holds.

   (a) By hypothesis, all u-rows in each pair of u-hypercubes $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ are isomorphic. Take a u-row $\tau^u = \{x^0, \ldots, x^{\max_u}\}$ from $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$.

   (b) If $t_{uu} > 0$ and in $\mathscr{X}_u$, $\mathfrak{X}_u = \{0, \ldots, t_{uu} - 1\}$, then $x^0 \in \mathscr{X}_u$ and $x^{\max_u} \in \mathscr{X}_u^{c_u}$. According to Lemma 3.3, it holds that $\mathrm{dest}(u, x^0) = \cdots = \mathrm{dest}(u, x^{t_{uu}-1})$ and $\mathrm{dest}(u, x^{t_{uu}}) = \cdots = \mathrm{dest}(u, x^{\max_u})$. This holds for all u-rows in $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$.

   (c) If $t_{uu} = 0$ and in $\mathscr{X}_u$, $\mathfrak{X}_u = \{0, \ldots, \max_u\}$, then $x^0, x^{\max_u} \in \mathscr{X}_u$ and $\mathscr{X}_u^{c_u} = \emptyset$. According to Lemma 3.3, it holds that $\mathrm{dest}(u, x^0) = \cdots = \mathrm{dest}(u, x^{\max_u})$. This holds for all u-rows in $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$.

   Hence, $\mathscr{X}_u$ has a unique destination value $\mathrm{dest}(u, \mathscr{X}_u) = \mathrm{dest}(u, x^0)$, and the same for $\mathscr{X}_u^{c_u}$, $\mathrm{dest}(u, \mathscr{X}_u^{c_u}) = \mathrm{dest}(u, x^{\max_u})$. Therefore, ASTG condition 3 holds.

According to Theorem 3.17, $\mathscr{G}_X$ is an ASTG. Therefore, Proposition 3.22 is proved. □

Next we want to show that for a set of u-hypercubes $H(u)$ in a state space X, each u-hypercube contains at least one extremal state.

Lemma 3.23 Consider a set of u-hypercubes $H(u)$ in a state space X, let $t_{uv} \in \{0, \ldots, \max_v\}$ for all $v, u \in V$. Then each u-hypercube $\mathscr{X}_u \in H(u)$ contains at least one extremal state of X. Moreover, if for all $u, v \in V$, $t_{uv} > 0$, then each u-hypercube contains exactly one extremal state.

Proof. In a state space X, for each $u \in V$, each u-hypercube $\mathscr{X}_u \in H(u)$ is of the form $\mathscr{X}_u := \Pi_{i=1}^{n} \mathfrak{X}_i \subseteq X$ with $\mathfrak{X}_i = \{0, \ldots, t_{ui} - 1\}$ or $\mathfrak{X}_i = \{t_{ui}, \ldots, \max_i\}$, and $t_{ui} \in \{0, \ldots, \max_i\}$. We can find an extremal state x^{ext} by considering each dimension of $\mathscr{X}_u$. For all $v \in V$, there are the following two cases.

1. If $t_{uv} = 0$, then $\mathfrak{X}_v = \{0, \ldots, \max_v\}$, there exist two extremal states and x_v^{ext} can be both 0 and $\max_v$.
2. If $t_{uv} > 0$, then $\mathfrak{X}_v$ can be either $\{0, \ldots, t_{uv} - 1\}$ or $\{t_{uv}, \ldots, \max_v\}$ with x_v^{ext} being either 0 or $\max_v$, respectively.

From case 2, if for all $u, v \in V$, $t_{uv} > 0$, then x_v^{ext} is either 0 or $\max_v$, for each $v \in V$. Therefore, each u-hypercube $\mathscr{X}_u \in H(u)$ contains only one extremal state. □

Similarly, for a set of u-hypercubes $H(u)$ in a state space X, each u-hypercube and its complement in u-direction contain at least one extremal row.

Lemma 3.24 Consider a set of u-hypercubes $H(u)$ in a state space X, let $t_{uv} \in \{0, \ldots, \max_v\}$ for all $v, u \in V$. Each pair of u-hypercubes $\mathscr{X}_u, \mathscr{X}_u^{c_u} \in H(u)$ contains at least one extremal u-row of X. Moreover, if for all $u, v \in V$, $t_{uv} > 0$, then each pair of u-hypercubes contains exactly one extremal u-row.

Proof. In a state space X, for each $u \in V$, there exist three possible cases. In each of these cases, for each u-hypercube $\mathscr{X}_u \in H(u)$, one can find at least one extremal u-row in $\mathfrak{X}_u \cup \mathfrak{X}_u^{c_u}$.

1. For all $v \in V$, $t_{uv} > 0$. According to Lemma 3.23, each $\mathscr{X}_u$ contains exactly one extremal state. Therefore, for each pair of u-hypercubes $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$, $\mathfrak{X}_u \cup \mathfrak{X}_u^{c_u} = \{0, \ldots, \max_u\}$, only one extremal row is contained.
2. $t_{uu} = 0$ and for all $v \neq u \in V$, $t_{uv} > 0$. In this case, $\mathfrak{X}_u = \{0, \ldots, \max_u\}$, thus $\mathfrak{X}_u^{c_u} = \emptyset$. For each pair of u-hypercubes $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$, $\mathfrak{X}_u \cup \mathfrak{X}_u^{c_u} = \{0, \ldots, \max_u\}$. Similar to case 1, $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ contains only one extremal u-row.
3. $t_{uu} = 0$ and for some $v \in V$, $t_{uv} = 0$.

   In this case, $\mathfrak{X}_v$ is $\{0, \ldots, \max_v\}$. $\mathscr{X}_u$ contains two extremal states x^{ext_1}, x^{ext_2} with $x_v^{ext_1} = 0$ and $x_v^{ext_2} = \max_v$. In $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$, $\mathfrak{X}_v = \mathfrak{X}_v^{c_u} = \{0, \ldots, \max_v\}$. Therefore, $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ contains two extremal u-rows where one row contains x^{ext_1} and the other one contains x^{ext_2}.

□

Let τ_{ext}^u denote an extremal u-row. The following Proposition 3.25 shows that by arbitrarily choosing each extremal row from the set of eligible row structures for a certain t value, we can obtain an ASTG.

Proposition 3.25 Given a state space X, for each $u \in V$, let $t_{uu} \in \{0, \ldots, \max_u\}$. By choosing for each extremal u-row an arbitrary row structure from the set of eligible u-row structures of

threshold value t_{uu}, an ASTG can be defined.

Proof. To prove this proposition, we have to construct an ASTG from the arbitrarily chosen extremal rows.

Let t_{uv} be some value from $\{1, \ldots, \max_v\}$, for all $v \neq u \in V$, and let $H(u)$ be the corresponding set of u-hypercubes. For each $\mathscr{X}_u \in H(u)$, we can do the following:

1. If $t_{uu} > 0$, then each u-hypercube $\mathscr{X}_u$ contains only one extremal state, as proved in Lemma 3.23. If $t_{uu} = 0$, then $\mathscr{X}_u$ contains two extremal states x and y which are in the same u-row because $x_v = y_v$ for all $v \neq u \in V$, $x_u = 0$ and $y_u = \max_u$.

2. Each pair of u-hypercubes $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ contains an extremal u-row, according to Lemma 3.24. Since $t_{uv} > 0$ for all $v \neq u \in V$, $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ contains exactly one extremal u-row. Let the extremal u-row be τ_{ext}^u, and its row structure be arbitrarily chosen from the set of eligible u-row structures of threshold value t_{uu}.

3. In each pair of u-hypercubes $\mathscr{X}_u \cup \mathscr{X}_u^{c_u}$, let all u-rows be isomorphic to the extremal u-row τ_{ext}^u.

According to Proposition 3.22, the resulting graph based on X is an ASTG. □

On a Boolean state space, Proposition 3.26 shows how to construct an ASTG using the 4 eligible u-row structures from Figure 3.16.

Proposition 3.26 An ASTG on a Boolean state space can be constructed by arbitrarily choosing for each row a row structure from the set of eligible Boolean u-row structures (in total 4).

Proof. Let the Boolean state space be X_B. All possible 4 Boolean row structures are shown in Figure 3.16, denoted by $\{\tau^u\}_{elg}^1$. If each row structure in X_B is arbitrarily chosen from $\{\tau^u\}_{elg}^1$, then a complete graph based on X_B is constructed. For each $u \in V$, let $t_{uv} = 1$ for all $v \in V$. Then each state defines a u-hypercube. According to Proposition 3.22, the constructed graph based on X_B is an ASTG. □

As a special case of Proposition 3.25, Proposition 3.27 shows how to construct all ASTGs for a given IG I. With the threshold values of I for each $u \in V$, the state space X can be partitioned into a set of u-hypercubes. Here an ASTG for I means that there exists a model $M = (I, K)$ which can generate this ASTG, but does not have to satisfy any model conditions. Therefore, the interactions of I may not be visible or observable in M. Moreover, if an interaction does not exist in I, then the influence of this interaction is not present in the ASTG.

Proposition 3.27 Consider an IG $I = (V, E, \varepsilon, \vartheta, \max)$. For all $v, u \in V$, if $(v, u) \in E$, let $t_{uv} := \vartheta(v, u)$, otherwise let $t_{uv} := 0$. Thus a set of u-hypercubes $H(u)$ is defined. All ASTGs for I can be enumerated in the following way. For each $u \in V$, for each pair of u-hypercubes $\mathscr{Y}_u = \mathscr{X}_u \cup \mathscr{X}_u^{c_u}$ in $H(u)$, choose arbitrarily one row structure from the set $\{\tau^u\}_{elg}^{t_{uu}}$ of eligible u-row structures of threshold value t_{uu}, and let all u-rows in $\mathscr{Y}_u$ be isomorphic to the chosen row structure.

Proof. We will first show that the construction given in the proposition leads to an ASTG T which admits a model $M = (I, K)$. Then we will show that all possible ASTGs for I can be obtained in this way.

According to Proposition 3.22, each graph constructed according to the lemma is an ASTG T. We have to show that there exists a model $M = (I, K)$ for T. In each u-hypercube $\mathscr{X}_u \in H(u)$, we have for all states $x, y \in \mathscr{X}_u$ that $\mathrm{Res}_u(x) = \mathrm{Res}_u(y)$. For all $u, v \in V$, one of the following two cases holds:

1. If $(v,u) \in E$, then $t_{uv} > 0$ and $\mathrm{Res}_u(\mathscr{X}_u) = \mathrm{Res}_u(\mathscr{X}_u^{c_v})\Delta\{v\}$. Let $K(u, \mathrm{Res}_u(\mathscr{X}_u)$ $:= \mathrm{dest}(u, \mathscr{X}_u)$ and $K(u, \mathrm{Res}_u(\mathscr{X}_u^{c_v})) := \mathrm{dest}(u, \mathscr{X}_u^{c_v})$. Since the u-row structure in $\mathscr{Y}_u$ and $\mathscr{Y}_u^{c_v}$ is arbitrarily chosen from $\{\tau^u\}_{elg}^{t_{uu}}$, one of the following two cases holds.
   - If for some pair of u-hypercubes $\mathscr{X}_u$ and $\mathscr{X}_u^{c_v}$, $\mathrm{dest}(u, \mathscr{X}_u) \neq \mathrm{dest}(u, \mathscr{X}_u^{c_v})$, then $K(u, \mathrm{Res}_u(\mathscr{X}_u)) \neq K(u, \mathrm{Res}_u(\mathscr{X}_u^{c_v}))$. The interaction (v,u) exists in all models of the ASTG, according to Lemma 3.16.
   - If for all pair of u-hypercubes $\mathscr{X}_u$ and $\mathscr{X}_u^{c_v}$, $\mathrm{dest}(u, \mathscr{X}_u) = \mathrm{dest}(u, \mathscr{X}_u^{c_v})$. Then, for all such pairs of u-hypercubes, $K(u, \mathrm{Res}_u(\mathscr{X}_u)) = K(u, \mathrm{Res}_u(\mathscr{X}_u^{c_v}))$. The interaction (v,u) is not visible in all models of the ASTG, according to Lemma 3.16.
2. If $(v,u) \notin E$, then $t_{uv} = 0$. For each u-hypercube $\mathscr{X}_u \in H(u)$, $\mathfrak{X}_v = \{0, \ldots, \max_v\}$ and $\mathfrak{X}_v^{c_v} = \emptyset$. Therefore, $v \notin \mathrm{Res}_u(\mathscr{X}_u)$. Let $K(u, \mathrm{Res}_u(\mathscr{X}_u)) := \mathrm{dest}(u, \mathscr{X}_u)$, for each $\mathscr{X}_u \in H(u)$.
   - Let $\tilde{t}_{uv}$ be an arbitrary value from $\{1, \ldots, \max_v\}$, such that each u-hypercube $\mathscr{X}_u$ is divided into $\tilde{\mathscr{X}}_u$ with $\tilde{\mathfrak{X}}_v = \{0, \ldots, \tilde{t}_{uv} - 1\}$ and $\tilde{\mathscr{X}}_u^{c_v}$ with $\tilde{\mathfrak{X}}_v^{c_v} = \{\tilde{t}_{uv}, \ldots, \max_v\}$. The set of u-hypercubes $H(u)$ becomes $\tilde{H}(u)$ with $\tilde{t}_{uw} := \tilde{t}_{uw}$ for all $w \neq v \in V$.
   - $\mathrm{dest}(u, \tilde{\mathscr{X}}_u) = \mathrm{dest}(u, \mathscr{X}_u) = \mathrm{dest}(u, \tilde{\mathscr{X}}_u^{c_v})$. Therefore, the constructed ASTG still satisfies the ASTG condition 3 on $\tilde{H}(u)$. The interaction (v,u) is not visible for any model of the constructed ASTG, according to Lemma 3.16.

Similar to the proof of Theorem 3.10 where K is assigned the destination values from T, the ASTG of $M = (I, K)$ is T. Therefore, each graph constructed by the lemma is an ASTG for I.

Now assume that T is an arbitrary ASTG for I. Each u-row in T is a member of $\{\tau^u\}_{elg}^{t_{uu}}$. If the structure of a u-row is not contained in $\{\tau^u\}_{elg}^{t_{uu}}$, then either it has a different threshold value for (u,u), or it does not belong to any valid ASTG. Therefore, all possible ASTGs for I are constructed by the proposition. □

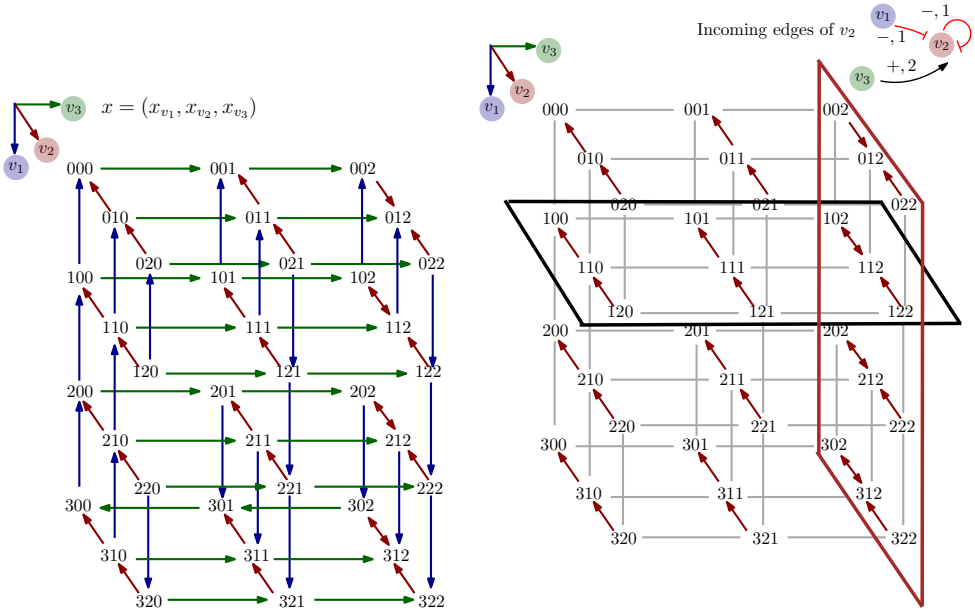
By Proposition 3.27, all ASTGs for a given IG can be constructed. However, some interactions of the IG are not visible in any model of some of the constructed ASTGs. For this reason, Remark 3.29 specifies how to construct all ASTGs for an IG, such that in all models of these ASTGs, each interaction of the IG is visible. First we define a *u-slice along v* as a group of parallel u-rows that differ only in their v values, see Figure 3.17 for two examples.

Definition 3.28 (u-slice along v) In a state space X, for u, $v \in V$, a *u-slice along v* is a set $\{\tau_0^u, \ldots, \tau_{\max_v}^u\}$ of neighbouring u-rows in the direction of v, where for all $x^i \in \tau_j^u$, x_k^i are the same for all $i \in \{0, \ldots, \max_u\}$ for all $k \neq u \neq v \in V$ and $x_v^i = j$.

In Proposition 3.27, the set $\{\tau^u\}_{elg}^{t_{uu}}$ contains all u-rows of *open* type. Since extremal u-rows can be arbitrarily chosen, interactions in I may not be visible in some of the ASTGs T for I or, to be more precise, in their corresponding models. An interaction (u,u) in I becomes invisible in an ASTG T for I if all u-rows in T are of *open* type. An interaction (v,u) in I becomes invisible in T if the u-row structures in every u-slice along v are isomorphic. The following Remark 3.29 explains how to construct all those ASTGs for a given IG I in which all interactions of I are visible.

Remark 3.29 To construct with Proposition 3.27 all ASTGs T for an IG $I = (V, E, \varepsilon, \vartheta, \max)$ such that in all models of T every interaction in I is visible, the following two constraints have to be satisfied.

1. For all $u \in V$, if $(u,u) \in E$, then at least one extremal u-row is of *pos* or *neg* type for threshold $\vartheta(u,u)$.



(a) An ASTG T. (b) v_2-slices.

Figure 3.17: (a) ASTG T from Figure 3.8. (b) One v_2-slice along v_1, and one along v_3. Each parallelogram contains one v_2-slice. At the top right is the incoming interactions of v_2 in the IG for T.

2. For all $v, u \in V$, if $(v, u) \in E$, at least one pair of extremal u-rows in the same u-slice along v have different structures.

By Lemma 3.15 and 3.16, Remark 3.29 guarantees that each interaction of I will be visible in all models of the resulting ASTGs.

Based on Proposition 3.27, the following proposition shows how to enumerate all ASTGs on a given state space X.

Proposition 3.30 All possible ASTGs on a state space X can be constructed by the following two steps.

1. For each $u \in V$, enumerate all possible values for $t(u) = (t_{ui})_{i \in V}$ with $t_{ui} \in \{0, \ldots, \max_i\}$,
2. For each $\mathbf{t} = (t(u))_{u \in V}$, construct all ASTGs, using Proposition 3.27.

Proof. By Step 1, all possible values for $t(u) = (t_{ui})_{i \in V}$ with $t_{ui} \in \{0, \ldots, \max_i\}$ for each $u \in V$ are enumerated. Therefore, all possible $\mathbf{t} = (t(u))_{u \in V}$ are obtained which specifies sets of u-hypercubes $H(u)$ for all $u \in V$.

Each $\mathbf{t} = (t(u))_{u \in V}$ can be transformed into an IG I by the following. For all $v, u \in V$, if $t_{uv} > 0$, then $(v, u) \in E$ and let $\vartheta(v, u) := t_{uv}$ and $\varepsilon(v, u) := +$. If $t_{uv} = 0$, then $(v, u) \notin E$. This way, the information given by $\mathbf{t} = (t(u))_{u \in V}$ is the same as the information given by an IG.

By step 2, for each $\mathbf{t} = (t(u))_{u \in V}$, one can construct all possible ASTGs using Proposition 3.27. Therefore, all possible ASTGs on X are constructed. □

While Proposition 3.30 allows enumerating all possible ASTGs in a state space X, not all of these ASTGs will admit compatible models, as explained by Remark 3.31.

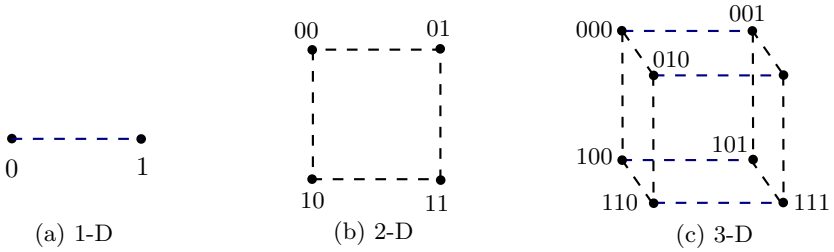
Remark 3.31 The ASTGs constructed in Proposition 3.30 do not always satisfy the ASTG condition 4.

Proof. The ASTGs on a state space X in Proposition 3.30 are constructed from eligible u-row structures of the same threshold value, for all $u \in V$ and for all extremal u-rows in X. For some of the constructed ASTGs and some $u \in V$, there will exist v-rows τ_1^v and τ_2^v and pairs of extremal states $x^0, x^{\max_v}$ in τ_1^v, and $y^0, y^{\max_v}$ in τ_2^v, such that $\mathrm{dest}(u, x^0) < \mathrm{dest}(u, x^{\max_v})$ and $\mathrm{dest}(u, y^0) > \mathrm{dest}(u, y^{\max_v})$. According to Corollary 3.19, such an ASTG does not satisfy the ASTG condition 4. □

3.1.4 Asynchronous state transition graphs in low dimension

Based on the ASTG conditions and the construction rules, all possible ASTGs in both Boolean and multi-valued state spaces can be enumerated. Early work on the enumeration of ASTGs in low dimension can be found in [Glass, 1977]. Glass gives a combinatorial method for classifying state transition diagrams in N-dimensional Boolean space based on symmetries. He considers only state transition diagrams with exactly one directed edge between any two states. The corresponding biological system has no self-regulation and does not need to satisfy the Snoussi-condition. In our work, self-regulation of components is possible. In the following examples, both cases with and without the Snoussi-condition are considered.

Using Proposition 3.26, all possible ASTGs in the three Boolean state spaces of 1-D, 2-D and 3-D are enumerated and the four ASTG conditions are checked. Figure 3.18 shows the 1-D, 2-D and 3-D Boolean state spaces.



(a) 1-D (b) 2-D (c) 3-D

Figure 3.18: (a) A 1-D Boolean space. (b) A 2-D Boolean space. (c) A 3-D Boolean space.

Remark 3.32 Enumerating all ASTGs on a Boolean space X_B in the following two ways leads to the same set of all possible ASTGs.

1. Enumerate all possible $\delta(u, x)$, for all $x \in X_B$ and for all $u \in V$.
2. Enumerate all u-rows in X_B for each component $u \in V$, chosen from all possible Boolean u-rows.

There are in total $2^{|V|2^{|V|}}$ ASTGs on X_B and each ASTG satisfies the three ASTG conditions.

Boolean case, dimension 1: *In a 1-D Boolean state space, $X_B = \{0, 1\}$, how many ASTGs do there exist? Do they all satisfy the four ASTG conditions?*

A 1-D Boolean state space contains only one row. As shown in Figure 3.16, there are 4 possible Boolean u-rows in an ASTG which are the eligible row structures of threshold value 1. Therefore, there are exactly 4 possible ASTGs on $X_B = \{0, 1\}$. Each state in the Boolean state

space defines a hypercube. According to Proposition 3.26, the enumerated ASTGs satisfy all four ASTG conditions.

Boolean case, dimension 2: *In a 2-D Boolean state space, $X_B = \{0,1\}^2$, how many ASTGs do there exist? How many of them satisfy all four ASTG conditions?*

A 2-D Boolean state space contains 2 rows in each dimension. As there are 4 possible Boolean u-rows, one can have $4^4 = 256$ graphs based on X_B, resulting from a full combination of the 4 u-rows. One can combine them freely, and this gives all possible graphs based on X_B.

According to Proposition 3.26, ASTG condition 1, 2 are satisfied. ASTG condition 3 is also satisfied on all these 256 graphs, where each state corresponds to a u-hypercube in each dimension. Moreover, every graph has a corresponding model (can be inferred by Lorenz algorithms). All 256 graphs based on X_B are ASTGs.

However, only 196 ASTGs satisfy ASTG condition 4, having compatible models after using the *Generalised Algorithm: Observability-Snoussi-Model*. In each direction, there are two u-rows in X_B. The two u-rows are from all combination of the 4 Boolean rows except two rows of *pos* and *neg* types, thus $4^2 - 2 = 14$ different combinations. Therefore, in all there are $14^2 = 196$ ASTGs in X_B.

The other $256 - 196 = 60$ ASTGs have at least one violation of the Snoussi-condition, where *pos* and *neg* types of u-rows coexist.

Boolean case, dimension 3: *In a 3-D Boolean state space, $X_B = \{0,1\}^3$, how many ASTGs do there exist? How many of them satisfy all four ASTG conditions?*

A 3-D Boolean state space contains 4 rows in each dimension. As there are 4 possible Boolean u-rows, one can have $4^{4\times 3} = 2^{24}$ ASTGs, resulting from a full combination of the 4 u-rows. According to Proposition 3.26, one can combine the 4 row structures freely to form all possible graphs based on X_B, which is the same as the set of graphs obtained by enumerating $\delta(u,x)$ for all 2^3 states in all 3 dimensions. ASTG condition 1, 2 and 3 are satisfied. Therefore, there exist 2^{24} ASTGs.

Like in the 2-D case, not all ASTGs satisfy ASTG condition 4. In the 2^{24} graphs, if we look only at the transitions in one dimension, there are 4^4 different combinations for the 4 rows.

The structure of the state transition in one dimension is isomorphic to the other dimensions. Due to this isomorphism, instead of checking for all 2^{24} graphs, ASTG condition 4 is checked only in one of the three dimensions.

Among 4^4 different row structures in one single dimension, there are only 104 satisfying ASTG condition 4. Since the compatibility model condition is based on the u-hypercubes for each dimension u, this implies that there are 104^3 ASTGs satisfying the compatible model condition.

The other $256 - 104 = 152$ cases all admit at least one violation of the Snoussi-condition, which have both *pos* and *neg* types of u-rows. This number is also verified by the *Generalised Algorithm: Observability-Snoussi-Model*.

The last example illustrates the ASTG construction in a non-Boolean state space based on Proposition 3.25.

Multi-valued case, dimension 2: *Given the state space, $X = \{0,1\} \times \{0,1,2\}$, how many possible ASTGs do there exist? How many of them satisfy all four ASTG conditions?*

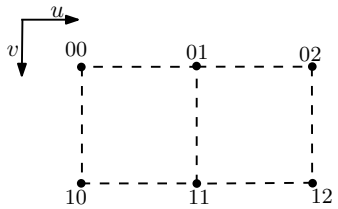


Figure 3.19: The state space $X = \{0,1\} \times \{0,1,2\}$.

Let v be the component with $x_v \in \{0,1\}$, and u the component with $x_u = \{0,1,2\}$. Similar to the Boolean case, all possible u-rows of length 3 are enumerated in Figure 3.20.

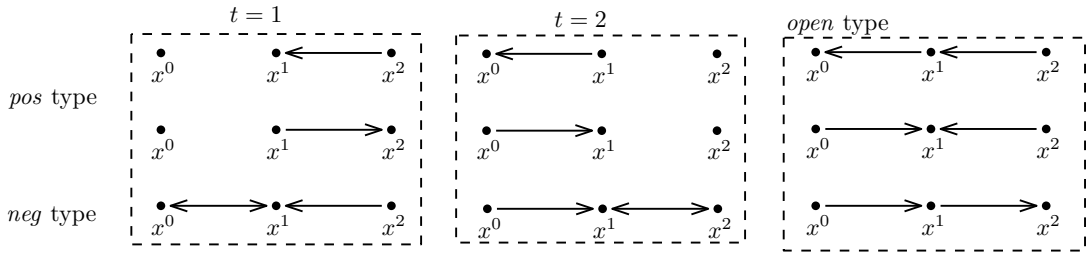


Figure 3.20: All possible u-rows of length 3, including *pos* and *neg* types of thresholds $t = 1$ and $t = 2$, and all the *open* types.

In $X = \{0,1\} \times \{0,1,2\}$, there is one u-slice along v including two u-rows. All u-rows are chosen from $\{\tau^u\}^t_{elg}$, the set of eligible u-row structures of a threshold value t. According to Proposition 3.27, all possible ASTGs on X can be enumerated. Both $\{\tau^u\}^{t=1}_{elg}$ and $\{\tau^u\}^{t=2}_{elg}$ contains 6 u-row structures including 3 u-rows of *open* type, 2 of *pos* type and 1 of *neg* type. For the two u-rows in X in the u-slice along v, there are the following cases.

1. The two u-rows are chosen from $\{\tau^u\}^{t=1}_{elg}$, there are 6^2 combinations.
2. The two u-rows are chosen from $\{\tau^u\}^{t=2}_{elg}$, there are 6^2 combinations.
3. The case when the two u-rows are both of *open* type are included in both cases above, which are 3^2 combinations.

Therefore, all together there are $6^2 + 6^2 - 3^2 = 63$ different structures u-slices along v.

Since v is Boolean, the two extremal rows in the v-slice along u (including 3 v-rows) are chosen from the 4 Boolean row structures, which gives 4^2 combinations. The v-row in the middle is chosen to be isomorphic to one to the extremal rows, which gives $4^2 \times 2$ combinations. However, if two extremal v-rows have the same structure, then the middle v-row has only one choice. Therefore, there are $4^2 \times 2 - 4 = 28$ possible structures for the three v-rows in the v-slice along u.

Therefore, there exists $63 \times 28 = 1764$ possible ASTGs on X. The result is verified by the *Generalised Algorithm Visibility-Model*.

However, the compatible model condition does not allow the coexistence of the *pos* and *neg* types of rows, or of different threshold values. For the two u-rows in the u-slice along v, the following cases will violate the ASTG condition 4.

1. One u-row is of *pos* type, and the other is of *neg* type. There exist 4 u-row structures of *pos* type, 2 of *neg* type. This gives $4 \times 2 \times 2 = 16$ different cases.
2. Two u-rows are of *pos* type, but with different threshold values. This gives $2 \times 2 \times 2 = 8$ cases.
3. Two u-rows are of *neg* type, but with different threshold values. There exist 2 cases.
4. Two u-rows (x^0, x^1, x^2) and (y^0, y^1, y^2), where $\mathrm{sgn}(\mathrm{dest}(u, x^0) - \mathrm{dest}(u, y^0)) \neq 0$ is not the same as $\mathrm{sgn}(\mathrm{dest}(u, x^2) - \mathrm{dest}(u, y^2)) \neq 0$. According to Corollary 3.19, this is a violation of ASTG condition 4. The following two pairs of u-rows are of this kind: (τ_1^u, τ_0^u) and (τ_2^u, τ_0^u), where
   - τ_0^u with $\delta(u, \tau_0^u) = (1, 0, -1)$ and the destination values $\mathrm{dest}(u, \tau_0^u) = (1, 1, 1)$,
   - τ_1^u with $\delta(u, \tau_1^u) = (0, -1, 0)$ and the destination values $\mathrm{dest}(u, \tau_1^u) = (0, 0, 2)$,
   - τ_2^u with $\delta(u, \tau_2^u) = (0, 1, 0)$ and the destination values $\mathrm{dest}(u, \tau_2^u) = (0, 2, 2)$.

   Thus, for the u-slice along v, there exist 2×2 cases containing a violation of ASTG condition 4.

In total, there are $9^2 - 16 - 8 - 2 - 4 = 51$ possible combinations for the u-slice along v in X.

Moreover, from the 2-D Boolean case 3.1.4, there exist $16 - 2 = 14$ different cases for the two extremal v-rows. For the v-slice along u with 3 v-rows, the v-row in the middle can be isomorphic to one of the extremal rows, which gives $14 \times 2 - 4 = 24$ structures. Here 4 denotes the number of cases where the two extremal rows are isomorphic.

Altogether, there are $51 \times 24 = 1224$ ASTGs satisfying all four ASTG conditions. The result is verified by the *Generalised Algorithm Observable-Snoussi-Model*.

3.2 Generalising Lorenz algorithms with ASTG conditions

Based on the three necessary and sufficient ASTG conditions together with the compatible model condition, Lorenz algorithms are *generalised* for processing general input, *i.e.*, arbitrary graphs based on a state space X. While the Lorenz algorithms as presented in Chapter 2 assume the input to be an ASTG, the *generalised* versions include ASTG detection. The Matlab package is available at https://github.com/riplotus/RE_LorenzAlgorithms.

3.2.1 Generalised Lorenz algorithms

Generalised Algorithm Logical-Parameters The *Generalised Algorithm Logical-Parameters* takes a graph based on the state space and a simple IG as input. For each resource of each component, if the corresponding extremal row does not belong to the set of eligible u-row structures, an error of invalid input will be returned.

If no error is detected, then the output is a logical parameter function such that the state transition of each extremal state can be regenerated, which is the same as from the original algorithm.

The following remark explains the cases when this algorithm will return errors.

Remark 3.33 A u-row does not belongs to a single row type if any of the following holds:

1. There exist two positions $i, j \in \{1, \ldots, \max_u\}$, $i \neq j$, such that $\delta(u, x^{i-1}) = 1$, $\delta(u, x^i) = -1$ and $\delta(u, x^{j-1}) = 1$, $\delta(u, x^j) = -1$.

2. There exist two positions $i, j \in \{1, \ldots, \max_u\}$, $i \neq j$, such that $\delta(u, x^{i-1}) \leq 0$, $\delta(u, x^i) \geq 0$ and $\delta(u, x^{j-1}) \leq 0$, $\delta(u, x^j) \geq 0$, which means that there are more than two states $\delta(u, \cdot) = 0$.

From the simple IG, the predecessor set of each component and the set of resources of each component are easy to find. Corresponding to each resource of each component, there is an extremal state. A row that contains this state is an extremal row. If an extremal row of the input graph visited during the algorithm does not belong to the set of eligible row structures according to Remark 3.33, then an error of invalid input is returned. If no error is found, the logical parameter function K is the output. The pseudocode is shown in Algorithm 3.1.

Algorithm 3.1: Generalised Algorithm Logical-Parameters

Input: A simple IG $\tilde{I} = (V, E, \varepsilon, \max)$ and a graph based on X, $\mathcal{G}_X = (X, S)$. /* S is given by $\delta_{\mathcal{G}_X} : V \times X \to \{-1, 0, +1\}$. */

Output: A logical parameter function K, such that $\tilde{I}$ together with K can define the transitions of all extremal states in T; or an error is returned if any extremal row of $\mathcal{G}_X$ is not from a valid ASTG.

foreach $u \in V$ **do**
  if $(u, u) \notin E$ **then**
    foreach $\omega \subseteq \mathrm{Pre}(u)$ **do**
      Construct an extremal state $x \in X : \mathrm{Res}_u(x) = \omega$;
      $(x^0, \ldots, x^{\max_u}) := \tau^u$;
      if $\exists!\, a \in \{0, \ldots, \max_u\} : \delta_{\mathcal{G}_X}(u, x^a) = 0$ **then**
        $K(u, \omega) := a$;
      else /* τ^u does not belong to *open*-type */
        return *Error. IG is not consistent with* $\mathcal{G}_X$.
  else
    foreach $\omega \subseteq \mathrm{Pre}(u) \backslash \{u\}$ **do**
      Construct an extremal state $x \in X : \mathrm{Res}_u(x) = \omega$;
      $(x^0, \ldots, x^{\max_u}) := \tau^u$;
      if $\exists!\, a < b \in \{0, \ldots, \max_u\} : \delta_{\mathcal{G}_X}(u, x^a) = \delta_{\mathcal{G}_X}(u, x^b) = 0$ **then** /* *pos* type */
        $K(u, \mathrm{Res}_u(x^0)) := a$;
        $K(u, \mathrm{Res}_u(x^{\max_u})) := b$;
      else if $\exists!\, a \in \{0, \ldots, \max_u\} : \delta_{\mathcal{G}_X}(u, x^a) = 0$ **then** /* *open* type */
        $K(u, \mathrm{Res}_u(x^0)) := a$;
        $K(u, \mathrm{Res}_u(x^{\max_u})) := a$;
      else if $\exists! t \in \{1, \ldots, \max_u\}$: $\delta_{\mathcal{G}_X}(u, x^{t-1}) = 1$, $\delta_{\mathcal{G}_X}(u, x^t) = -1$ **then** /* *neg* type */
        $K(u, \mathrm{Res}_u(x^0)) := \max_u$;
        $K(u, \mathrm{Res}_u(x^{\max_u})) := 0$;
      else /* does not belong to a single row type */
        return *Error. Invalid input* $\mathcal{G}_X$.
return K

Generalised Algorithm Activity-Value The *Generalised Algorithm Activity-Value* keeps the main procedure of the original algorithm and adds error detection to deal with more general inputs. It takes (u, v), ω, $\delta_{\mathcal{G}_X}(v, \cdot)$ as input. However, the following two special cases and an error case are considered.

1. Two special cases:
   (a) For a self-loop (u, u) and a resource ω of u, if the extremal u-row corresponding to ω belongs to the *open* type, then let $\vartheta(u, u) := 0$.
   (b) For an interaction (v, u) and a resource ω of u, find the u-slice along v via the extremal state x with $\mathrm{Res}_u(x) = \omega$.

   If all the u-row structures in the u-slice along v are isomorphic to each other, then let $\vartheta(u, v) := 0$.
2. If more than one change occurs in the u-row structures in the u-slice along v, an error will be returned.

The pseudocode is given in Algorithm 3.2.

Algorithm 3.2: Generalised Algorithm Activity-Value

Input: $u, v \in V, \omega \subseteq V \backslash \{u\}, \delta_{\mathcal{G}_X}(v, \cdot)$
Output: $\vartheta(u, v)$

Construct an extremal state $x \in X : \mathrm{Res}_u(x) = \omega$;
$\tau^v := (x^0, \ldots, x^{\max_u})$; /* $x \in \tau^v$ */
if $u = v$ **then**
  if τ^v *is of type* pos **then**
    $\exists! \, i < \max_u : \delta_{\mathcal{G}_X}(v, x^i) \leq 0 \wedge \delta_{\mathcal{G}_X}(v, x^{i+1}) \geq 0$;
    $\vartheta(u, v) := i + 1$;
  else if *is of type* neg **then**
    $\exists! \, i < \max_u : \delta_{\mathcal{G}_X}(v, x^i) = 1 \wedge \delta_{\mathcal{G}_X}(v, x^{i+1}) = -1$;
    $\vartheta(u, v) := i + 1$;
  else /* is of type *open* */
    $\vartheta(u, v) := 0$ /* From ω, (u, v) does not exist. */
else /* $u \neq v$ */
  $x_u := 0$;
  $\tau^v := (x^0, \ldots, x^{\max_v})$; /* $x_u = 0$, $x \in \tau^v$ */
  if $\exists! l \in \{1, \ldots, \max_u\} : \tau^v_{x+(l-1)\mathbf{e}_u} \ncong \tau^v_{x+l\mathbf{e}_u}$ **then**
    $\vartheta(u, v) := l$;
  else if $\forall l \in \{1, \ldots, \max_u\} : \tau^v \cong \tau^v_{x+l\mathbf{e}_u}$ **then**
    $\vartheta(u, v) := 0$;
  else /* > 1 row structure changes */
    return *Error. Invalid input, graph allows influence from multiple edges.*
return $\vartheta(u, v)$

Generalised Algorithm Visibility-Model This algorithm also keeps the main procedure of the original algorithm, but includes error detection. It takes a graph based on X as input and gives as output a *visible* model if the input is a valid ASTG. In the following cases, an error will be returned.

1. Errors returned by calling *Generalised Algorithm Logical-Parameters* and *Generalised Algorithm Activity-Value*.
2. If different threshold values are obtained for the same interaction.
3. If the ASTG of the output model is not isomorphic to the input graph.

The pseudocode is shown in Algorithm 3.3.

Algorithm 3.3: Generalised Algorithm Visibility-Model

Input: A graph $\mathcal{G}_X = (X, S)$ based on X with the component set V and the maximal activity levels $\max(\cdot)$; /* S is given by $\delta_{\mathcal{G}_X} : V \times X \to \{-1, 0, +1\}$. */

Output: A model $M = (I, K)$ with minimal number of edges, and ASTG$(M) \cong \mathcal{G}_X$; errors otherwise.

$E := \varnothing = \emptyset$;
$\hat{E} := V \times V$;
$\hat{\varepsilon} :\equiv +$;
$\hat{I} := (V, \hat{E}, \hat{\varepsilon}, \max)$;
$\hat{K}(\cdot,\cdot) :=$ Generalised-Logical-Parameters$(\hat{I}, T)$; /* if no errors are found. */
foreach $(u, v) \in \hat{E}$ **do**
    edge_candidate $:= \emptyset$;
    foreach $\omega \subseteq V \backslash \{u\} : \hat{K}(v, \omega) \neq \hat{K}(v, \omega \cup \{u\})$ **do**
        $\vartheta(u, v) :=$ Generalised-Activity-Value$(u, v, \omega, \delta_{\mathcal{G}_X}(v, \cdot))$;
        if $\hat{K}(v, \omega) < \hat{K}(v, \omega \cup \{u\})$ **then**
            edge_candidate $:=$ edge_candidate $\cup \{(+, \vartheta(u, v))\}$;
        else /* $\hat{K}(v, \omega) > \hat{K}(v, \omega \cup \{u\})$ */
            edge_candidate $:=$ edge_candidate $\cup \{(-, \vartheta(u, v))\}$;
    if $\exists\,!\ \vartheta(u, v)$ *in* edge_candidate **then**
        $E := E \cup \{(u, v)\}$;
        $(\varepsilon(u, v), \vartheta(u, v)) :=$ one element of edge_candidate ; /* pick an element */
    else if edge_candidate *has more than one threshold value* **then**
        return *Error. Illegal input.*
$I := (V, E, \varepsilon, \vartheta, \max)$;
foreach $v \in V$ **do**
    foreach $\omega \subseteq \mathrm{Pre}_I(v)$ **do**
        $\varsigma := \{u \in \omega \mid \varepsilon(u, v) = +\} \cup \{u \in \mathrm{Pre}_I(v) \backslash \omega \mid \varepsilon(u, v) = -\}$;
        $K(v, \omega) := \hat{K}(v, \varsigma)$;
if *ASTG*$(M) \cong \mathcal{G}_X$ **then** /* Validation */
    return $M = (I, K)$
else
    return *Error.* $\mathcal{G}_X$ *does not admit a model.*

Generalised Algorithm Observability-Snoussi-Model The *Generalised Algorithm Observability-Snoussi-Model* again keeps the main procedure of the original algorithm and includes error detection. It takes a graph based on X as input and gives as output a *compatible* model, if the input is a valid ASTG and satisfies the compatible model condition. In the following cases, the algorithm returns an error.

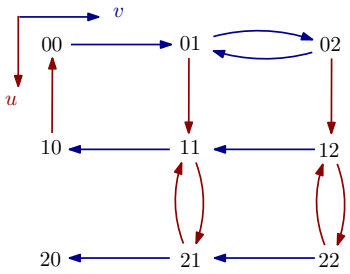
1. Errors from using the *Generalised Algorithm Logical-Parameters* and *Generalised Algorithm Activity-Value*.
2. If for one interaction, both signs and/or different threshold values are found, then an error *multiple edges* will be returned.
3. If the ASTG of the resulting model is not isomorphic to the input graph.

The pseudocode is shown in Algorithm 3.4.

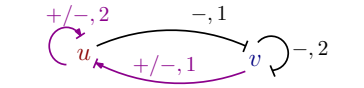
For an ASTG satisfying the four ASTG conditions, the logical parameter function of the model obtained by the *Generalised Algorithm Observability-Snoussi-Model* may be different from the one obtained by *Generalised Algorithm Visibility-Model*.

3.2.2 Examples for generalised Lorenz algorithms

Example 3.34 Figure 3.21 shows a graph based on X and the output model from *Generalised Algorithm Visibility-Model*.



(a) $\mathcal{G}_X$



in the case $\varepsilon(u,u) = +$, $\varepsilon(v,u) = -$:

ω	$K(u,\omega)$	ω	$K(v,\omega)$
$\emptyset$	1	$\emptyset$	0
$\{v\}$	0	$\{v\}$	0
$\{u\}$	0	$\{u\}$	0
$\{u,v\}$	2	$\{u,v\}$	2

(b) A possible $M = (I, K)$

Figure 3.21: (a) A graph $\mathcal{G}_X$ based on X. (b) One possible output model using *generalised Visibility-Model*.

The extremal u-row $(00, 10, 20)$ is of *pos* type, while the other extremal u-row $(01, 12, 22)$ is of *neg* type.

Using *Generalised Algorithm Visibility-Model*, the signs of the two interactions (u, u) and (v, u) are not unique. After initialisation, the pseudo logical parameters $\hat{K}(u, \cdot)$ are $\hat{K}(u, \emptyset) = 0$, $\hat{K}(u, \{v\}) = \hat{K}(u, \{u\}) = 2$ and $\hat{K}(u, \{u, v\}) = 1$. For the two incoming interactions of u, this gives the following.

1. Sign $\varepsilon(u, u)$:
   (a) $\hat{K}(u, \emptyset) = 0 < \hat{K}(u, \{u\}) = 2$ infers $\varepsilon(u, u) := +$.
   (b) $\hat{K}(u, \{v\}) = 2 > \hat{K}(u, \{u, v\}) = 1$ infers $\varepsilon(u, u) := -$.
2. Sign $\varepsilon(v, u)$:
   (a) $\hat{K}(u, \emptyset) = 0 < \hat{K}(u, \{v\}) = 2$ infers $\varepsilon(v, u) := +$.
   (b) $\hat{K}(u, \{u\}) = 2 > \hat{K}(u, \{u, v\}) = 1$ infers $\varepsilon(v, u) := -$.

However, when using the *Generalised Algorithm Observability-Snoussi-Model*, an error of admitting multiple edges will be returned. This algorithm first checks the self-loop of a component and then other incoming interactions. If the sign of an interaction is not unique, then

Algorithm 3.4: Generalised Algorithm Observability-Snoussi-Model

Input: A graph $\mathcal{G}_X = (X, S)$ based on X with the component set V and the maximal activity levels $\max(\cdot)$ /* S is given by $\delta_{\mathcal{G}_X} : V \times X \to \{-1, 0, +1\}$. */

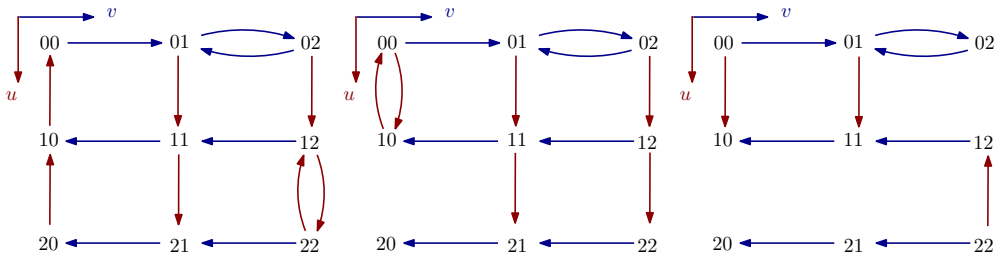
Output: A model $M = (I, K)$ with ASTG$(M) \cong \mathcal{G}_X$, which has minimal number of edges, satisfying the observability condition and Snoussi-condition in every interaction; otherwise, return error.

$E := \varnothing$;
$\hat{E} := V \times V$;
$\hat{I} := (V, \hat{E}, \hat{\varepsilon} :\equiv +, \max)$;
$\hat{K}(\cdot,\cdot) :=$ Generalised-Logical-Parameters$(\hat{I}, T)$; /* if no errors are found. */
foreach $v \in V$ **do**
  foreach $\omega \subseteq V\backslash\{v\} : \hat{K}(v,\omega) \neq \hat{K}(v,\omega \cup \{v\})$ **do**
    $\vartheta(v,v) :=$ Generalised-Activity-Value$(v, v, \omega, \delta_{\mathcal{G}_X}(v,\cdot))$;
    if $\hat{K}(v,\omega) < \hat{K}(v,\omega \cup \{v\})$ **then**
      edge_candidate := edge_candidate $\cup \{(+, \vartheta(v,v))\}$;
    else /* $\hat{K}(v,\omega) > \hat{K}(v,\omega \cup \{v\})$ */
      edge_candidate := edge_candidate $\cup \{(-, \vartheta(v,v))\}$;
  if edge_candidate *has a unique element* **then**
    $E := E \cup \{(v,v)\}$;
    $(\varepsilon(v,v), \vartheta(v,v)) :=$ edge_candidate;
    foreach $\hat{\omega} \subseteq V$ **do**
      if $\hat{K}(v,\hat{\omega})$ *does not lie on* $\hat{\omega}$*-side from* v **then** /* ε is in I, $\hat{\omega} \subseteq \mathrm{Pre}_{\hat{I}}(v)$ */
        $$\hat{K}(v,\hat{\omega}) := \begin{cases} \vartheta(v,v) - 1 & \varepsilon(v,v) = +, v \in \hat{\omega} \\ \vartheta(v,v) & \varepsilon(v,v) = +, v \notin \hat{\omega} \\ 0 & \varepsilon(v,v) = -, v \in \hat{\omega} \\ \max_v & \varepsilon(v,v) = -, v \notin \hat{\omega} \end{cases};$$
  else if edge_candidate *has more than one element* **then**
    return *Error. Invalid input,* $\mathcal{G}_X$ *admits influences from multiple edges.*
  foreach $u \in V\backslash\{v\}$ **do**
    foreach $\omega \subseteq V\backslash\{u\} : \hat{K}(v,\omega) \neq \hat{K}(v,\omega \cup \{u\})$ **do**
      $\vartheta(u,v) :=$ Generalised-Activity-Value$(u, v, \omega, \delta(v,\cdot))$;
      if $\hat{K}(v,\omega) < \hat{K}(v,\omega \cup \{u\})$ **then**
        edge_candidate := edge_candidate $\cup \{(+, \vartheta(u,v))\}$;
      else /* $\hat{K}(v,\omega) > \hat{K}(v,\omega \cup \{u\})$ */
        edge_candidate := edge_candidate $\cup \{(-, \vartheta(u,v))\}$;
    if edge_candidate *has only one element* **then**
      $E := E \cup \{(u,v)\}$;
      $(\varepsilon(u,v), \vartheta(u,v)) :=$ edge_candidate;
    else if edge_candidate *has more than one element* **then**
      return *Error. Invalid input,* $\mathcal{G}_X$ *admits influences from multiple edges.*
$I := (V, E, \varepsilon, \vartheta, \max)$;
foreach $v \in V$ **do**
  foreach $\omega \subseteq \mathrm{Pre}_I(v)$ **do**
    $\varsigma := \{u \in \omega \mid \varepsilon(u,v) = +\} \cup \{u \in \mathrm{Pre}_I(v)\backslash\omega \mid \varepsilon(u,v) = -\}$;
    $K(v,\omega) := \hat{K}(v,\varsigma)$;
if *ASTG*$(M) \cong \mathcal{G}_X$ **then** /* Validation */
  return $M = (I, K)$
else
  return *Error.* $\mathcal{G}_X$ *does not admit a model.*

the compatible model condition is violated. For this reason, the algorithm will stop after finding the first error on (u, u).

Example 3.35 Figure 3.22 shows three graphs based on X which are not ASTGs. They will be used as input for *Generalised Algorithm Visibility-Model* and *Generalised Algorithm Observability-Snoussi-Model*. For each graph, the following errors will be detected by the generalised Lorenz algorithms.

1. Graph $\mathcal{G}_X^1$ in Figure 3.22a: an error will be returned during the call of *Generalised Algorithm Activity-value* when checking the value for $\vartheta(v, u)$. The reason lies in the u-slice along v, which contains three different u-rows.
2. Graph $\mathcal{G}_X^2$ in Figure 3.22b: an error will be returned during the call of *Generalised Algorithm Logical-Parameters* when checking the row type of $(00, 10, 20)$, which does not belong to any single row type.
3. Graph $\mathcal{G}_X^3$ in Figure 3.22c: an error will be returned when checking the interaction (u, u) in both *Generalised Algorithm Visibility-Model* and *Generalised Algorithm Observability-Snoussi-Model*. All u-rows in $\mathcal{G}_X^3$ are of *pos* type, however with different threshold values.



(a) $\mathcal{G}_X^1$ (b) $\mathcal{G}_X^2$ (c) $\mathcal{G}_X^3$

Figure 3.22: Three graphs based on X. (a) $\mathcal{G}_X^1$, three different u-rows in the same u-slice along v. (b) $\mathcal{G}_X^2$, one u-row does not belong to any single row type. (c) $\mathcal{G}_X^3$, u-rows of *pos* type with different threshold values.

Chapter 4

Model analysis and discrete modelling workflows

Dynamic system behaviours can be modelled as attractors or specific pathways in the corresponding ASTGs. In general, there may exist many regulatory structures generating a specific dynamic behaviour. One focus of reverse engineering is to explore the structural requirements necessary for obtaining a given dynamic behaviour. In this context, analysing regulatory structures and characterising them as a whole is of high interest. In what follows, a regulatory structure generating a desired dynamic behaviour will be called a *functional GRN*.

Robustness is a classic measure to describe the ability of an individual structure to maintain its functionality under perturbations. In the context of discrete modelling, Section 4.1 introduces the notion of *realizability* in order to measure the ability of an IG or all functional IGs to generate a given dynamic property depending on the logical parameter functions.

A set of GRNs can be analysed by detecting motifs in the IGs or other regulatory patterns. If functionality of a GRN is specified by a Boolean function on the set of all possible GRNs for a given set of nodes, functional GRNs can be characterised by logical expressions. The thesis by [Palinkas, 2015] showed a way of encoding multi-valued logical networks by Boolean expressions and minimising the logical functions to get a minimal representation. Here in Section 4.2, a *logical analysis method* is proposed to obtain a compact or minimal description of a set of models and to detect interesting patterns in it via Quine-McCluskey algorithm. IGs are transformed into Boolean expressions using a suitable encoding. The same logical method may also be applied to a set of logical parameter functions.

Two discrete modelling workflows are proposed in Section 4.3, which allow searching for all functional models and analysing them. Starting from some attractors representing a biological function of interest, the *forward modelling* workflow enumerates all possible models in the discrete modelling framework. In contrast, the *reverse engineering* workflow first constructs all possible graphs on the presumed state space that exhibit the desired property. Then it makes use of the generalised Lorenz algorithms from Chapter 3 to infer the functional models.

4.1 *Realizability*

Various concepts of robustness have been proposed in the literature to study complex networks and biological systems. [Kitano, 2004] defines robustness as the ability of a system to maintain its function under perturbations. [Perkins et al., 2010] propose a hybrid system approach to relate the logical structure of genetic networks to robust dynamics, trying to understand the

essentials of robustness. Based on Kitano's definition, [Wagner, 2013] extends to notion of robustness to the genetic, protein and organism levels of living systems.

[Breindl et al., 2010] use a robustness measure to obtain all GRNs generating certain multistabilities. The GRNs are modelled by qualitative differential equations, the robustness measure is calculated from the global minimum of all perturbations to the regulatory functions. After discretizing the continuous concentrations to Boolean or multi-valued ranges, the state space becomes a Cartesian product of intervals, so-called hyperrectangles. For small perturbations within those ranges the continuous system will stay in the same orthant, without explicit changes in the discrete states.

The regulatory structures in our discrete modelling framework are IGs, and the dynamic behaviours of the system can be modelled by one or more attractors in the corresponding ASTGs. An IG I can have many compatible logical parameter functions and hence different attractors can be generated by these models with I. If one of those attractors related with I is exactly the desired system's behaviours A, this IG I and the related models are called functional for some desired properties A.

Definition 4.1 (Functional model for a property A) Consider a model $M = (I, K)$ with an IG I and a compatible logical parameter function K. If the ASTG of M contains some desired properties A, then M is called *functional* for A. Moreover, I is a *functional* IG for A. Together with I, also K can be called *functional* for A.

A compatibility model can guarantee that all interactions are observable and agreed by the logical parameters. To characterise the ability of an IG on realising the desired properties A, *realizability* is defined in Definition 4.2.

Definition 4.2 (Realizability of an IG I functional for some desired properties A) For some desired properties A, the *realizability* of an IG I is defined as the ratio of its functional logical parameter functions among all compatible K's in the parameter space.

$$\mathcal{R}_I^A = \frac{|\{\text{functional } K_I\}|}{|\{\text{compatible } K_I\}|} \tag{4.1}$$

Here $|S|$ means the cardinality of the set S. This definition gives a way to quantify the ability of a topology to realise the desired attractors.

For simplicity, in this section, a logical parameter function K is called a *parameter*. The functional parameters of I for A is a subset of its compatible parameters. The compatible parameters of I is a subset in its parameter space. The parameter space of an IG I depends the maximal activity value and the size of the predecessor set of each node. For a Boolean IG, the size of the parameter space is $\Pi_{v \in V} 2^{2^{|\mathrm{Pre}(v)|}}$. In a compatible model, the parameters of every component are highly related with the predecessor set. For all compatible parameters of an IG, the parameters of each component need to be compatible with all incoming interactions of the component.

In a functional model for some desired properties, the parameters for each component are a subset of the compatible ones. Due to this reason, the realizability of an IG I for some desired properties can be decomposed into individual components.

Definition 4.3 (Realizability of a component of an IG I functional for some desired properties A) Consider an IG I functional for some desired properties A. The *realizability of a component* in I for A is defined as the ratio of the number of the functional logical parameters on this

component from all compatible ones, termed as $\mathcal{R}^A_{I,v}$.

$$\mathcal{R}^A_{I,v} = \frac{|\{\text{functional } K_I(v,\cdot)\}|}{|\{\text{compatible } K_I(v,\cdot)\}|} \tag{4.2}$$

Here, $K_I(v,\cdot)$ denotes the logical parameters for component v in IG I and contains $K_I(v,\mathrm{Res})$ for all resources $\mathrm{Res} \subseteq \mathrm{Pre}(v)$. The relation between $\mathcal{R}^A_I$ $\mathcal{R}^A_{I,v}$ is shown by Equation 4.3.

$$\mathcal{R}^A_I = \Pi_{v\in V}\mathcal{R}^A_{I,v}. \tag{4.3}$$

The *capacity measure* of A describes the possibility of A to be realised by all functional IGs.

Definition 4.4 (Capacity measure) Known all functional IGs $\mathcal{I}$ for a desired property A, the *capacity measure* of A is defined as the ratio of the number of functional Ks for all IGs in $\mathcal{I}$ out of the number of all compatible Ks for all IGs in $\mathcal{I}$.

$$\mathcal{C}^A = \frac{\sum_{I\in\mathcal{I}}|\{\text{functional } K_I\}|}{\sum_{I\in\mathcal{I}}|\{\text{compatible } K_I\}|} = \frac{|\{\text{functional models}\}|}{|\{\text{compatible models}\}|} \tag{4.4}$$

$\sum_{I\in\mathcal{I}}|\{\text{functional } K_I\}|$ is the same as the number of all possible functional models for A, $\sum_{I\in\mathcal{I}}|\{\text{compatible } K_I\}|$ is the same as the number of all compatible models that contain this set of IGs. $\mathcal{C}^A$ is mainly used when there exists multiple desired properties.

4.2 Logical analysis method

A *logical analysis method* is proposed to get a compact description of a set of functional models for a property A, in order to uncover interesting structural patterns. Moreover, this compact description contains all structural information for the system to have the desired property A.

The property A of the system is given by a Boolean function f such that the value of f for all functional IGs resp. functional parameters is TRUE. Each functional IG or parameter will be described by a *minterm* of f. The compact description will be the shortest or minimal DNF or CNF representation of f. Getting a shortest DNF (or CNF) representation of the Boolean function f is a logic minimisation problem. A classical way of solving the logic minimisation problem is the Quine-McCluskey algorithm, see Section 1. Theorem 3.14 in [Crama and Hammer, 2011] shows that the logic minimisation problem is NP-hard when its input is a Boolean function given by the set of its true points. The task of getting a "minimal" description of all functional IGs for A is exactly this NP-hard problem.

4.2.1 Logical analysis on IGs

Firstly, to analyse all functional IGs using the logical analysis method, each IG will be transformed into a Boolean expression. For any pair of components in the IG, there can be a positive, a negative interaction or no interaction at all. Therefore, IGs are represented by Boolean variables for the interactions.

Encoding IGs with Boolean variables Consider an IG $I = (V, E, \varepsilon, \vartheta, \max)$, where $V = \{v_1, v_2, \ldots, v_N\}$. Every interaction (v, u), for all $v, u \in V$ is labelled with a sign $\varepsilon(v, u) \in \{+, -\}$ and a threshold $\vartheta(v, u) \in \{1, \ldots, \max_v\}$.

Suppose first that $\max_i = 1$ for all $i \in V$, *i.e.*, I is a Boolean network. All thresholds are by default 1. From v to u, there can be a positive or a negative edge, or no edge at all. Two Boolean variables are used to denote the positive and negative edges:

- *positive* edge (v, u): $x^+_{uv} = 1$ denotes the existence of a *positive* edge (u, v) and $x^+_{uv} = 0$ denotes the non-existence of it.
- *negative* edge (v, u): $x^-_{uv} = 1$ denotes the existence of a *negative* edge (u, v) and $x^-_{uv} = 0$ denotes the non-existence of it.

And in an IG, from v to u, there can be the following three cases:

1. $\varepsilon(v, u) = +$, then $x^+_{uv} = 1$ and $x^-_{uv} = 0$, which is $x^+_{uv}\overline{x^-_{uv}}$.
2. $\varepsilon(v, u) = -$, then $x^+_{uv} = 0$ and $x^-_{uv} = 1$, which is $\overline{x^+_{uv}}x^-_{uv}$.
3. $(v, u) \notin E$, then $x^+_{uv} = 0$ and $x^-_{uv} = 0$, which is $\overline{x^+_{uv}}\,\overline{x^-_{uv}}$.

Therefore, for each possible edge (u, v) with $u, v \in V$, 2 Boolean variables are needed. A Boolean IG of N nodes can have N^2 possible edges. Therefore, $2N^2$ Boolean variables are needed to describe a Boolean IG of N nodes. This IG can be encoded by the product or conjunction of these $2N^2$ variables.

If an IG I is multi-valued, an interaction can have different thresholds. Similarly, two variables are used for each possible interaction. For example, let $(v, u) \in E$ and $\max_v = 2$. Then the interaction (v, u) can be $+$ or $-$ and the threshold can be 1 or 2. Thus, 4 Boolean variables are needed to describe (v, u): x^{+1}_{uv}, x^{-1}_{uv}, x^{+2}_{uv} and x^{-2}_{uv}.

In general, there are the following possibilities for an edge from v to u: an interaction (v, u) with $\vartheta(v, u) = a \in \{1, \ldots, \max_v\}$, or no edge. Then there are the following three cases:

1. $\varepsilon(v, u) = +$, then $x^{+a}_{uv} = 1$ and $x^{-a}_{uv} = 0$. (v, u) is described by: $x^{+a}_{uv}\overline{x^{-a}_{uv}} \cdot \psi$, where $\psi = \prod_{1 \le i \le \max_v}^{i \neq a} \overline{x^{+i}_{uv}}\,\overline{x^{-i}_{uv}}$.
2. $\varepsilon(v, u) = -$, then $x^{+a}_{uv} = 0$ and $x^{-a}_{uv} = 1$. (v, u) is described by: $\overline{x^{+a}_{vu}}x^{-a}_{uv} \cdot \psi$, where $\psi = \prod_{1 \le i \le \max_v}^{i \neq a} \overline{x^{+i}_{uv}}\,\overline{x^{-i}_{uv}}$.
3. no edge from v to u, then $x^{+i}_{uv} = 0$ and $x^{-i}_{uv} = 0$, for $i \in \{1, \ldots, \max_v\}$. (v, u) is described by: $\prod_{1 \le i \le \max_v} \overline{x^{+i}_{uv}}\,\overline{x^{-i}_{uv}}$.

To encode a multi-valued IG, $\sum_{i=1}^{N}(2\max_i N^2)$ Boolean variables are needed. However, no multiple edges are allowed in these IGs. Therefore, it is transformed into a constraint on the related Boolean variables.

Constraint 4.5 For an IG I without any multiple edges, for the possible interaction from v to u with $v, u \in V$, there are the following constraints:

1. if $(v, u) \in E$ with with $\vartheta(v, u) = a$, then one of the Boolean variables x^{+a}_{uv} and x^{-a}_{uv} is true (1), *i.e.*, $x^{+a}_{uv} \oplus x^{-a}_{uv} = 1$. Here $\oplus$ means exclusive disjunction.

2. if $(v,u) \in E$ with with $\vartheta(v,u) = a$, then for all $i \neq a \in \{1,\ldots,\max_v\}$, x_{uv}^{+i} and x_{uv}^{-i} are both false (or 0).

3. if $(v,u) \notin E$, then for all $i \neq a \in \{1,\ldots,\max_v\}$, x_{uv}^{+i} and x_{uv}^{-i} are false (or 0).

Using Constraint 4.5, the encoding of an interaction can be simplified, which will be called *partial encoding*.

If v is Boolean, then for the three cases $\varepsilon(v,u) = +$, $\varepsilon(v,u) = -$ and $(v,u) \notin E$, the full and partial encodings are the following:

cases	x_{uv}^{+a}	x_{uv}^{-}	constraint	full encoding	partial encoding
$\varepsilon(v,u) = +$	1	0	$x_{uv}^{+} \oplus x_{uv}^{-} = 1$	$x_{uv}^{+}\overline{x_{uv}^{-}}$	x_{uv}^{+}
$\varepsilon(v,u) = -$	0	1	$x_{uv}^{+} \oplus x_{uv}^{-} = 1$	$\overline{x_{uv}^{+}}x_{uv}^{-}$	x_{uv}^{-}
$(v,u) \notin E$	0	0		$\overline{x_{uv}^{+}}\,\overline{x_{uv}^{-}}$	$\overline{x_{uv}^{+}}\,\overline{x_{uv}^{-}}$

If v is not Boolean, then for the cases $\varepsilon(v,u) = +$ and $-$, let $\vartheta(v,u) := a$, and the case $(v,u) \notin E$, the full and partial encodings are the following:

cases	x_{uv}^{+a}	x_{uv}^{-a}	constraint	full encoding	partial encoding
$\varepsilon(v,u) = +$	1	0	$x_{uv}^{+a} \oplus x_{uv}^{-a} = 1$	$x_{uv}^{+a}\overline{x_{uv}^{-a}} \cdot \prod_{1\le i\le \max_v}^{i\neq a} \overline{x_{uv}^{+i}}\,\overline{x_{uv}^{-i}}$	x_{uv}^{+a}
$\varepsilon(v,u) = -$	0	1	$x_{uv}^{+a} \oplus x_{uv}^{-a} = 1$	$\overline{x_{uv}^{+a}}x_{uv}^{-a} \cdot \prod_{1\le i\le \max_v}^{i\neq a} \overline{x_{uv}^{+i}}\,\overline{x_{uv}^{-i}}$	x_{uv}^{-a}
$(v,u) \notin E$	0	0		$\prod_{i=1}^{\max_v} \overline{x_{uv}^{+i}}\,\overline{x_{uv}^{-i}}$	$\prod_{i=1}^{\max_v} \overline{x_{uv}^{+i}}\,\overline{x_{uv}^{-i}}$

Under the Constraint 4.5, the full encoding of an interaction (v,u) in I can be simplified into the partial encoding.

Example 4.6 As an example, Figure 4.1 shows a Boolean network of two components, being transformed into Boolean variables using full and partial encodings. Let $V = \{v_1, v_2\}$, then $2 \times 2^2 = 8$ variables are needed. To simplify notation, nodes are denoted by numbers in the Boolean variables. For example, (v_2, v_1) is denoted by x_{12}^{+} and x_{12}^{-}.

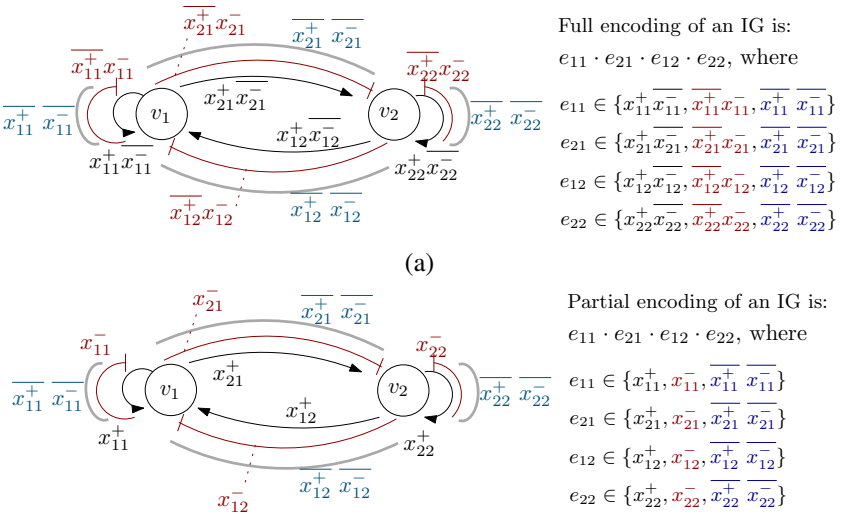


Figure 4.1: Encoding of IGs of two components. (a) Full encoding. (b) Partial encoding. Arcs without directions denote "no interaction", arrows denote "activation" and blunt arcs denote "inhibition". e_{ij} denotes the Boolean expression for an edge (j, i).

Minimal representation: logic patterns Obtaining a minimal representation of the Boolean expressions of all functional IGs for a given property A is a logic minimisation problem. The Quine-McCluskey algorithm can be applied to obtain a minimal disjunctive normal form (DNF). The software PyBoolNet [Klarner et al., 2016] is used to calculate the minimal DNF.

The minimal DNF covers every true point, which are exactly the Boolean expressions of all functional IGs for A. The clauses in the minimal DNF are translated back to interactions between components, which are called *logical IG patterns*. These logical IG patterns provide a full description of all the functional IGs for A.

Example 4.7 will illustrate the logical analysis method on a set of Boolean IGs.

Example 4.7 Assume that for a desired property A, there are the 4 functional IGs in Figure 4.2a. These 4 functional IGs are transformed into Boolean expressions. 8 Boolean variables are needed. We have $V := \{v_1, v_2\}$ and x_{11}^+ resp. x_{11}^- represents $\varepsilon(v_1, v_1) = +$ and $-$, respectively. Similarly, x_{21}^+ and x_{21}^- represents $\varepsilon(v_1, v_2) = +$ and $-$, etc. see Figure 4.2a.

Given as input the sum of the four Boolean expressions, PyBoolNet is applied to obtain the minimal DNF where the core function is the Quine-McCluskey algorithm. The details of using PyBoolNet to get the minimal DNF is omitted here. Figure 4.2b shows the results.

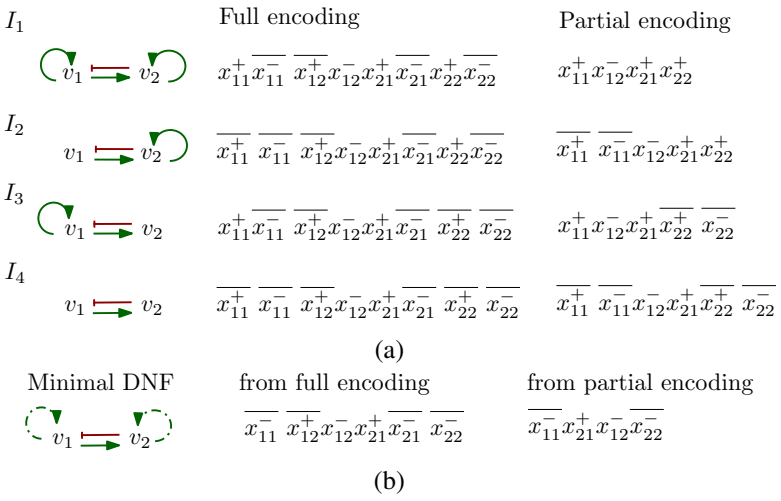


Figure 4.2: (a) 4 functional IGs for some property A, the Boolean expressions in full and partial encodings. (b) The minimal DNF from both full and partial encoding. There is only one clause in the minimal DNF, which is translated back into a graphical logical IG pattern. Arrows denote "activation", blunt arcs "inhibition". The dashed arrows denote "no inhibition", which is either an activation or no influence. △

4.2.2 Logical analysis on parameters

Given a set of functional parameters for a desired property A, the logical analysis method can be applied to obtain a compact description and to discover potential *logical parameter patterns*. It should be noted that logic minimisation in the logical analysis method is based on a a two-valued logic, while the logical parameters can be Boolean or multi-valued depending on the discrete variables that are used.

Give one IG functional for property A, it is interesting to find out which rules / parameters can enable it to be functional.

The compatible parameters of an IG reflect the correctness of the signs and monotonicity of the regulations. Given a compatible model of an IG and a parameter function K, changing the sign of an interaction does not change the compatibility of the parameter with the new IG. From only a compatible parameters of the IG, the signs of the interactions are not certain.

There are two sets of parameters of high interest.

1. The set of all functional parameters for *one* functional IG for A.
2. The set of all functional parameters for *all* functional IGs for A.

For simplicity, we discuss only Boolean parameters. A component $u \in V$ has at most $2^{|\mathrm{Pre}_u|}$ resources. For each resource $\omega \subseteq \mathrm{Pre}(u)$, $K(u,\omega) \in \{0,1\}$ because u is Boolean. Intuitively, one Boolean variable $x_\omega \in \{0,1\}$ is used to represent $K(u,\omega)$. Therefore, to encode the parameters of u, $2^{|\mathrm{Pre}_u|}$ Boolean variables are needed. The conjunction of all $2^{|\mathrm{Pre}_u|}$ Boolean variables can represent the parameters of u for all resources. Each $K(u,\omega)$ is represented by one Boolean variable. To encode a complete K (for Boolean IGs), at most $N2^N$ variables are needed.

Due to the high complexity of the logic minimisation problem, a large number of Boolean variables may render impossible the practical solution. However, the logical parameters for

a component are independent from those of another component. Given a set of parameters $\{K^1, \ldots, K^m\}$, one can encode the parameters for each component independently, and the number of Boolean variables can be considerably reduced.

Once the target set of parameters has been encoded into Boolean expressions, PyBoolNet can be applied on the conjunction of these expressions to obtain the minimal DNF. The resulting minimal DNF can be understood as the basic regulatory rule of u.

Example 4.8 will illustrate the logical analysis on functional parameters of an IG.

Example 4.8 Consider a functional IG of 4 components $\{a, b, c, d\}$ for a desired property A which has 14 functional parameters. Let K_u denote the logical parameters $K(u, \cdot)$. Among those 14 parameters, K_a and K_b are identical, while K_c and K_d are different from each other. From the IG, $\mathrm{Pre}(c) = \{a, b, d\}$ and $\mathrm{Pre}(d) = \{c, d\}$. Thus c has 8 resources and d has 4. K_c and K_d are analysed together. Let $K_{c,d}$ represent K_c and K_d, as shown in Table 4.1, for all 14 cases.

The process of translating all $K_{c,d}$'s into Boolean expressions is not included in the table. Let the Boolean variables $x_1, \ldots, x_8, x_9, \ldots, x_{12}$ represent the 8 resources of c and 4 of d. For example, $K^1_{c,d}$ can be translated into the Boolean expression: $\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}\,\overline{x_5}\,\overline{x_6}\,\overline{x_7}x_8\overline{x_9}x_{10}x_{11}x_{12}$. The minimal DNF obtained from the sum of 14 products by PyBoolNet contains 12 clauses, so-called $K^p_{c,d}$ patterns. They are listed in the lower part of Table 4.1.

The minimal DNF has 12 clauses, which is only slightly less than the 14 at the beginning. 10 patterns K^p have only one position for a choice between $\{0, 1\}$. This is not the most efficient way of encoding the parameters. There are better ways of combining the $K_{c,d}$'s to get a shorter representation, as illustrated next. Developing a general method for applying logical analysis to logical parameter functions is a topic for further research.

variable	$K_c(\emptyset)$	$K_c(\{d\})$	$K_c(\{b\})$	$K_c(\{b,d\})$
x_1	0	0	0	0
x_2	0	0	0	1
x_3	0	1	1	1
x_4	0	1	0	1
x_5	0	0	1	1

(a) Encoding K_{c-1}

variable	$K_c(\{a\})$	$K_c(\{a,d\})$	$K_c(\{a,b\})$	$K_c(\{a,b,d\})$
y_1	0	0	0	1
y_2	1	1	1	1
y_3	0	1	1	1
y_4	0	0	1	1
y_5	0	1	0	1

(b) Encoding K_{c-2}

variable	$K_d(\emptyset)$	$K_d(\{d\})$	$K_d(\{c\})$	$K_d(\{c,d\})$
z_1	0	0	0	1
z_2	0	1	1	1

(c) Encoding K_d

Figure 4.3: (a) 5 variables encode the 1st part of 4 resources of K_c. (b) 5 variables encode the 2nd part of 4 resources of K_c. (c) 2 variables encode the 4 resources of K_d.

In Table 4.1, there are many repetitions of $K(c, \omega)$ for the first 4 resources $\emptyset, \{c\}, \{b\}, \{b, c\}$.

$K_{c,d}$(Res)	$K_c - \emptyset$	$\{d\}$	$\{b\}$	$\{b,d\}$	$\{a\}$	$\{a,d\}$	$\{a,b\}$	$\{a,b,d\}$	$K_d - \emptyset$	$\{d\}$	$\{c\}$	$\{c,d\}$	$function$
$K^1_{c,d}$	0	0	0	0	0	0	0	1	0	1	1	1	1
$K^2_{c,d}$	0	0	0	0	0	1	1	1	0	1	1	1	1
$K^3_{c,d}$	0	0	0	1	0	0	1	1	0	0	0	1	1
$K^4_{c,d}$	0	0	0	1	0	0	1	1	0	1	1	1	1
$K^5_{c,d}$	0	0	0	1	0	1	0	1	0	0	0	1	1
$K^6_{c,d}$	0	0	0	1	0	1	0	1	0	1	1	1	1
$K^7_{c,d}$	0	0	0	1	0	1	1	1	0	0	0	1	1
$K^8_{c,d}$	0	0	0	1	0	1	1	1	0	1	1	1	1
$K^9_{c,d}$	0	0	0	1	1	1	1	1	0	0	0	1	1
$K^{10}_{c,d}$	0	1	1	1	1	1	1	1	0	0	0	1	1
$K^{11}_{c,d}$	0	0	1	1	0	1	1	1	0	0	0	1	1
$K^{12}_{c,d}$	0	0	1	1	0	1	1	1	0	1	1	1	1
$K^{13}_{c,d}$	0	1	0	1	0	1	1	1	0	0	0	1	1
$K^{14}_{c,d}$	0	1	0	1	0	1	1	1	0	1	1	1	1
$K^{p_1}_{c,d}$	0	0	0	0	0	0	0	1	0	1	1	1	(K^1)
$K^{p_2}_{c,d}$	0	1	1	1	1	1	1	1	0	0	0	1	(K^{10})
$K^{p_3}_{c,d}$	0	0	0	1	0	–	1	1	0	0	0	1	$(K^3 + K^7)$
$K^{p_4}_{c,d}$	0	0	0	1	0	–	1	1	0	1	1	1	$(K^4 + K^8)$
$K^{p_5}_{c,d}$	0	0	0	1	0	1	–	1	0	0	0	1	$(K^5 + K^7)$
$K^{p_6}_{c,d}$	0	0	0	1	0	1	–	1	0	1	1	1	$(K^6 + K^7)$
K^{p_7}	0	–	0	1	0	1	1	1	0	0	0	1	$(K^7 + K^{13})$
$K^{p_8}_{c,d}$	0	0	–	1	0	1	1	1	0	0	0	1	$(K^7 + K^{11})$
$K^{p_9}_{c,d}$	0	0	0	1	–	1	1	1	0	0	0	1	$(K^7 + K^9)$
$K^{p_{10}}_{c,d}$	0	–	0	1	0	1	1	1	0	1	1	1	$(K^8 + K^{14})$
$K^{p_{11}}_{c,d}$	0	0	–	1	0	1	1	1	0	1	1	1	$(K^8 + K^{12})$
$K^{p_{12}}_{c,d}$	0	0	0	–	0	1	1	1	0	1	1	1	$(K^2 + K^8)$

Table 4.1: 14 $K_{c,d}$s and 12 $K^p_{c,d}$ patterns. '–' means it can be either 1 or 0.

$K_{c,d}$(Res)	K_c-1	K_c-2	K_d	f
$K^1_{c,d}$	x_1	y_1	z_2	1
$K^2_{c,d}$	x_1	y_3	z_2	1
$K^3_{c,d}$	x_2	y_4	z_1	1
$K^4_{c,d}$	x_2	y_4	z_2	1
$K^5_{c,d}$	x_2	y_5	z_1	1
$K^6_{c,d}$	x_2	y_5	z_2	1
$K^7_{c,d}$	x_2	y_3	z_1	1
$K^8_{c,d}$	x_2	y_3	z_2	1
$K^9_{c,d}$	x_1	y_1	z_1	1
$K^{10}_{c,d}$	x_3	y_1	z_1	1
$K^{11}_{c,d}$	x_5	y_3	z_1	1
$K^{12}_{c,d}$	x_5	y_3	z_2	1
$K^{13}_{c,d}$	x_4	y_3	z_1	1
$K^{14}_{c,d}$	x_4	y_3	z_2	1

(a) 14 $K_{c,d}$'s.

$K_{c,d}$(Res)	K_c-1	K_c-2	K_d	
K^{1+2}	x_1	$(y_1\|y_3)$	z_2	$\surd$
K^{3+4}	x_2	y_4	$(z_1\|z_2)$	$\rightarrow$
K^{5+6}	x_2	y_5	$(z_1\|z_2)$	$\rightarrow$
K^{7+8}	x_2	y_3	$(z_1\|z_2)$	$\rightarrow$
K^{9+10}	$(x_1\|x_3)$	y_2	z_1	$\surd$
K^{11+12}	x_5	y_3	$(z_1\|z_2)$	$\rightarrow$
K^{13+14}	x_4	y_3	$(z_1\|z_2)$	$\rightarrow$

(b) First order combine.

$K_{c,d}$(Res)	K_c-1	K_c-2	K_d	
$K^{3+4+5+6}$	x_2	$(y_4\|y_5)$	$(z_1\|z_2)$	$\rightarrow$
$K^{5+6+7+8}$	x_2	$(y_3\|y_5)$	$(z_1\|z_2)$	$\rightarrow$
$K^{3+4+7+8}$	x_2	$(y_4\|y_3)$	$(z_1\|z_2)$	$\rightarrow$
$K^{11\sim14}$	$(x_5\|x_4)$	y_3	$(z_1\|z_2)$	$\surd$

(c) Second order combine.

$K_{c,d}$(Res)	K_c-1	K_c-2	K_d	
$K^{3\sim8}$	x_2	$(y_3\|y_4\|y_5)$	$(z_1\|z_2)$	$\surd$

(d) Third order combine.

Figure 4.4: (a) 14 $K_{c,d}$'s encoded by 12 Boolean variables. (b) The 1st order combine from (a). (c) The 2nd order combine from (b). (d) The 3rd order combine from (c).

Similar repetitions can be found in the other resources of K_c and K_d. Another way of encoding these parameters is introduced where a Boolean variable is used to encode a group of logical parameters, as shown in Figure 4.3. Each $K_{c,d}$ can be represented by much less Boolean variables than the number of resources, see in Figure 4.4a. For example, the Boolean representation of $K^1_{c,d}$ is $x_1y_1z_2$. A compact representation is obtained using Quine-McKluskey algorithm. The process is shown in Figure 4.4.

A compact representation of f is below, with $+$ denoting "disjunction" / "or".

$$f \;=\; x_1(y_1+y_3)z_2 \;+\; (x_1+x_3)y_2z_1 \;+\; x_2(y_3+y_4+y_5)(z_1+z_2) \;+\; (x_5+x_4)y_3(z_1+z_2).$$

$\triangle$

4.3 Discrete modelling workflows

Consider a set V of N genes which are related to some desired property of the system. Given the maximal activity levels $\max = (\max_i)|_{i\in V}$, the state space $X = \Pi_{i\in V}\{0,\ldots,\max_i\}$ can be determined. Assume that the desired property of the system can be modelled by a set of attractors A in the ASTG. Then the set of all ASTGs on X that contain A represents all the possible dynamics of the system that exhibit the desired property. To explore the relations between regulatory structures and the desired property, two discrete modelling workflows are proposed: "*Forward modelling*" and "*Reverse engineering*", see the flowchart in Figure 4.5.

Both workflows start from V, $\max$ and A to determine the functional models for A. Then

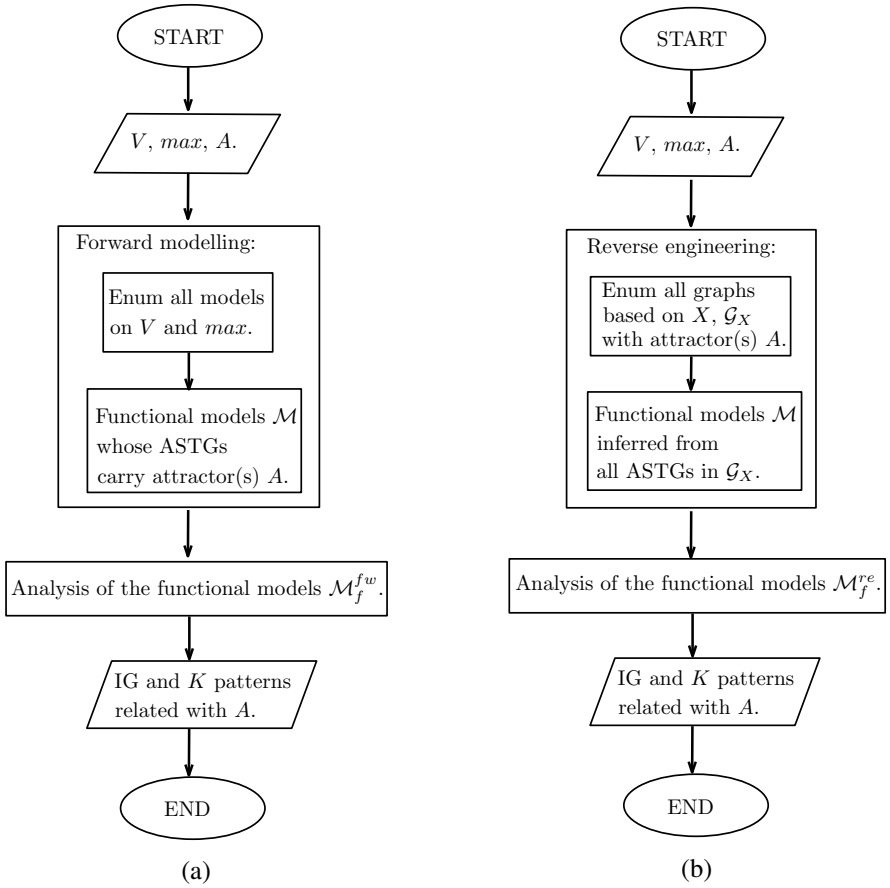


Figure 4.5: Discrete modelling workflows. (a) Forward modelling workflow. (b) Reverse engineering workflow.

these models are analysed by computing the realizability measure and by applying the logical analysis method to uncover interesting logical patterns related to the desired property.

The difference of these two workflows lies in how they find all functional models for A.

1. The "*forward modelling*" workflow first enumerates all compatible models based on V and $\max$ and then checks whether their ASTGs carry A. Getting the ASTGs from the models follows the standard discrete modelling approach, from the model to its ASTG. Thus, this workflow is called "*forward modelling*". For example, both [Cotterell and Sharpe, 2010] and [Breindl et al., 2011] used "forward" modelling within a continuous framework.

2. The "*reverse engineering*" workflow first enumerates all ASTG's based on the state space X and then uses the generalised Lorenz Algorithms to obtain all functional models. Constructing the model from the ASTG corresponds to a reverse engineering. Therefore, this workflow is named "*reverse engineering*".

4.3.1 Forward modelling workflow

Given the set of genes V and their maximal activity levels $\max = (\max_i)|_{i \in V}$, $|V| = N$, the process of "*enumerate all models on V and* $\max$" is the following.

1. Enumerate the set $\mathcal{I}$ of all IGs. A component u can activate or inhibit another component v at a threshold value $\{1, \ldots, \max_u\}$, or has no influence on v. This leads to $(2\max_u + 1)$ possibilities. Therefore, there are in total $\Pi_{u \in V}(2\max_u + 1)^N$ IGs in $\mathcal{I}$.

2. Enumerate all compatible parameters K^I_{cmpt} for all IG $I \in \mathcal{I}$. For each $I \in \mathcal{I}$,

   (a) each gene $u \in V$ has $2^{|\mathrm{Pre}(u)|}$ resources. For all $\omega \subseteq \mathrm{Pre}(u)$, $K(u, \omega) \in \{0, \ldots, \max_u\}$. Thus, there are $(\max_u + 1)^{2^{|\mathrm{Pre}(u)|}}$ possible parameters for $K(u, \cdot)$.

   (b) all $K(u, \cdot)$ are checked whether they are compatible with the incoming interactions of u. Store compatible $K(u, \cdot)$, for all $u \in V$, denoted by K^I_{cmpt}.

All possible models are enumerated, denote as $\mathcal{M} = (\mathcal{I}, \mathcal{K}|^{K^I_{cmpt}}_{I \in \mathcal{I}})$. A model is functional for A if its ASTG carries the desired attractor A. After checking each model in $\mathcal{M}$, all functional models for A are obtained, denoted by $\mathcal{M}^{fw}_f$.

The most time consuming part is enumerating all compatible parameters for all possible IGs on V. For a component $u \in V$ and predecessors $\mathrm{Pre}(u)$, compatibility checking involves the following two conditions: for all $\omega \subseteq \varsigma \subseteq \mathrm{Pre}(u)$, whether $K(u, \omega) \leq K(u, \varsigma)$, and for each $v \in \mathrm{Pre}(u)$, there exists $\omega \subseteq \mathrm{Pre}(u) \backslash \{v\}$ with which $K(u, \omega) < K(u, \omega \cup \{v\})$.

Generating all possible models on V and $\max$ is very expensive. For example, a Boolean component with 5 predecessors has 2^5 resources. The parameter space of this component is 2^{2^5}, which is more than 4 billion. If each parameter requires 0.01s, the compatible check on this component will take more than one year. For this reason, the forward modelling workflow will be applied only on systems where the maximal in-degree of each component is 4.

4.3.2 Reverse engineering workflow

Given the desired attractors A, the reverse engineering workflow starts from enumerating the set of all graphs $\mathcal{G}_X$ based on the state space X that contain the attractors A. This involves the following: For each $x \in X$,

1. if $x \notin A$, then for each $u \in V$, $\delta(u, x)$ can be any of $\{-1, 0, +1\}$.

2. if $x \in A$, then depending on A,

   (a) if x is a stable state, for all $u \in V$, $\delta(u, x) = 0$;

   (b) if x is in a cyclic attractor, then x has only the outgoing transitions which are contained in A, and no other outgoing transitions.

Therefore, there are at least $3^{|X|-|A|}$ graphs based on X. All possible ASTGs carrying the attractors A are included in these enumerated graphs. For each $\mathcal{G}_X$, the *generalised Lorenz Algorithms* are applied. If $\mathcal{G}_X$ is an ASTG and satisfies the compatible model condition, then a compatible model can be inferred. This model is called functional for A, because its ASTG has the desired attractors A. After the model inference for all enumerated graphs, the functional models for A are obtained, denoted by $\mathcal{M}^{re}_f$.

The running time depends on the enumeration of all graphs based on X. The more information in A, the less graphs will be obtained from the enumeration.

4.3.3 Analysis of the functional models

The following methods are applied for analysing the set of functional models.

1. IG building blocks. Extracting all functional IGs from $\mathcal{M}_f$. A *building block* of a node consists of the incoming interactions which are included in the functional IGs. Building blocks for each node can be found. The idea is to decompose all functional IGs into building blocks which can be combined to recover all functional IGs.
2. Realizability and capacity measure. Realizability of each IG for the desired property is the ratio of the functional parameters out of the parameter space. The capacity measure gives out how possible the desired property can be realised.
3. Logical analysis method on IGs and parameters. A compact logical description of the whole set of functional IGs can give an overview of the whole set, as well as uncover interesting logical IG patterns. Similarly, the logical analysis method can be applied to a set of parameters to uncover interesting logical parameter patterns.

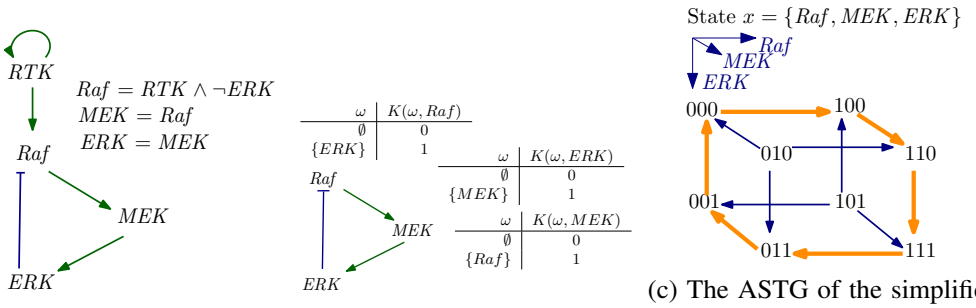
Both the forward modelling and the reverse engineering workflow are aimed to identify underlying structural features related to certain dynamic properties of the system. The forward modelling workflow searches the functional models $\mathcal{M}_f^{fw}$ from all possible models on the presumed components. By contrast, the reverse engineering workflow first enumerates all possible dynamics involving the desired property. Then it makes use of the generalised Lorenz Algorithms to infer the corresponding functional models $\mathcal{M}_f^{re}$. For the same desired property on Boolean systems, the two workflows lead to the same set of functional models. If the desired property contains a high amount of information on the dynamics, the reverse engineering workflow is preferable. Otherwise, one can choose any one of the workflows, for they have similar cost.

Chapter 5

Application: structures reproducing homeostasis

In an asynchronous state transition graph (ASTG), homeostasis corresponds to cyclic attractors. This chapter focuses on exploring the underlying GRN structures that can realise a homeostatic behaviour in the dynamics. The Mitogen-Activated Protein Kinase (MAPK) cascade is one of the most significant and well described signalling pathways in cellular systems [Orton et al., 2005]. The authors of [Orton et al., 2005] established a mathematical model of the crosstalk between the MAPK cascade and the mammalian Target of Rapamycin (mTOR) signalling pathway. These two pathways were evidenced to show a lot of mutations in cancer cells [Chappell et al., 2011, Mendoza et al., 2011, Winter et al., 2011]. A crosstalk in a signalling transduction network is like a feedback edge in an interaction graph (IG). The main focus of the thesis by [Thobe, 2017] is the crosstalk analysis between MAPK and PI3K pathways, which play an important role in cancer research. In this chapter, we consider a simplified core model of the MAPK pathway taken from [Thobe et al., 2014]. The core of the MAPK model is the Raf/MEK/ERK pathway which begins with the activation of Raf by self-activating receptor tyrosine kinases (RTK). Raf is a serine/threonine-protein kinase which can activate MEK. MEK is the MAPK/ERK kinase which activates ERK, the extracellular-signal-regulated kinase which inhibits Raf. This simplified MAPK cascade pathway is modelled by a Boolean IG of four components Raf, MEK, ERK and an input RTK. If the input RTK is on, the dynamics of the system has a cyclic attractor. If RTK is off, the system has a trivial stable state with all components off.

To focus on the cyclic attractor part, RTK is set constantly on. The IG is simplified into three components. The original rules in [Thobe et al., 2014] are transformed into a logical parameter function for the simplified IG. Figure 5.1 shows the MAPK model and the corresponding ASTG.



(a) The core of MAPK (b) Simplified MAPK model (c) The ASTG of the simplified model.

Figure 5.1: (a) The core of the MAPK cascade model and its logical rules [Thobe et al., 2014]. (b) Fix $RTK = 1$, the simplified model and the logical parameter function. (c) The ASTG of the simplified model.

For the simplified model, the ASTG has a cyclic attractor $\{$ 000, 100, 110, 111, 011, 001$\}$ which covers 6 states out of 2^3 in the state space.

5.1 Functional models preserving the cyclic attractor

Do there exist other IGs which can realise the same cyclic attractor? If the answer is yes, what are they? Are there any common features among these IGs? Applying the reverse engineering workflow, we will answer these questions.

5.1.1 Using the reverse engineering workflow

The *reverse engineering workflow* is applied. First, all functional structures for the cyclic attractor are searched.

1. All possible graphs based on $X = \{0,1\}^3$ containing the cyclic attractor are enumerated, termed as $\mathscr{G}_X$.

2. The *generalised algorithm Observability-Snoussi-Model* is applied on these graphs to infer compatible models from all graphs in $\mathscr{G}_X$.

3. Analysis of the functional models.

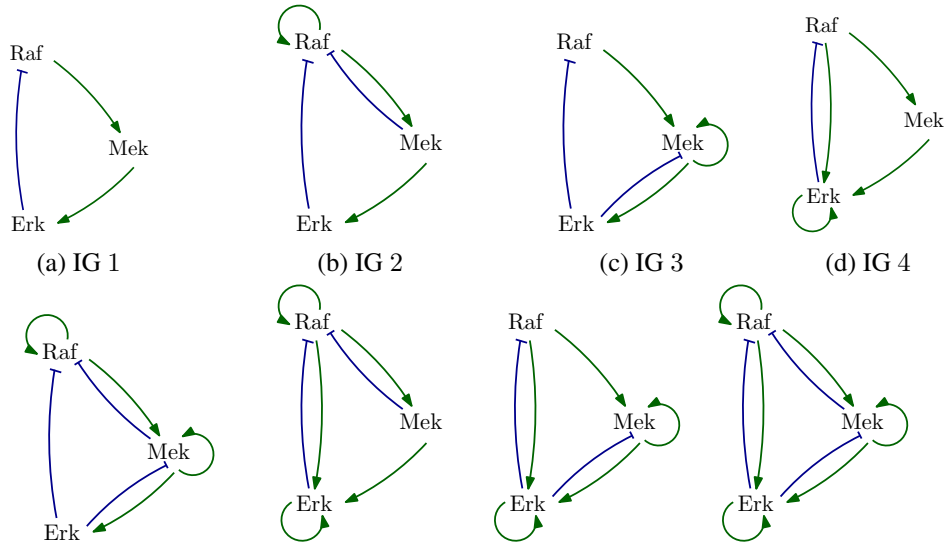
The cyclic attractor includes 6 states, $(0,0,0) \to (1,0,0) \to (1,1,0) \to (1,1,1) \to (0,1,1) \to (1,0,1) \to (0,0,0)$. In the state transition function δ, each of these 6 states has only outgoing transitions included in this cycle. The other two states $(0,1,0)$ and $(1,0,1)$ in X can have outgoing transitions in 3 directions.

A graph based on X containing the cyclic attractor means that for these 6 states no other outgoing transitions are allowed other than those in the cyclic attractor. The remaining two states $(0,1,0)$ and $(1,0,1)$ may or may not have outgoing transitions in every dimension. Thus, $2^{(2^3-6)*3} = 2^6 = 64$ graphs based on X are enumerated.

According to Proposition 3.26, all 64 graphs based on X are valid ASTGs. Applying the *generalised Observability-Snoussi-Model* algorithm, 64 compatible models are inferred from these graphs.

5.1.2 Functional IGs and their ASTGs

Among these 64 compatible models, there are only 8 different IGs, which are shown in Figure 5.2.



(a) IG 1 (b) IG 2 (c) IG 3 (d) IG 4

(e) IG 5 (f) IG 6 (g) IG 7 (h) IG 8

Figure 5.2: 8 functional IGs from the 64 functional models.

IG	# models	Show figures
IG 1	1	Figure 5.1
IG 2	3	Figure 5.3
IG 3	3	Figure 5.4
IG 4	3	Figure 5.5
IG 5	9	Figure 5.6
IG 6	9	Figure 5.7
IG 7	9	Figure 5.8
IG 8	27	Figure 5.9

Table 5.1: The IGs in Figure 5.2 are contained in multiple models, which were inferred from all enumerated ASTGs. Some of the corresponding ASTGs are shown in the figures listed in the third column.

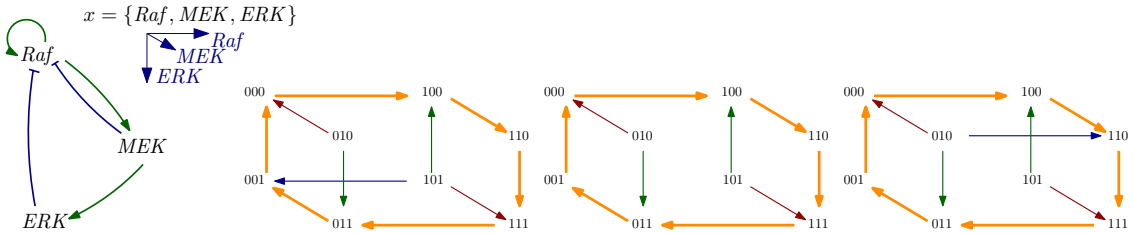
We call the ASTG in Figure 5.1 the original ASTG. The IGs 5, 6 and 7 in Figure 5.2e, 5.2f and 5.2g resp. each contain two self-loops.

1. IG 5 has two self-loops on Raf and MEK. The ASTGs related with IG 5 are different from the original ASTG in transitions in direction of Raf and MEK.

2. IG 6 has two self-loops on Raf and ERK and the ASTGs related with IG 6 have different transitions from the original ASTG in Raf and ERK.

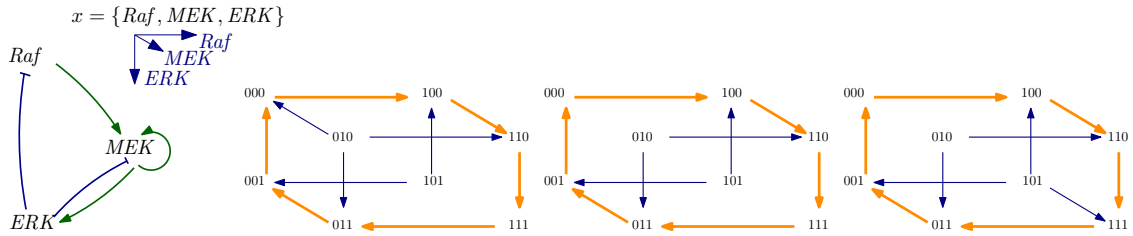
3. IG 7 has two self-loops on Raf and ERK and those ASTGs related with IG 7 differ from the original ASTG in direction of MEK and ERK.

IG 8 in Figure 5.2h has the most interactions among all 8 IGs. There are 27 models with IG 8 from 27 corresponding ASTGs. 5 out of these 27 ASTGs have been selected for illustration in Figure 5.9.



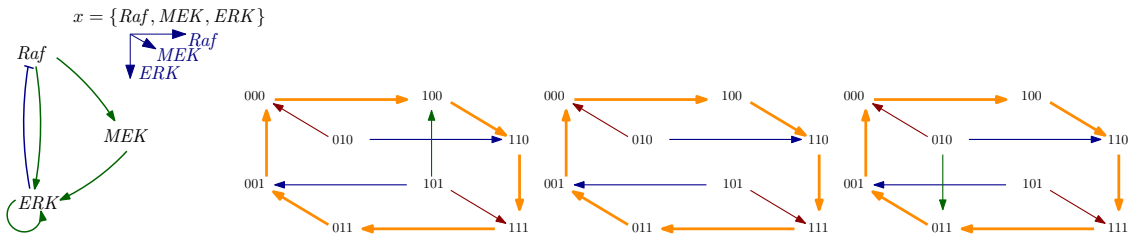
(a) IG 2 (b) ASTG 1 (c) ASTG 2 (d) ASTG 3

Figure 5.3: (a) IG 2 (5.2b) from Figure 5.2. (b), (c), (d) are the 3 ASTGs among all 64 ASTGs which carry the models with IG 2.



(a) IG 3 (b) ASTG 1 (c) ASTG 2 (d) ASTG 3

Figure 5.4: (a) IG 3 (5.2c) from Figure 5.2. (b), (c), (d) are the 3 ASTGs among all 64 ASTGs which carry the models with IG 3.



(a) IG 4 (b) ASTG 1 (c) ASTG 2 (d) ASTG 3

Figure 5.5: (a) IG 4 (5.2d) from Figure 5.2. (b), (c), (d) are the 3 ASTGs among all 64 ASTGs which carry the models with IG 4.

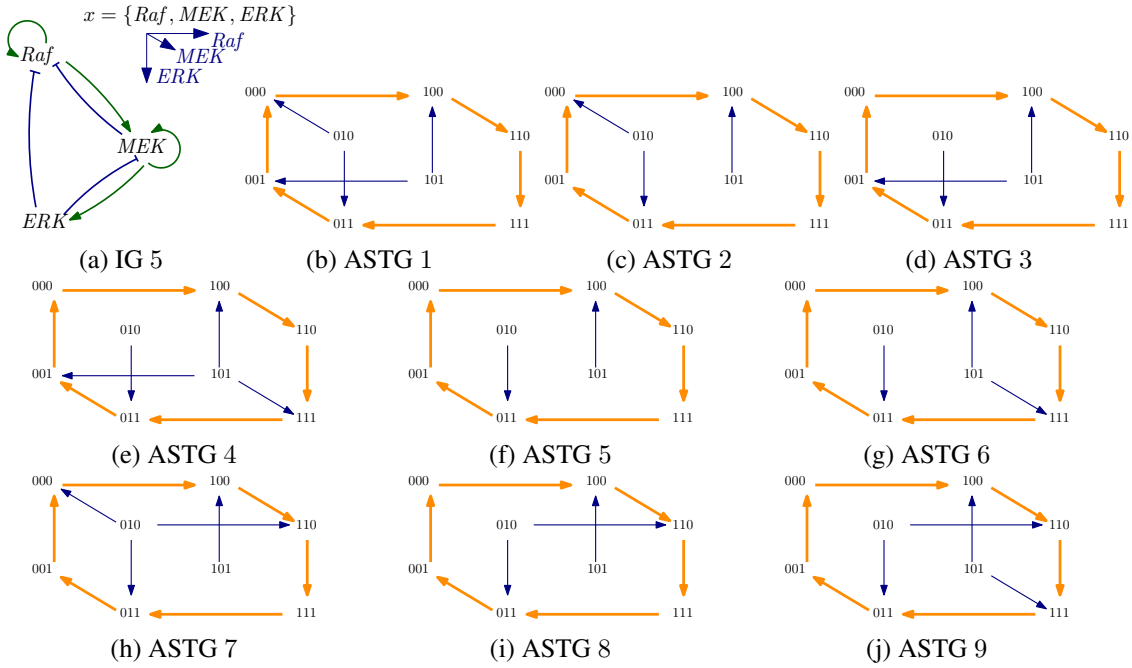


Figure 5.6: (a) IG 5 (5.2e). All 9 ASTGs ((b) to (j)) are the 9 ASTGs among all 64 ASTGs which carry the models of IG 5.

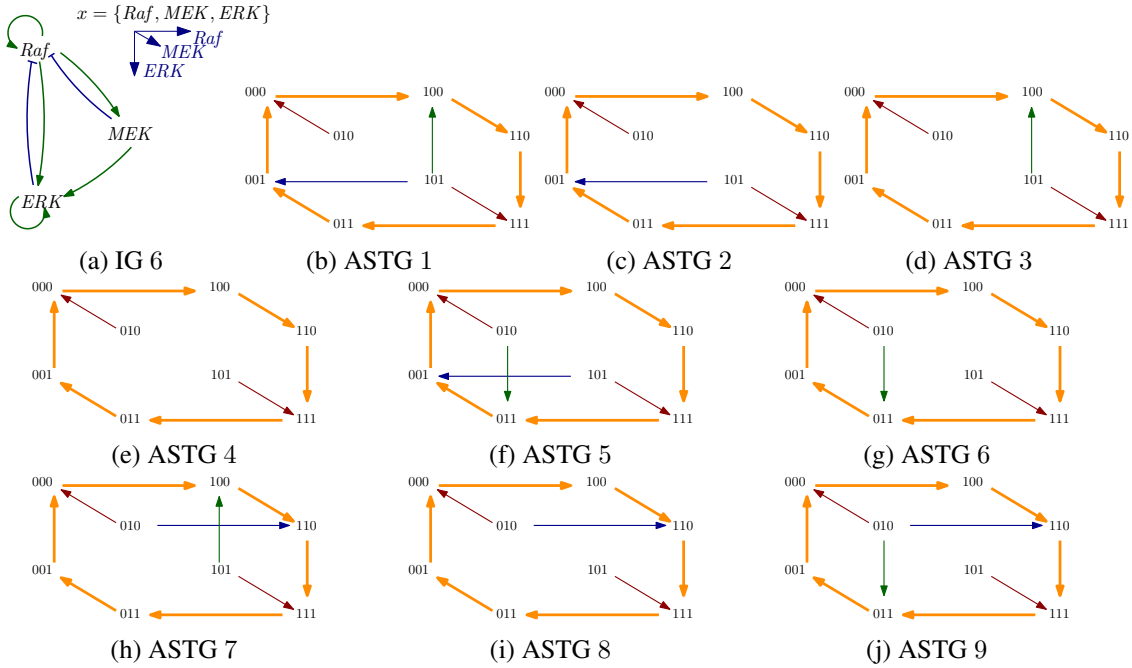
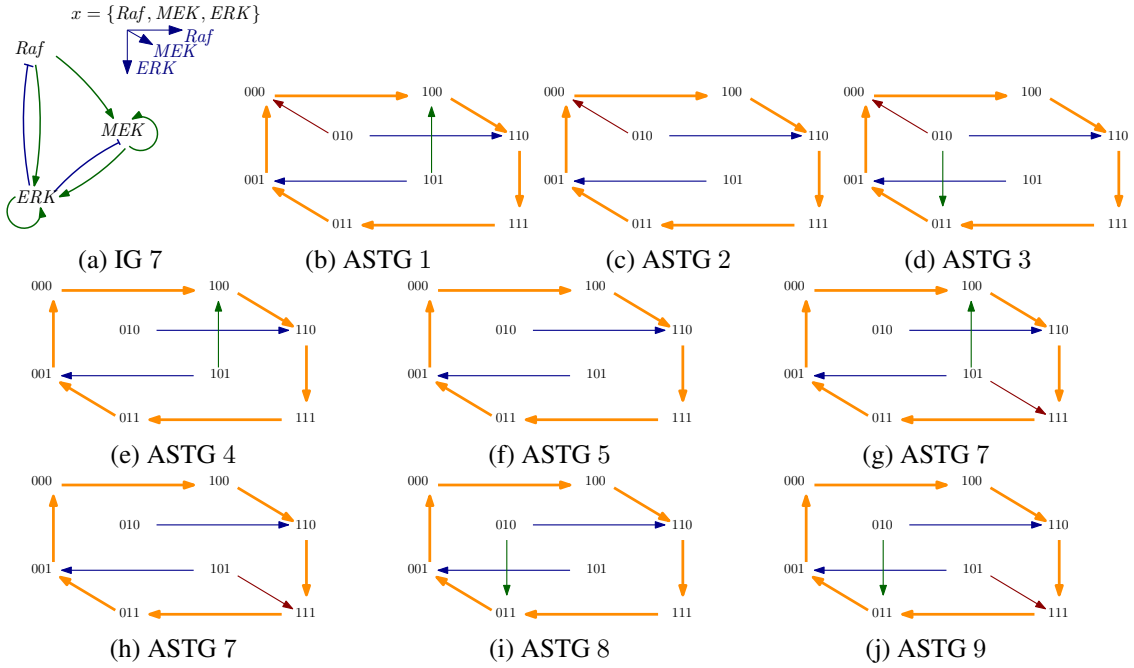


Figure 5.7: (a) IG 6 (5.2f). (b) to (j) are all 9 ASTGs among all 64 ASTGs which carry the models of IG 6.

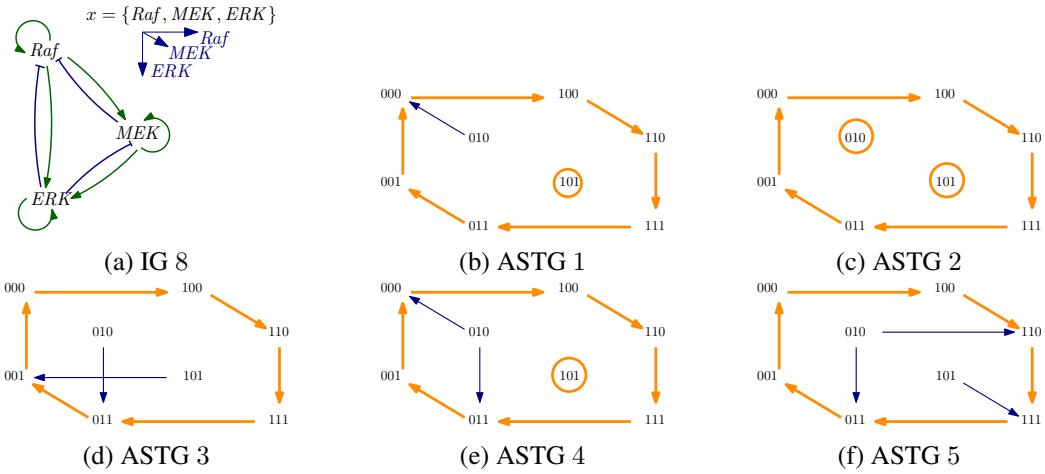


(a) IG 7 (b) ASTG 1 (c) ASTG 2 (d) ASTG 3

(e) ASTG 4 (f) ASTG 5 (g) ASTG 7

(h) ASTG 7 (i) ASTG 8 (j) ASTG 9

Figure 5.8: (a) IG 7 (5.2g). (b) to (j) are all 9 ASTGs among all 64 ASTGs which carry the models of IG 7.



(a) IG 8 (b) ASTG 1 (c) ASTG 2

(d) ASTG 3 (e) ASTG 4 (f) ASTG 5

Figure 5.9: (a) IG 8 (5.2h). (b) to (f) are 5 selected ASTGs out of 27 in total.

5.1.3 Realizability of functional IGs

The realizability (see Chapter 4) of every functional IG for the cyclic attractor has been calculated. The size of the predecessor set plays an important role in the number of compatible logical parameter functions for an IG. To illustrate this, Table 5.2 shows the following items.

- the in-degree for each component,
- the number of functional and compatible parameter and the realizability for each component of each IG
- the number of functional and compatible parameter and the realizability for each IG,

- the capacity measure of the cyclic attractor.

IG	1	2	3	4	5	6	7	8	others
in-degree Raf	1	3	1	1	3	3	1	3	
in-degree MEK	1	1	3	1	3	1	3	3	
in-degree ERK	1	1	1	3	1	3	3	3	
functional $K(Raf,\cdot)'$s	1	3	1	1	3	3	1	3	
compatible $K(Raf,\cdot)'$s	1	9	1	1	9	9	9	9	
realizability $\mathcal{R}^A_{I,Raf}$	1	$1/3$	1	1	$1/3$	$1/3$	$1/3$	$1/3$	
functional $K(MEK,\cdot)'$s	1	1	3	1	3	1	3	3	
compatible $K(MEK,\cdot)'$s	1	1	9	1	9	1	9	9	
realizability $\mathcal{R}^A_{I,MEK}$	1	1	$1/3$	1	$1/9$	1	$1/3$	$1/3$	
functional $K(ERK,\cdot)'$s	1	1	1	3	1	3	3	3	
compatible $K(ERK,\cdot)'$s	1	1	1	9	1	9	9	9	
realizability $\mathcal{R}^A_{I,ERK}$	1	1	1	$1/3$	1	$1/3$	$1/3$	$1/3$	
functional K's	1	3	3	3	9	9	9	27	$\sum = 64$
compatible K's	1	9	9	9	81	81	81	729	$\sum = 1000$
realizability $\mathcal{R}^A_I$	1	$1/3$	$1/3$	$1/3$	$1/9$	$1/9$	$1/9$	$1/27$	
capacity measure	$\sum_I$ functional K$'$s/ $\sum_I$ compatible K$'$s $= 64/1000 = 0.064$								

Table 5.2: The charts of incoming degrees, the number of functional and compatible parameters K's, and the realizability of 8 functional IGs for the cyclic attractor A. The last row is the capacity measure of the required cyclic attractor.

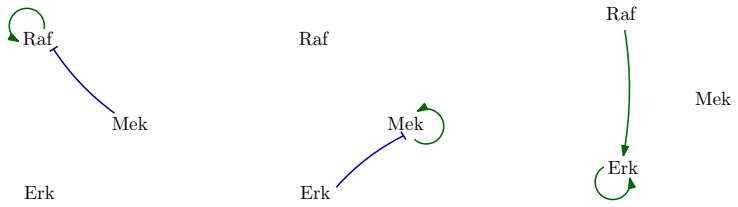
For each IG from IG 1 to IG 8, $\mathcal{R}^A_I = \mathcal{R}^A_{I,Raf}\mathcal{R}^A_{I,MEK}\mathcal{R}^A_{I,ERK}$.

5.1.4 Coupled interactions

All 8 functional IGs share IG 1 as a common subgraph (Figure 5.2a), which is called the *core motif*. Besides the core motif, 3 pairs of interactions are found in the other 7 IGs, which we call *coupled interactions*.

Definition 5.1 (Coupled interactions) Consider all functional IGs for a desired property A, two interactions are called *coupled interactions* if in every functional IG, either both of them are present, or both are absent.

(Raf, Raf) and (MEK, Raf) are coupled interactions, because both of them are present in IG 2, 5, 6, 8 and both are absent in the other IGs. Similarly, the other two coupled interactions are (MEK, MEK) and (ERK, MEK) resp. (ERK, ERK) and (Raf, ERK), as shown in Figure 5.10.



(a) Coupled interactions 1. (b) Coupled interactions 2. (c) Coupled interactions 3.

Figure 5.10: 3 pairs of coupled interactions. (a) (Raf, Raf) and (MEK, Raf). (b) (MEK, MEK) and (ERK, MEK). (c) (ERK, ERK) and (Raf, ERK). Blunt edges represent inhibitions and arrows represent activations.

The core motif together with the 8 possible combinations of the three coupled interactions yields exactly the 8 functional IGs. However, the simplest IG, which is the core motif, can already generate the desired cyclic attractor. How to understand the other IGs which have extra coupled interactions? To answer this question, the way to obtain these coupled interactions in the functional models has to be clarified.

Take IG 3 as example. Figure 5.4 shows three ASTGs whose transitions differ from the original ASTG in direction of MEK. For an input graph based on X, the generalised algorithm *Observability-Snoussi-Model* first constructs during initialisation a pseudo logical parameter function $\hat{K}$, assuming a complete positive IG. Then in the second step, an IG is inferred from $\hat{K}$.

Let the original ASTG be the input. From the information of the states in the cyclic attractor, only part of the $\hat{K}$ can be decided.

1. *Initialisation.* A pseudo logical parameter function $\hat{K}$ is constructed using the *generalised algorithm Logical-Parameters*. All 3 ASTGs share the same cyclic attractor as the original ASTG. From the cyclic attractor $(0,0,0) \to (1,0,0) \to (1,1,0) \to (1,1,1) \to (0,1,1) \to (1,0,1) \to (0,0,0)$ part of $\hat{K}$ can be constructed. For example, on component MEK, some values of $\hat{K}(MEK, \cdot)$ can be inferred already from the cyclic attractor, which will be denoted by the subscript "$_c$". Table 5.3 shows the process of obtaining part of $\hat{K}_c(MEK, \cdot)$, values $\hat{K}_o(MEK, \cdot)$ for the original ASTG, and $\hat{K}_1(MEK, \cdot)$, $\hat{K}_2(MEK, \cdot)$, $\hat{K}_3(MEK, \cdot)$ for the other 3 ASTGs related with IG 3.

2. *Inferring* (MEK, MEK) *and* (Raf, MEK). Table 5.4 and 5.5 show the inference of the interaction (MEK, MEK) and (Raf, MEK) respectively, for the original ASTG and the other three ASTGs.

In summary, each change in $\delta_o(MEK, 010)$ and $\delta_o(MEK, 101)$ will cause the appearance of the coupled interactions, *i.e.*, positive (MEK, MEK) and negative (ERK, MEK).

Similarly, if we change the transitions in the original ASTG 5.1c in direction of Raf and ERK, the other two coupled interactions will appear in the resulting IGs. IG 3 and IG 4 with the corresponding ASTGs are shown in Figure 5.4 and 5.5, respectively.

5.2 Logical analysis of functional models

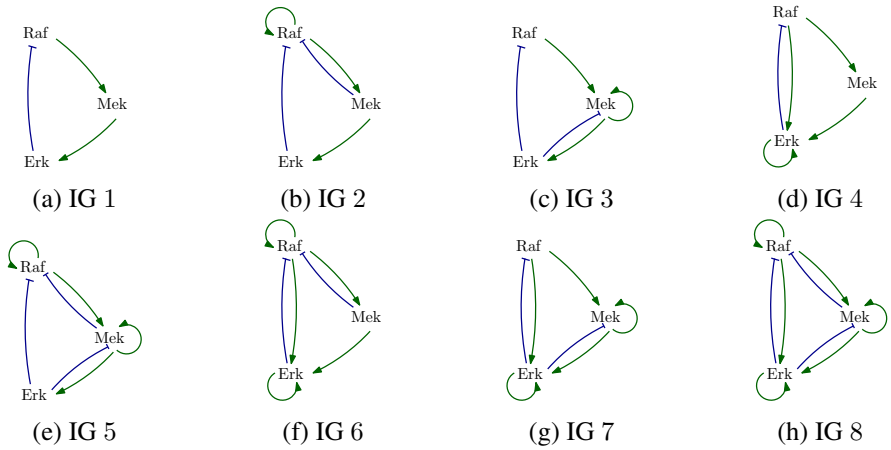
The logical analysis method introduced in Chapter 3 is now applied to the functional IGs and the logical parameter functions of some of those IGs.

$\omega \subseteq V\backslash\{MEK\}$	ϕ	$\{Raf\}$	$\{ERK\}$	$\{Raf, ERK\}$
extremal state x	**000**	**100**	**001**	101
τ^{MEK}	(**000**, 010)	(**100**, 110)	(**001**, 011)	(101, 111)
$\delta_c(MEK, \tau^{MEK})$	$(0, \sim)$	$(1, 0)$	$(0, -1)$	$(\sim, 0)$
row type	$\sim$	*open*	*open*	$\sim$
$\hat{K}_c(MEK, \omega)$	$\sim$	1	0	$\sim$
$\hat{K}_c(MEK, \omega \cup \{MEK\})$	$\sim$	1	0	$\sim$
$\delta_o(MEK, \tau^{MEK})$	$(0, -1)$	$(1, 0)$	$(0, -1)$	$(1, 0)$
row type	*open*	*open*	*open*	*open*
$\hat{K}_o(MEK, \omega)$	0	1	0	1
$\hat{K}_o(MEK, \omega \cup \{MEK\})$	0	1	0	1
$\delta_1(MEK, \tau^{MEK})$	$(0, -1)$	$(1, 0)$	$(0, -1)$	$(0, 0)$
row type	*open*	*open*	*open*	*pos*
$\hat{K}_1(MEK, \omega)$	0	1	0	0
$\hat{K}_1(MEK, \omega \cup \{MEK\})$	0	1	0	1
$\delta_2(MEK, \tau^{MEK})$	$(0, 0)$	$(1, 0)$	$(0, -1)$	$(0, 0)$
row type	*pos*	*open*	*open*	*pos*
$\hat{K}_2(MEK, \omega)$	0	1	0	0
$\hat{K}_2(MEK, \omega \cup \{MEK\})$	1	1	0	1
$\delta_3(MEK, \tau^{MEK})$	$(0, 0)$	$(1, 0)$	$(0, -1)$	$(0, 0)$
row type	*pos*	*open*	*open*	*open*
$\hat{K}_3(MEK, \omega)$	0	1	0	1
$\hat{K}_3(MEK, \omega \cup \{MEK\})$	1	1	0	0

Table 5.3: The $\hat{K}(MEK, \cdot)$ during *initialisation*: the cyclic attractor ("$_c$"), the original ASTG ("$_o$") and three ASTGs related with IG 3 ("$_1$", "$_2$", "$_3$"). Row 1: the resources $\omega \subseteq V\backslash\{MEK\}$. Row 2 and 3 show the corresponding extremal states and extremal rows. Below is the process on looking for $\hat{K}(MEK, \omega)$ and $\hat{K}(MEK, \omega \cup \{MEK\})$. The states of the cycle attractor are in bold, "$\sim$" stands for "unknown".

5.2.1 Functional IGs

The 8 IGs in Figure 5.2 can be translated into Boolean expressions. For example, $x^-_{RM} \in \{0, 1\}$ describes the existence of an inhibition of Raf by MEK, *i.e.*, $x^-_{RM} = 1$ (resp. 0) means there exists (resp. not exists) an inhibition (MEK, Raf).



(a) IG 1 (b) IG 2 (c) IG 3 (d) IG 4

(e) IG 5 (f) IG 6 (g) IG 7 (h) IG 8

Figure 5.11: 8 IGs contained in the 64 functional models for the cyclic attractor.

Each IG is encoded using the partial encoding from Section 4.2.1 with 18 Boolean variables.

$K_o(MEK, \cdot)$ original ASTG

$\omega \subseteq V \backslash \{MEK\}$	ϕ	$\{Raf\}$	$\{ERK\}$	$\{Raf, ERK\}$
$K_o(MEK, \omega)$	0	1	0	1
$K_o(MEK, \omega \cup \{MEK\})$	0	1	0	1
$\exists(MEK, MEK)?$	$\nexists$			

$K_1(MEK, \cdot)$ ASTG 1

$\omega \subseteq V \backslash \{MEK\}$	ϕ	$\{Raf\}$	$\{ERK\}$	$\{Raf, ERK\}$
$K_1(MEK, \omega)$	0	1	0	**0**
$K_1(MEK, \omega \cup \{MEK\})$	0	1	0	**1**
$\exists(MEK, MEK)?$				$\exists$
$\varepsilon(MEK, MEK)$				$+$

$K_2(MEK, \cdot)$ ASTG 2

$\omega \subseteq V \backslash \{MEK\}$	ϕ	$\{Raf\}$	$\{ERK\}$	$\{Raf, ERK\}$
$K_2(MEK, \omega)$	**0**	1	0	**0**
$K_2(MEK, \omega \cup \{MEK\})$	**1**	1	0	**1**
$\exists(MEK, MEK)?$	$\exists$			$\exists$
$\varepsilon(MEK, MEK)$	$+$			$+$

$K_3(MEK, \cdot)$ ASTG 3

$\omega \subseteq V \backslash \{MEK\}$	ϕ	$\{Raf\}$	$\{ERK\}$	$\{Raf, ERK\}$
$K_3(MEK, \omega)$	**0**	1	0	1
$K_3(MEK, \omega \cup \{MEK\})$	**1**	1	0	1
$\exists(MEK, MEK)?$	$\exists$			
$\varepsilon(MEK, MEK)$	$+$			

Table 5.4: Inferring (MEK, MEK): original ASTG ("$_o$") and the 3 ASTGs from IG 3 ("$_1$", "$_2$", "$_3$").

To get a compact logical description of the 8 IGs, the Quine-McCluskey algorithm is applied using the software tool PyBoolNet [Klarner et al., 2016] on the disjunction (or sum) of these 8 Boolean expressions. It turns out that the minimal disjunctive normal form (DNF) of the 8 Boolean expressions is the same as the input, which means that it cannot be further simplified. However, the conjunctive normal form (CNF) of these 8 IGs gives a shorter expression and characterises the core and the three coupled interactions in a meaningful way. The process of getting the minimal CNF is now further described.

From Table 5.6, we can get the following knowledge on the 8 IGs.

1. $x^-_{RR} \equiv 0$, $x^-_{MM} \equiv 0$, $x^-_{EE} \equiv 0$, because the three self-loops (Raf, Raf), (MEK, MEK) and (ERK, ERK) are never negative.

2. $x^+_{RM} \equiv 0$, $x^+_{RE} \equiv 0$, for there does not exist positive (Raf, MEK) and positive (Raf, ERK).

3. $x^-_{MR} \equiv 0$, $x^+_{ME} \equiv 0$, for there does not exist negative (Raf, MEK) and positive (MEK, ERK).

4. $x^-_{ER} \equiv 0$, $x^-_{EM} \equiv 0$, for there does not exist negative (ERK, Raf) and negative (ERK, MEK).

Therefore, only 9 Boolean variables are needed to encode an IG, which are for Raf, $\{x^+_{RR}$, x^-_{RM}, $x^-_{RE}\}$, for MEK, $\{x^+_{MM}$, x^+_{MR}, $x^-_{Me}\}$ and for ERK, $\{x^+_{EE}$, x^+_{ER}, $x^+_{EM}\}$. Using these 9 variables, the 8 IGs are transformed into the following expressions:

$K_o(MEK,\cdot)$ original ASTG

$\omega \subseteq V\backslash\{ERK\}$	ϕ	$\{Raf\}$	$\{MEK\}$	$\{Raf, MEK\}$
$K_o(MEK,\omega)$	0	1	0	1
$K_o(MEK,\omega\cup\{ERK\})$	0	1	0	1
$\exists(ERK, MEK)?$			$\nexists$	

$K_1(MEK,\cdot)$ ASTG 1

$\omega \subseteq V\backslash\{ERK\}$	ϕ	$\{Raf\}$	$\{MEK\}$	$\{Raf, MEK\}$
$K_1(MEK,\omega)$	0	**1**	0	1
$K_1(MEK,\omega\cup\{ERK\})$	0	**0**	0	1
$\exists(ERK, MEK)?$		$\exists$		
$\varepsilon(ERK, MEK)$		$-$		

$K_2(MEK,\cdot)$ ASTG 2

$\omega \subseteq V\backslash\{ERK\}$	ϕ	$\{Raf\}$	$\{MEK\}$	$\{Raf, MEK\}$
$K_2(MEK,\omega)$	0	**1**	**1**	1
$K_2(MEK,\omega\cup\{ERK\})$	0	**0**	**0**	1
$\exists(ERK, MEK)?$		$\exists$	$\exists$	
$\varepsilon(ERK, MEK)$		$-$	$-$	

$K_3(MEK,\cdot)$ ASTG 3

$\omega \subseteq V\backslash\{ERK\}$	ϕ	$\{Raf\}$	$\{MEK\}$	$\{Raf, MEK\}$
$K_3(MEK,\omega)$	0	1	**1**	1
$K_3(MEK,\omega\cup\{ERK\})$	0	1	**0**	1
$\exists(ERK, MEK)?$			$\exists$	
$\varepsilon(ERK, MEK)$			$-$	

Table 5.5: Inferring (ERK, MEK): original ASTG ("$_o$") and the 3 ASTGs from IG 3 ("$_1$", "$_2$", "$_3$").

Edges	(R,R)	(R,M)	(R,E)	(M,R)	(M,M)	(M,E)	(E,R)	(E,M)	(E,E)
IG 1 :	$\overline{x^+_{RR}}\,\overline{x^-_{RR}}$	$\overline{x^+_{RM}}\,\overline{x^-_{RM}}$	x^-_{RE}	x^+_{MR}	$\overline{x^+_{MM}}\,\overline{x^-_{MM}}$	$\overline{x^+_{ME}}\,\overline{x^-_{ME}}$	$\overline{x^+_{ER}}\,\overline{x^-_{ER}}$	x^+_{EM}	$\overline{x^+_{EE}}\,\overline{x^-_{EE}}$
IG 2 :	x^+_{RR}	x^-_{RM}	x^-_{RE}	x^+_{MR}	$\overline{x^+_{MM}}\,\overline{x^-_{MM}}$	$\overline{x^+_{ME}}\,\overline{x^-_{ME}}$	$\overline{x^+_{ER}}\,\overline{x^-_{ER}}$	x^+_{EM}	$\overline{x^+_{EE}}\,\overline{x^-_{EE}}$
IG 3 :	x^+_{RR}	$\overline{x^+_{RM}}\,\overline{x^-_{RM}}$	x^-_{RE}	$x^+_{MR}x^+_{MM}$	x^-_{ME}	$\overline{x^+_{ER}}\,\overline{x^-_{ER}}$	x^+_{EM}	$\overline{x^+_{EE}}\,\overline{x^-_{EE}}$	
IG 4 :	$\overline{x^+_{RR}}\,\overline{x^-_{RR}}$	$\overline{x^+_{RM}}\,\overline{x^-_{RM}}$	x^-_{RE}	x^+_{MR}	$\overline{x^+_{MM}}\,\overline{x^-_{MM}}$	$\overline{x^+_{ME}}\,\overline{x^-_{ME}}$	x^+_{ER}	x^+_{EM}	x^+_{EE}
IG 5 :	x^+_{RR}	x^-_{RM}	x^-_{RE}	x^+_{MR}	x^+_{MM}	x^-_{ME}	$\overline{x^+_{ER}}\,\overline{x^-_{ER}}$	x^+_{EM}	$\overline{x^+_{EE}}\,\overline{x^-_{EE}}$
IG 6 :	x^+_{RR}	x^-_{RM}	x^-_{RE}	x^+_{MR}	$\overline{x^+_{MM}}\,\overline{x^-_{MM}}$	$\overline{x^+_{ME}}\,\overline{x^-_{ME}}$	x^+_{ER}	x^+_{EM}	x^+_{EE}
IG 7 :	$\overline{x^+_{RR}}\,\overline{x^-_{RR}}$	$\overline{x^+_{RM}}\,\overline{x^-_{RM}}$	x^-_{RE}	x^+_{MR}	x^+_{MM}	x^-_{ME}	x^+_{ER}	x^+_{EM}	x^+_{EE}
IG 8 :	x^+_{RR}	x^-_{RM}	x^-_{RE}	x^+_{MR}	x^+_{MM}	x^-_{ME}	x^+_{ER}	x^+_{EM}	x^+_{EE}

Table 5.6: Encoding the 8 functional IGs with 18 Boolean variables. The Boolean expression of each IG is the product of all variables in the row.

Edges	(R,R)	(R,M)	(R,E)	(M,R)	(M,M)	(M,E)	(E,R)	(E,M)	(E,E)
IG 1 :	$\overline{x^+_{RR}}$	$\overline{x^-_{RM}}$	x^-_{RE}	x^+_{MR}	$\overline{x^+_{MM}}$	$\overline{x^-_{ME}}$	$\overline{x^+_{ER}}$	x^+_{EM}	$\overline{x^+_{EE}}$
IG 2 :	x^+_{RR}	x^-_{RM}	x^-_{RE}	x^+_{MR}	$\overline{x^+_{MM}}$	$\overline{x^-_{ME}}$	$\overline{x^+_{ER}}$	x^+_{EM}	$\overline{x^+_{EE}}$
IG 3 :	x^+_{RR}	$\overline{x^-_{RM}}$	x^-_{RE}	x^+_{MR}	x^+_{MM}	x^-_{ME}	$\overline{x^+_{ER}}$	x^+_{EM}	$\overline{x^+_{EE}}$
IG 4 :	$\overline{x^+_{RR}}$	$\overline{x^-_{RM}}$	x^-_{RE}	x^+_{MR}	$\overline{x^+_{MM}}$	$\overline{x^-_{ME}}$	x^+_{ER}	x^+_{EM}	x^+_{EE}
IG 5 :	x^+_{RR}	x^-_{RM}	x^-_{RE}	x^+_{MR}	x^+_{MM}	x^-_{ME}	$\overline{x^+_{ER}}$	x^+_{EM}	$\overline{x^+_{EE}}$
IG 6 :	x^+_{RR}	x^-_{RM}	x^-_{RE}	x^+_{MR}	$\overline{x^+_{MM}}$	$\overline{x^-_{ME}}$	x^+_{ER}	x^+_{EM}	x^+_{EE}
IG 7 :	$\overline{x^+_{RR}}$	$\overline{x^-_{RM}}$	x^-_{RE}	x^+_{MR}	x^+_{MM}	x^-_{ME}	x^+_{ER}	x^+_{EM}	x^+_{EE}
IG 8 :	x^+_{RR}	x^-_{RM}	x^-_{RE}	x^+_{MR}	x^+_{MM}	x^-_{ME}	x^+_{ER}	x^+_{EM}	x^+_{EE}

Table 5.7: Encoding the 8 functional IGs with 9 Boolean variables. The Boolean expression of each IG is the product of all variables in the row.

Next we apply the answer set programming (ASP) code by [Becker et al., 2016] to obtain the minimal CNF. The minimal CNF of these 8 IGs is

$$\begin{aligned}\text{minCNF} = {} & x^+_{EM}x^-_{RE}x^+_{MR}(\overline{x^+_{ER}} + x^+_{EE})(x^+_{ER} + \overline{x^+_{EE}}) \\ & (\overline{x^+_{MM}} + x^-_{ME})(x^+_{MM} + \overline{x^-_{ME}})(\overline{x^+_{RR}} + x^-_{RM})(x^+_{RR} + \overline{x^-_{RM}}).\end{aligned} \tag{5.1}$$

This CNF yields a very compact Boolean representation of all 8 functional IGs and can be explained in four parts.

1. $x^+_{EM}x^-_{RE}x^+_{MR}$, denotes the three interactions in the core motif;
2. $(\overline{x^+_{ER}} + x^+_{EE})\ (x^+_{ER} + \overline{x^+_{EE}}) = (x^+_{ER}x^+_{EE} + \overline{x^+_{ER}}\ \overline{x^+_{EE}})$, denotes the coupled interactions (Raf, ERK) and (ERK, ERK);
3. $(x^+_{MM} + \overline{x^-_{ME}}) = (x^+_{MM}x^-_{ME} + \overline{x^+_{MM}}\ \overline{x^-_{ME}})$, denotes the coupled interactions (MEK, ERK) and (MEK, MEK);
4. $(\overline{x^+_{RR}} + x^-_{RM})(x^+_{RR} + \overline{x^-_{RM}}) = (x^+_{RR}x^-_{RM} + \overline{x^+_{RR}}\ \overline{x^-_{RM}})$, denotes the coupled interactions (Raf, Raf) and (ERK, Raf).

5.2.2 Functional parameters

In the 8 IGs, a component can have at most $2^3 = 8$ resources. If we use one Boolean variable per logical parameter $K(node, res)$, then 24 Boolean variables are needed to encode a logical parameter function. However, for this number of variables, it is already very hard to compute a minimal logical representation.

Applying the logical analysis method directly to the functional K's is not practical. However, the regulations of one component and those of another component do not depend on each other. Thus, the logical parameters of one component and those of another component are also independent. Therefore, without changing the final representation, the logical analysis method can be applied individually on the logical parameters of each component, which requires at most 8 variables.

In the 64 logical parameter functions, there are only 4 different functions for v, namely K_v, for each $v \in \{Raf, MEK, ERK\}$. Among these 4 functions for v, there are 3 functions for 8 resources where $|\mathrm{Pre}(v) = 3|$ and 1 function is for 2 resources where $|\mathrm{Pre}(v)| = 1$.

The logical analysis method is applied to get a compact description of the 3 functions for v where v has 8 resources. Table 5.8, 5.9, 5.10 shows the results of using the logical analysis method on the functions of all three components K_{Raf}, K_{MEK} and K_{ERK}.

Res $\rightarrow$	$\emptyset$	$\{ERK\}$	$\{MEK\}$	$\{MEK, ERK\}$	$\{Raf\}$	$\{Raf, ERK\}$	$\{Raf, MEK\}$	$\{Raf, MEK, ERK\}$	f
K^1_{Raf}	0	0	0	1	0	1	0	1	1
K^2_{Raf}	0	0	0	1	0	1	1	1	1
K^3_{Raf}	0	1	0	1	0	1	1	1	1
$K^{p_1}_{Raf}$	0	0	0	1	0	1	–	1	$(K^1 + K^2)$
$K^{p_2}_{Raf}$	0	–	0	1	0	1	1	1	$(K^2 + K^3)$

Table 5.8: Minimal representation of 3 K_{Raf}'s, 2 K^p_{Raf}'s. "–" denotes either 0 or 1.

In Table 5.8, the functions K^i_{Raf} for $i \in \{1, \ldots, 3\}$ are first transformed into Boolean expressions. For each K^i_{Raf}, let $x_{\mathrm{Res}_1} = 1$ or 0 if $K^i(Raf, \mathrm{Res}_1) = 1$ or 0, respectively. For example,

K^2_{Raf} is encoded as $\overline{x_1}\ \overline{x_2}\ \overline{x_3}x_4\overline{x_5}x_6x_7x_8$. The sum of all 3 Boolean expressions is given to PyBoolNet to calculate the minimal DNF. The resulting minimal DNF is $\overline{x_1x_2x_3}x_4\overline{x_5}x_6x_8 + \overline{x_1x_3}x_4\overline{x_5}x_6x_7x_8$. The resulting minimal DNF is transformed back into the form of logical parameters, so called K^p_{Raf} patterns.

Therefore, if $|\mathrm{Pre}(Raf)| = 3$, then K_{Raf} can be represented by $K^{p1}_{Raf} + K^{p2}_{Raf}$. If $\mathrm{Pre}(Raf) = \{ERK\}$, then there is only one choice: $K(Raf, \emptyset) = 0$ and $K(Raf, \{ERK\}) = 1$.

Similar analysis can be done for MEK and ERK. The results are shown in Table 5.9 and Table 5.10, respectively.

Res →	$\emptyset$	$\{ERK\}$	$\{MEK\}$	$\{MEK, ERK\}$	$\{Raf\}$	$\{Raf, ERK\}$	$\{Raf, MEK\}$	$\{Raf, MEK, ERK\}$	f
K^1_{MEK}	0	0	0	0	0	1	1	1	1
K^2_{MEK}	0	0	0	1	0	1	1	1	1
K^3_{MEK}	0	0	0	1	1	1	1	1	1
$K^{p_1}_{MEK}$	0	0	0	–	0	1	1	1	$(K^1 + K^2)$
$K^{p_2}_{MEK}$	0	0	0	1	–	1	1	1	$(K^2 + K^3)$

Table 5.9: Minimal representation of 3 K^i_{MEK}s, 2 K^p_{MEK}. "–" denotes either 0 or 1.

Res →	$\emptyset$	$\{ERK\}$	$\{MEK\}$	$\{MEK, ERK\}$	$\{Raf\}$	$\{Raf, ERK\}$	$\{Raf, MEK\}$	$\{Raf, MEK, ERK\}$	f
K^1_{ERK}	0	0	0	1	0	0	1	1	1
K^2_{ERK}	0	0	0	1	0	1	1	1	1
K^4_{ERK}	0	0	1	1	0	1	1	1	1
$K^{p_1}_{ERK}$	0	0	0	1	0	–	1	1	$(K^1 + K^2)$
$K^{p_2}_{ERK}$	0	0	–	1	0	0	1	1	$(K^2 + K^3)$

Table 5.10: Minimal representation of 3 K^i_{ERK}s, 2 K^p_{ERK}. "–" denotes either 0 or 1.

5.3 Conclusion and discussion

This chapter studied all structures that can reproduce the homeostasis of a simplified 3-node MAPK-cascade signalling network. The cyclic attractor in the 3-dimensional Boolean state space is $(000 \to 100 \to 110 \to 111 \to 101 \to 001 \to 000)$. [Perkins et al., 2010] investigated a similar cyclic attractor originating from a repressilator system of 3 Boolean components (v_1 inhibits v_2 which inhibits v_3 which inhibits v_1), which is $(100 \to 101 \to 001 \to 011 \to 010 \to 110 \to 100)$. The ASTG and IG from [Perkins et al., 2010] and those from this Chapter are shown in Figure 5.12. Both ASTGs are based on a 3-node Boolean system. One can observe that v_1 receives only one inhibition from v_3 and similarly, only ERK inhibits Raf. The state transitions in direction of v_1 are isomorphic to those in direction of Raf. Moreover, the incoming regulations of v_2 and v_3 are inhibitions, while for MEK and ERK there are activations. The state transitions for v_2 and v_3 are all in opposite direction to those for MEK and ERK, respectively.

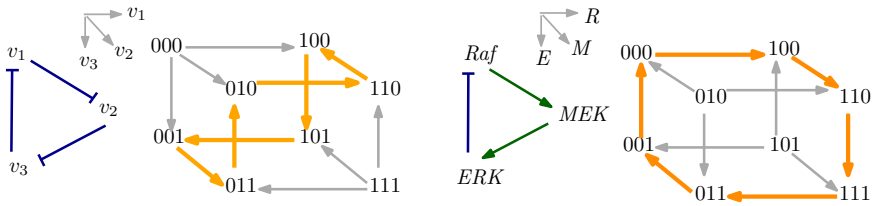


Figure 5.12: The repressilator system and its ASTG in [Perkins et al., 2010] (left) and those from this chapter (right).

The reverse engineering workflow was applied. Starting from the cyclic attractor in the ASTG of the simplified 3-node model, 64 graphs based on the state space $\{0,1\}^3$ were enumerated. From these, 64 functional models were obtained using the *generalised algorithm Observability-Snoussi-Model*. The results show that the 64 models contain only 8 functional IGs, which consist of a core motif and three pairs of coupled interactions.

All 64 enumerated graphs based on $\{0,1\}^3$ were verified to be ASTGs and all carry a compatible model. If an ASTG can be obtained by changing some transitions of the original ASTG, then we call it a *modification* of the original ASTG. Correspondingly, the IG of the model obtained from a modified ASTG is usually different from the original IG. While keeping the cyclic attractor in the enumerated ASTGs, modifications are only allowed on the outgoing transitions from the states 010 and 101. We have the following observations.

1. If a modification is done on a state $x \in \{010, 101\}$ in direction of $v \in V$, then the IG of the new model changes by a pair of coupled interactions of v, see examples in Figures 5.3, 5.4 and 5.5.
2. If modifications are done on transitions of two states $\{010, 101\}$ in direction of $v \in V$, then the IG of the new model changes by a pair of coupled interactions of v, see examples in Figures 5.3, 5.4 and 5.5.
3. If modifications are done on one state or two states in more than one directions, then the IG of the new model changes by the corresponding pairs of coupled interactions of the revised components. This can be seen as a combination of the IG changes caused by each individual modification, see examples in Figures 5.6, 5.7 and 5.8.

Section 5.1.2 showed all 8 functional IGs for the required cyclic attractor. IG 1 has the 3 interactions, one compatible model which is 1 functional, therefore, the realizability is 1. The 3 interactions are the core motif which are contained in all 8 functional IGs. IG 2, 3 and 4 has the core motif and one pair of coupled interactions. Each of them has 9 compatible models of which 3 are functional. IG 8 has the highest number of interactions with 27 functional models. The core motif is needed for all IGs to be functional.

The logical analysis method has been applied in Section 5.2 both on functional IGs and logical parameter function. Firstly, the 8 functional IGs are encoded into Boolean expressions, and the Quine-McCluskey algorithm is applied using the software PyBoolNet to get a minimal DNF. The resulting minimal DNF is still the full sum of the 8 IGs. However, the minimal CNF obtained by using the ASP code from [Becker et al., 2016] can characterise these 8 IGs in a shorter way. Among the 64 K's contained in all 64 models, there are only 4 distinctive functions K_v for each component v. Therefore, the logical analysis method could be applied on the logical parameters of individual components to get a compact description, which is given in Table 5.8, 5.9 and 5.10.

Chapter 6

Application: structures reproducing multistability

Multistability is an important dynamical property referring to the existence of multiple stable states or attractors in a regulatory network. In biology, this can be related to cell differentiation or fate decisions. To analyse the underlying regulatory structures, both continuous and discrete methods can be applied. Using a continuous framework together with a robustness measure to decide whether a GRN can generate the required functionality, [Breindl et al., 2010, Schittler et al., 2010, Breindl et al., 2011] studied all GRNs composed of 3 transcriptional regulators (TRs) that are able to produce 3 distinct stable steady states.

Here, we explore this question using a discrete modelling approach. The *forward* modelling workflow enumerates all possible regulatory structures together with their regulatory rules and searches for models that can generate the required multistability. The *reverse* engineering workflow enumerates all graphs based on the state space that contain the required stable states. Then it uses the generalised Lorenz algorithms to infer all the underlying models. The results from the two workflows can also be used to give a validation of each other.

The continuous method from [Breindl et al., 2011] will be introduced in Section 6.1. Afterwards in Section 6.2, the initial requirements from Breindl et al. are adapted to the discrete setting. The two discrete modelling workflows will be applied to obtain all functional IGs for the required multiple stable states. In particular, in the reverse engineering workflow, we may or may not allow the stable states to be isolated and the existence of other attractors. As a result, using the discrete modelling approach, many more functional IGs are discovered than with the continuous method. The different results in terms of their building blocks are compared in Section 6.3. Realizability and the logical analysis method are applied to analyse the functional IGs under each hypothesis. Additionally, the logical analysis method is applied to compare the results of the two hypotheses.

6.1 Continuous method: robustness measure

[Breindl et al., 2011] studied the structure and the robustness of small networks composed of three interacting transcriptional regulators (TRs) crucial for cell differentiation. They applied a modelling framework based on ordinary differential equations (ODEs) (introduced in Sec 1.2), which uses qualitative information on the network and will be introduced in Section 6.1.1. With the help of a robustness measure on the network structure presented in Section 6.1.3, they are

able to find those regulatory structures which can reproduce certain specified stable steady states (Section 6.1.2).

6.1.1 Continuous modelling framework

The regulations between the TRs are described by monotonous activation and inhibition functions.

Definition 6.1 (Activation and Inhibition Function) [Breindl et al., 2011] An *activation (inhibition) function* is $\nu : [0, \infty) \to [0, N) (\mu : [0, \infty) \to (0, N])$ with $N \in \mathbb{R}_+$, and

i) ν (μ) is continuously differentiable,

ii) $\nu(0) = 0$ and $\nu(x) \to N$ as $x \to \infty$ ($\mu(0) = N$ and $\mu(x) \to 0$ as $x \to \infty$),

iii) $\nu(x)$ ($\mu(x)$) is *monotonously* increasing (decreasing).

This definition covers commonly used rate laws like Michaelis-Menten and Hill kinetics. The dynamics of a regulatory network with n TRs is described by

$$\dot{x}_i = -k_i x_i + f_i(x),\ i = 1, \ldots, n. \tag{6.1}$$

Here $x \in \mathbb{R}^n_+$ is the concentration vector of the TRs, the term $-k_i x_i$ describes linear degradation and f_i is a function representing all the influences from the TRs in the network on TR i. In what follows, the symbol φ denotes both ν and μ, while "$\circ$" stands for both sum '$+$' and multiplication "$\cdot$". The function f_i then can be written as

$$f_i(x) = \varphi_{i,j_1}(x_{j_1}) \circ \cdots \circ \varphi_{i,j_k}(x_{j_k}), \tag{6.2}$$

where $\{j_1, \ldots, j_k\} \subseteq \{1, \ldots, n\}$. For the activation or inhibition function $\varphi_{i,j_q}(x_{j_q})$, with $q \in \{1, \ldots, k\}$, i is a TR which is regulated by the TRs $j_1, \ldots, j_k$, where j_q specifies the q-th transcription factor and k denotes the number of TRs. It is assumed that each TR can only either activate, inhibit, or has no influence on the production of another TR. This means that a variable x_{j_q} can be the argument of at most one function φ_{i,j_q}, $q \in \{1, \ldots, k\}$, for each $i \in \{1, \ldots, n\}$. Because k_i and the exact shape of the activation or inhibition function $\varphi_{i,j_k}(x_{j_k})$ are generally unknown, the equation (6.1) only captures the topology of the network.

An IG of three components is represented by an interaction matrix:

$$A = \{a_{ij}\}_{i,j\in\{1,2,3\}},\ a_{ij} := \begin{cases} 1 & j \text{ activates } i \\ -1 & j \text{ inhibits } i \\ 0 & \text{no interaction} \end{cases}. \tag{6.3}$$

To show an example of the interaction matrix:

$$A = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & -1 \\ 1 & 0 & -1 \end{bmatrix}.$$

Then the network will be:

$$\begin{aligned} \dot{x}_1 &= -k_1 x_1 + \nu_1(x_1) \cdot \mu_2(x_2), \\ \dot{x}_2 &= -k_2 x_2 + \nu_3(x_1) \cdot \nu_4(x_2) \cdot \mu_5(x_3), \\ \dot{x}_3 &= -k_3 x_3 + \nu_6(x_1) \cdot \mu_7(x_3). \end{aligned}$$

6.1.2 Specification of stable steady states

For each TR, concentration levels x_i^{low}, x_i^{high} and $x_i^{\max}$ with $0 \leq x_i^{low} \leq x_i^{high} \leq x_i^{\max}$ are specified. A concentration x_i in the interval $\mathcal{I}_{x_i}^{low} = [0, x_i^{low}]$ or $x_i \in \mathcal{I}_{x_i}^{high} = [x_i^{high}, x_i^{\max}]$ is considered as *low* or *high* respectively.

Definition 6.2 (Forward-invariant set) [Breindl et al., 2010] A set $\mathcal{F} \in \mathbb{R}^n$ is forward-invariant for system (6.1) if, for each initial condition $x(0) = x_0 \in \mathcal{F}$, its solution $x(t; x_0)$ remains in $\mathcal{F}$, *i.e.*, for all $t > 0 : x(t; x_0) \in \mathcal{F}$.

The stable steady states of the network determine the cell types. In the state space, for the variability between cells and the uncertainties during measurements, the concentration values of the components are considered to be intervals instead of exact values [Breindl et al., 2010]. So that, it is assumed that a steady state can be represented by a forward-invariant hyperrectangular set $x = \mathcal{I}_{x_1}^{l_1} \times \cdots \times \mathcal{I}_{x_n}^{l_n}$ with $l_i \in \{low, high\}$. Multiple stable steady states are treated as a group of forward-invariant sets in the state space.

[Breindl et al., 2011] are interested in a set of three stable steady states: a progenitor factor *A*, a differentiated cell type *B*, and a competing differential cell type *C*, which are given as $x^{(i)} = (l_1^{(i)}, l_2^{(i)}, l_3^{(i)}), i \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$. The states $x^{(\mathrm{B})}$ and $x^{(\mathrm{C})}$ are assumed to carry a high level of x_2 and x_3, respectively. For $x^{(\mathrm{A})}$, two opposite specifications of stable steady states are assumed according to whether TR x_1 works as a progenitor or differentiation factor.

(S1) x_1 is a progenitor factor. The cell achieves differentiation by *down-regulating* it.

$$\begin{aligned} x^{(\mathrm{A})} &= (high, low, low) \\ x^{(\mathrm{B})} &= (low, high, low) \\ x^{(\mathrm{C})} &= (low, low, high) \end{aligned} \tag{6.4}$$

(S2) x_1 is a differentiation factor. The cell achieves differentiation by *up-regulating* it.

$$\begin{aligned} x^{(\mathrm{A})} &= (low, low, low) \\ x^{(\mathrm{B})} &= (high, high, low) \\ x^{(\mathrm{C})} &= (high, low, high) \end{aligned} \tag{6.5}$$

6.1.3 Robustness measure and its computation

A robustness measure $\mathcal{R}$ is defined to determine to what extent a regulatory network can reproduce the required multistability [Breindl et al., 2010, Breindl et al., 2011]. The idea of $\mathcal{R}$ is to quantify whether and how well a system (6.1) can generate the forward-invariant sets in (6.4) and (6.5). The definition of $\mathcal{R}$ involves several steps [Breindl et al., 2010].

Definition 6.3 (Perturbation measure) [Breindl et al., 2010]
Given an activation (inhibition) function φ and a perturbed function φ^p, the integral

$$\mathcal{P}(\varphi - \varphi^p) = \int_0^\infty |\varphi(x) - \varphi^p(x)| \mathrm{d}x$$

is a measure for the perturbation of φ.

The desired cell types are characterised by the forward-invariant sets $\mathbf{x}^z, z = \{1, \ldots, m\}$. [Breindl et al., 2011] define the robustness value $\mathcal{R} \in \mathbb{R}$ in terms of the interaction matrix $\mathbf{A}$. The basic intuition is as follows:

1. If a value $\mathcal{R}$ is assigned to the system $\mathbf{A}$, then there exist monotonous functions φ_{i,j_q} such that $\mathbf{x}^z, z = 1, \ldots, m$ are forward-invariant;

2. If all the monotonous functions are perturbed by $\mathcal{R}$ or less, *i.e.*, for all i, j_q, $\mathcal{P}(\varphi_{i,j_q} - \varphi^p_{i,j_q}) \leq \mathcal{R}$, then $\mathbf{x}^z, z = 1, \ldots, m$ remain forward-invariant;

3. If any of the monotonous function is perturbed by more than $\mathcal{R}$, *i.e.*, there exists i, j_q, $\mathcal{P}(\varphi_{i,j_q} - \varphi^p_{i,j_q}) > \mathcal{R}$, then $\mathbf{x}^z, z = 1, \ldots, m$ are no longer forward-invariant.

Two more notions are needed before giving the definition of $\mathcal{R}$. A tube for an activation (inhibition) function is defined next and illustrated in Figure 6.1.

Definition 6.4 (Tube for an activation (inhibition) function) [Breindl et al., 2010]
A 3-tuple of pairs of positive real numbers

- $T_\nu = \Big((x^{low}, \gamma^{low}), (x^{high}, \gamma^{high}), (x^{\max}, \gamma^{\max})\Big)$ with $\gamma^{low} \leq \gamma^{high} \leq \gamma^{\max}$ and $x^{low} \leq x^{high} \leq x^{\max}$ is called a *tube for an activation function*.

- $T_\mu = \Big((x^{low}, \gamma^{high}), (x^{high}, \gamma^{low}), (x^{\max}, \gamma^{\min})\Big)$ with $\gamma^{\min} \leq \gamma^{low} \leq \gamma^{high}$ and $x^{low} \leq x^{high} \leq x^{\max}$ is called a *tube for an inhibition function*.

Definition 6.5 (An activation (inhibition) function satisfies a tube) [Breindl et al., 2011]
An activation function ν (inhibition function μ) is said to satisfy a tube T_ν (T_μ), denoted by $\nu \models T_\nu$ ($\mu \models T_\mu$) if the following inequalities hold:

$$\forall x \leq x^{low} : \nu(x) \leq \gamma^{low}(\mu(x) \geq \gamma^{high}), \tag{6.6}$$

$$\forall x \geq x^{high} : \nu(x) \geq \gamma^{high}(\mu(x) \leq \gamma^{low}), \tag{6.7}$$

$$\forall x \leq x^{\max} : \nu(x) \leq \gamma^{\max}(\mu(x) \geq \gamma^{\min}). \tag{6.8}$$

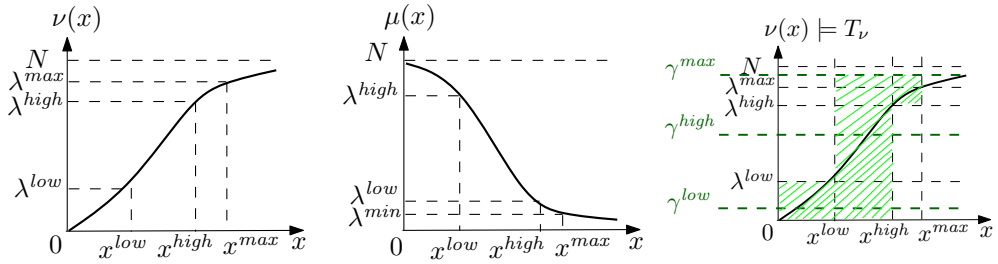


Figure 6.1: (a) An activation function. (b) An inhibition function. (c) A tube for an activation function.

Breindl et al. compute the robustness measure $\mathcal{R}$ by relaxing all monotonous functions ν or μ in a tube T_ν or T_μ as much as possible, such that the set $\mathbf{x}$ is still forward-invariant. This idea is introduced in the following proposition [Breindl et al., 2010].

Proposition 6.6 [Breindl et al., 2010]
Given a set $\mathbf{x} = \mathcal{I}^{l_1}_{x_1} \times \cdots \times \mathcal{I}^{l_n}_{x_n}$ with $l_i \in \{low, high\}$ and given tubes $T^{i,k}$ which satisfy the conditions

$$\forall i \in \{1, \ldots, n\} : -k_i \cdot \underline{x}_i + \underline{\gamma}_{i,1} \cdot \cdots \cdot \underline{\gamma}_{i,q_i} \geq 0 \tag{6.9}$$

where $\underline{x}_i = \min_{x_i \in \mathcal{I}^{l_i}_{x_i}} x_i$ and, with x_j denoting the argument of $\varphi_{i,k}$,

$$\underline{\gamma}_{i,k} = \begin{cases} 0 & \text{if } \varphi_{i,k} = \nu_{i,k} \wedge 0 \in \mathcal{I}^{l_j}_{x_j} \\ \min\{\gamma : (x,\gamma) \in T^{i,k} \wedge x \in \mathcal{I}^{l_j}_{x_j}\} & \text{otherwise} \end{cases}$$

and

$$\forall i \in \{1,\ldots,n\} : -k_i \cdot \bar{x}_i + \bar{\gamma}_{i,1} \cdot \cdots \cdot \bar{\gamma}_{i,q_i} \leq 0 \tag{6.10}$$

where $\bar{x}_i = \max_{\mathcal{I}^{l_i}_{x_i}} x_i$ and, with x_j denoting the argument of $\varphi_{i,k}$,

$$\bar{\gamma}_{i,k} = \max\{\gamma : (x,\gamma) \in T^{i,k} \wedge x_j \in \mathcal{I}^{l_j}_{x_j}\}.$$

If for all $i,k : \varphi_{i,k} \models T^{i,k}$, then $\mathbf{x}$ is a forward-invariant set for system (6.1).

Proposition 6.6 is a sufficient but not necessary condition for $\mathbf{x}^z$ being forward-invariant sets, as proved in [Breindl et al., 2010].

For each tube $T^{i,k}$, the goal is to find a function $\tilde{\varphi}_{i,k}$ best centred to the tube $T^{i,k}$. This means that $\tilde{\varphi}_{i,k}$ needs more perturbation than any other function $\varphi_{i,k} \models T^{i,k}$ to violate one of the constraints from (6.6) to (6.8). In order to achieve a violation of the tube constraints, the minimal perturbation of the best centred function $\tilde{\varphi}_{i,k}$ is reformulated as an optimisation problem [Breindl et al., 2010]:

$$\mathcal{R}^{\max}(T^{i,k}) = \max_{\varphi_{i,k} \models T^{i,k}} \mathcal{R}^{\min}(\varphi_{i,k}, T^{i,k}), \tag{6.11}$$

with

$$\mathcal{R}^{\min}(\varphi_{i,k}, T^{i,k}) = \min_{\varphi_{i,k} \not\models T^{i,k}} \mathcal{P}(\varphi_{i,k} - \varphi^p_{i,k}). \tag{6.12}$$

The robustness measure is obtained by maximising the minimal $\mathcal{P}(\varphi_{i,k}, T^{i,k})$ over all tubes $T^{i,k}$ that satisfy Proposition 6.6.

Definition 6.7 (Robustness measure) [Breindl et al., 2011]
Given a system (6.1) with unspecified activation and inhibition functions. The robustness measure $\mathcal{R}$ for the system (6.1) is defined as

$$\mathcal{R} = \max_{T^{i,k}} \min_{i,k} \mathcal{R}^{\max}(T^{i,k}) \tag{6.13}$$

such that for all $\mathbf{x}^z$: (6.9) and (6.10) hold.

Proposition 6.6 gives only a sufficient condition for $\mathbf{x}^z$ to be forward-invariant. Thus, $\mathcal{R}$ is a lower bound of the maximally obtainable robustness values.

Next an analytical solution for Equation 6.11 is given.

Proposition 6.8 [Breindl et al., 2010]
Given a tube T, the maximal value $\mathcal{R}^{\max}(T)$ is given by

$$\mathcal{R}(T_\nu) = \frac{\gamma^{low} \cdot (\gamma^{\max} - \gamma^{high})}{\gamma^{low} + (\gamma^{\max} - \gamma^{high})} \cdot (x^{high} - x^{low}), \tag{6.14}$$

if T is a tube for activation functions, and

$$\mathcal{R}(T_\mu) = \frac{(N - \gamma^{high}) \cdot (\gamma^{low} - \gamma^{\min})}{(N - \gamma^{high}) + (\gamma^{low} - \gamma^{\min})} \cdot (x^{high} - x^{low}), \tag{6.15}$$

if T is a tube for inhibition functions.

With the analytic solution for Equation 6.11 in Proposition (6.8), the robustness measure in (6.13) can be transformed into a convex optimisation problem if " $\circ$ " takes only multiplication " . ".

If the optimisation problem in (6.13) is *feasible*, then there exists a realization for the IG with monotonous functions to regenerate the required forward-invariant sets. In other words, for an IG given by system (6.1), if a value for $\mathcal{R}$ from the solution of problem (6.13) can be assigned, then this IG is able to regenerate the required multistability.

6.2 Discrete modelling methods

The specific multistability problem from Section 6.1.1 is now analysed by our discrete methods. The two specifications with the low and high concentration levels from the continuous method can be naturally discretized into Boolean states. If l is low or $high$, x^l is assigned 0 or 1 as Boolean values. The three stable states, a progenitor factor *A*, a differentiated cell type *B* and a competing differential cell type *C*, i.e., $x^{\mathrm{A}}, x^{\mathrm{B}}, x^{\mathrm{C}}$ are discretized into x^a, x^b, x^c. The two hyperrectangular sets (S1) and (S2) can be transformed as follows:

(S1) and the corresponding Boolean states:

$$\begin{aligned} x^a &= (high, low, low) \Rightarrow (1,0,0), \\ x^b &= (low, high, low) \Rightarrow (0,1,0), \\ x^c &= (low, low, high) \Rightarrow (0,0,1). \end{aligned} \tag{6.16}$$

(S2) and the corresponding Boolean states:

$$\begin{aligned} x^a &= (\ low,\ low,\ low) \Rightarrow (0,0,0), \\ x^b &= (high, high, low) \Rightarrow (1,1,0), \\ x^c &= (high, low, high) \Rightarrow (1,0,1). \end{aligned} \tag{6.17}$$

The discrete modelling method will be applied to obtain those functional IGs which can regenerate the two hypotheses (S1) and (S2). In order to compare the results from the continuous and the discrete method, the initial setup of the continuous method is transformed into an equivalent discrete version (Section 6.2.1). The two discrete modelling workflows from Chapter 4.3 are then applied in Section 6.2.3 resp. 6.2.4 to obtain all functional IGs for the multistabilities (S1) and (S2) .

6.2.1 Initial setup

The first initial setup is that every IG shall be at least *weakly connected*. A *weakly connected digraph* is a digraph that, after removing the directions of the edges, is connected as an undirected graph.

The second initial setup is the *monotonicity* of the activating and inhibiting functions in Definition 6.1. In the discrete modelling framework, activation and inhibition are represented through resources and the logical parameter functions K.

i) $\nu(\mu)$ *is continuously differentiable:* Continuous variables are discretized into integers representing intervals. The logical parameters for a component result from discretizing sigmoid functions into step functions [Thomas and D'Ari, 1990]. Although a step function can be approximated with arbitrary precision by continuously differentiable functions, the condition of continuous differentiability has no counterpart in the discrete setting.

ii) $\nu(0) = 0$ *and* $\nu(\infty) \to N$ *(*$\mu(0) = N$ *and* $\mu(\infty) \to 0$*):* ν and μ are the regulatory functions among TRs. If one TR is activated by another, it stays on the low level if the other TR is at zero, and goes to the maximal possible level if the other TR is at its highest level. If one TR is inhibited by another, the regulated TR goes to 0 as the regulating TR goes to its highest value, and vice versa. In the discrete setting, an influence from one component to another is reflected by the logical parameter functions. For example, an interaction (u, v) is observable, if there exists a resource ω of v which enables $K(v, \omega) < K(v, \omega \cup \{u\})$. Therefore, requirement ii) is replaced with the observability condition that there exists $\omega \subseteq \mathrm{Pre}(v)\backslash\{u\}$, such that, $K(v, \omega) < K(v, \omega \cup \{u\})$, which means that (u, v) is observable.

iii) $\nu(\mu)$ *is monotonously increasing (decreasing):* let (u, v) be an interaction in a discrete model. This monotonicity can be expressed as follows: for all resources $\omega\backslash\{u\}$, there is $K(v, \omega) \leq K(v, \omega \cup \{u\})$, *i.e.*, (u, v) should satisfy the Snoussi-condition. Therefore, requirement iii) is captured by the Snoussi-condition.

iv) *In Equation 6.2,* $f_i(x)$ *there are only multiplications of activation and inhibition functions:* this related to the question of how to deal with the combined effect of all regulators on a TR. Allowing only multiplication of regulatory functions makes the optimisation problem for the robustness measure easily solvable. The multiplication of two activation functions is still an activation function, and that of two inhibition functions is still an inhibition function. However, when multiplying an activation and an inhibition function, it is hard to decide which one is more dominant without knowing the exact shapes of both functions. In discrete modelling, the combined effect from several regulating components is given by a logical parameter. The effect of these combined functions can be illustrated as: let u be a component in the system, in terms of $K(u, \cdot)$, for all resource $\zeta \subseteq \omega \subseteq \mathrm{Pre}(u)$, there is always $K(u, \zeta) \leq K(u, \omega)$.

This means that every component satisfies the Snoussi-condition. Part of the K's correspond to the continuous parameters which can be only combined by multiplication. One can find another part of K's have no correspondence in the continuous method [Breindl et al., 2011], and still satisfy the Snoussi-condition. It is expected that more structures will be found using discrete methods.

As a summary for the second setup, all models shall be compatible, *i.e.*, satisfy the observability and the Snoussi-condition.

The third initial setup in Section 6.1.1 is that *a TR can only activate or inhibit or give no influence to a TR*. This means that no multiple edges are allowed, which is the same as in our discrete modelling framework.

6.2.2 Different constraints on the desired stable states

Some ASTGs may have more stable states than those specified by (S1) or (S2). In the discrete setting, stable states may also be isolated, *i.e.*,, they cannot be reached from any other state.

From the specification in [Breindl et al., 2011], it is not clear whether the specified states (S1) or (S2) can be isolated or not, or whether the existence of other stable states is excluded.

We may consider the following scenarios:

(a) There can be other stable states besides the specified ones; isolated states are allowed.

(b) There are only the specified stable states; isolated states are allowed.

(c) There are only the specified stable states; no isolated state is allowed.

(d) There can be other stable states besides the specified ones; no isolated state is allowed.

Constraint (a) imposes no restriction, thus more ASTGs are included in the enumeration. The ASTGs obtained under the other constraints form a subset of those from (a). In their continuous setting, [Breindl et al., 2011] do not address the question of isolated states and other possible attractors. For this reason, constraint (a) has been chosen for the discrete modelling method. The results are shown in Section 6.3.2 and 6.3.2.4.

6.2.3 Forward modelling workflow: from IGs to ASTGs

The forward modelling workflow starts from the enumeration of all possible IGs to find those that are functional for the target stable states (S1) or (S2). The process is shown below.

1. *IG-enumeration*. Between any two components (including self-loops), there are three possible cases: (*activation*, *inhibition* and *no influence*). For the interactions in a 3-component system, there are 9 possible cases and 3^9 different IGs.

2. *K-enumeration*. For every component $u \in V$, there are $2^{|\mathrm{Pre}(u)|}$ resources, where $|\mathrm{Pre}(u)|$ is the cardinality of u's predecessor set. For each resource $\omega \subseteq \mathrm{Pre}(u)$, $K(u,\omega) \in \{0,\ldots,\max_u\}$. Thus there are $(\max_u + 1)^{2^{|\mathrm{Pre}(u)|}}$ combinations in K-enumeration. A parameter K is compatible with an IG, if the logical parameters $K(u,\cdot)$ from each component u are compatible with the incoming interactions of u in the IG. Therefore, the parameter space of K can be obtained by finding the compatible $K(u,\cdot)$'s from each component.

   The table below shows different cardinality up to 3 of the predecessor set of a Boolean component u and the number of compatible logical parameters $K(u,\cdot)$, which is obtained by checking the compatibility of all possible logical parameters.

# predecessor	# resources	# $K(v,\cdot)$	# compatible $K(v,\cdot)$
$\lvert\mathrm{Pre}(v)\rvert$	$2^{\lvert\mathrm{Pre}(v)\rvert}$	$2^{2^{\lvert\mathrm{Pre}(v)\rvert}}$	
1	2	4	1
2	4	16	2
3	8	256	9

3. *ASTG generation and checking stable states.* The enumerated IGs and their compatible K's form the set of all compatible models $\mathcal{M}$. Those models whose ASTGs carry (S1) or (S2) are called functional.

4. *Extracting functional IGs.* The IGs of all functional models from both (S1) and (S2) are extracted for further analysis of structural properties.

Step 2 for the K enumeration is the most time and memory consuming. A Boolean component u with k predecessors has 2^k resources. The logical parameter under a resource $K(u,\omega)$ has two possible values $\{0,1\}$. The number of all logical parameters of u under all resources,

$K(u,\cdot)$, is $2^{2^{|\mathrm{Pre}|}} = 2^{2^k}$, i.e., the number is double exponentially increasing with the size of the predecessor set.

6.2.4 Reverse engineering workflow: from ASTGs to IGs

The reverse engineering workflow starts from enumerating all graphs based on the state space that have the required stable states from (S1) or (S2). The process is shown below.

1. *Enumerate all graphs $\mathcal{G}_X$ based on $X = \{0,1\}^3$.* For all $x \in X$, $u \in V$, there are 2 possibilities for x_u: it either keeps its value or is negated. For the required conditions (S1) or (S2), the transition function of the three stable states is fixed to $(0,0,0)$. The number of the other possible transitions is: $2^{(8*3-3*3)} = 2^{15}$.

2. *ASTG-checking and model inference for all $\mathcal{G}_X$.* The *generalised algorithm Observability-Snoussi-Model* is applied on all graphs $\mathcal{G}_X$. All possible functional models are obtained.

3. *Extracting functional IGs.* The IGs from all functional models for both (S1) and (S2) are extracted for further analysis of structural properties.

6.3 Results and analysis

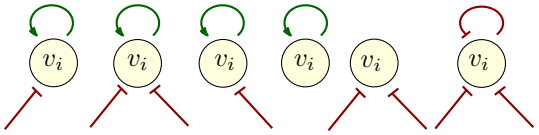
This section contains the results of the continuous and the discrete method together with a comparison between them. On the functional IGs resulting from the discrete method, the realization measure and logical analysis are also applied.

6.3.1 Results from the continuous method: robustness measure

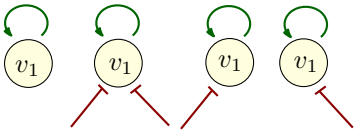
A TR can activate, inhibit or have no effect on another TR, thus there are three possibilities ($a_{ij} \in \{+1,-1,0\}$) for each entry of the interaction matrix A. [Breindl et al., 2011] studied only those interacting structures which are at least weakly connected. The number of all weakly connected interaction matrices A of three TRs is 19008. For each interaction matrix A, a robustness measure $\mathcal{R}$ is calculated.

Each interaction matrix denotes a topology. A topology is considered to be able to reproduce the forward-invariant set in spite of perturbations, if there exists a feasible solution for $\mathcal{R}$ the optimisation problem (6.13).

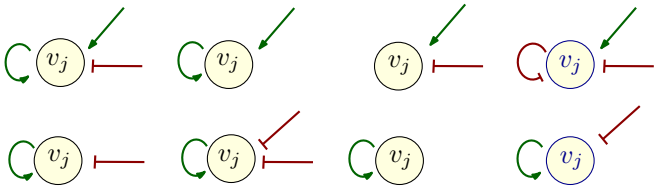
[Breindl et al., 2011] found 206 GRNs that can reproduce (S1) and 242 GRNs that can reproduce (S2). For these functional GRNs, the complete enumeration of all "*building blocks*" are shown in Figure 6.2. Here a *building block* is a TR with only incoming links. All networks that are assembled from these building blocks are able to generate the specified (S1) or (S2), as shown in Figure 6.2.



(a) For (S1), $i \in \{1, 2, 3\}$.



(b) For (S2), x_1.



(c) For (S2), $i \in \{2, 3\}$.

Figure 6.2: [Breindl et al., 2011] Building blocks from the robustness measure method in [Breindl et al., 2011]. (a) shows the results from (S1), the two interactions entering x_i from the bottom are from the other two TRs. (b) and (c) show results from (S2). In (b), those incoming arrows from bottom left and right denote regulations from x_2 and x_3 respectively. In (c), those incoming arrows from the top right and right side are regulations from x_1, and from TRs other than x_i, respectively.

6.3.2 Results from discrete methods

The functional models for the hypotheses (S1) and (S2) found by the forward modelling workflow are the same as those from the reverse engineering workflow. The results for (S1) and (S2) are summarised below.

6.3.2.1 Hypothesis (S1)

The forward modelling workflow obtained from 3^9 IG candidates 2197 functional models, whose ASTGs satisfy (S1) (with stable states $(0, 0, 1)$, $(0, 1, 0)$, $(1, 0, 0)$). The reverse engineering workflow started from enumerating all the 2^{15} graphs based on X with fixed (S1), and inferred the same set of 2197 functional models. These 2197 functional models contain 512 different IGs and 1000 different K's.

There are 512 functional IGs for (S1). Besides 10 non-weakly connected IGs, there are 502 weakly connected IGs with 2187 models. The 1000 K's are comprised of 10 different logical parameters for each component K_{v_1}, K_{v_2} and K_{v_3}, respectively.

The 512 IGs have different numbers of functional models. There is one IG which has 216 functional models. Some IG can have 36 functional models and overall there are 21 such IGs. A complete statistics is given below.

# IGs	1	21	147	343
# funct. models	216	36	6	1

6.3.2.2 Hypothesis (S2)

The forward modelling workflow obtained from 3^9 IG candidates 2197 functional models, whose ASTGs satisfy (S2) (with stable states $(0,0,0)$, $(1,0,1)$, $(1,1,0)$). The reverse engineering workflow found the same set of 2197 functional models from enumerating all 2^{15} graphs based on X with fixed (S2). These 2197 functional models also contain 512 different IGs and 1000 different K's.

Like for (S1), there are 512 functional IGs for (S2). Besides 10 non-weakly connected IGs, there are 502 weakly connected IGs with 2187 models. The 1000 K's are composed by 10 different logical parameters for K_{v_1}, K_{v_2} and K_{v_3}, respectively.

A complete statistics for (S2) is given below.

#IGs	1	21	147	343
#funct. models	216	36	6	1

Note that although the numbers for (S2) are the same as for (S1), the set of functional models for (S1) and (S2) are different.

6.3.2.3 Building blocks

The components x_1, x_2 and x_3 in the continuous method are v_1, v_2 and v_3 in the discrete approach. An IG can be decomposed into three parts by looking at the incoming interactions of each component. A building block for one component includes the incoming interactions contained in the functional IGs. All functional IGs can be decomposed into such building blocks, and they can also be recomposed from these.

The 512 IGs functional for (S1) are a full combination of the 8 building blocks for every component v_1, v_2 and v_3. Figure 6.3 shows the building blocks for (S1). Similarly, the 512 functional IGs for (S2) contain also 8 building blocks from every component v_1, v_2 and v_3. Figure 6.4 shows the building blocks for (S2).

For each IG assembled from these building blocks, there exists at least one model which can realise (S1) (or (S2)). By combining three building blocks out of these 8 one can obtain all 512 IGs.

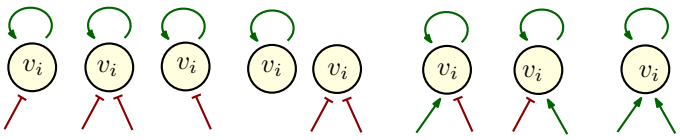
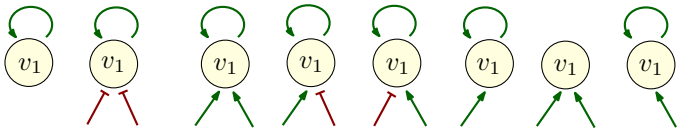
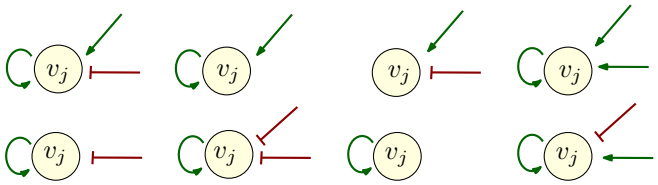


Figure 6.3: The building blocks of (S1) from discrete modelling method. The arrows going to v_i from below left and right are from v_j and v_k, respectively, where $j \neq i$ and $k = \{1,2,3\}\backslash\{i,j\}$.



(a) For (S2), v_1.



(b) For (S2), $i \in \{2, 3\}$.

Figure 6.4: The building blocks of (S2) from discrete modelling method. (a) Building blocks of component v_1, the arrows going to v_1 from below left and right are from v_2 and v_3, respectively. (b) Building blocks of components v_j with $j \in \{2, 3\}$, the links going to v_j from top right is from v_1, and from right side is v_k, $k = \{2, 3\}\backslash\{j\}$.

Proposition 6.9 1. All functional IGs for (S1) can be obtained as combinations from these building blocks. The IGs that are composed by choosing any of the 8 building blocks of each of the three components in Figure 6.3 are functional for (S1).

2. All functional IGs for (S2) can be obtained as combinations from these building blocks. The IGs that are composed by choosing any of the 8 building blocks of each of the components in Figure 6.4 are functional for (S2).

Proof. For (S1), the following cases hold.

1. The forward modelling workflow is the process of searching for all compatible models for all possible IGs of 3 components and extracting all functional IGs. If an IG has at least one compatible model that can regenerate (S1), then this IG is included in the result. It follows that the 512 IGs cover all possible functional IGs of 3 components.

2. The 8 building blocks from each of the three components can be freely combined to recover the full set of $8^3 = 512$ IGs.

Analogously, the above cases also hold for (S2). Proposition 6.9 is proved. □

6.3.2.4 Additional constraints on stable states

While the numbers of functional models, IGs and building blocks are the same for (S1) and (S2), the corresponding sets are different. The functional models under the constraints (b), (c) and (d) in Section 6.2.2 are subsets of those under the constraint (a). The 8 building blocks for each component appear also for the other constraints. For both (S1) and (S2), the 8 building blocks of the three components can be combined freely to recover all 512 functional IGs under constraint (a. This does not hold for the other constraints. As the most general case, the results for constraint (a) are chosen for further analysis.

The results based on all four constraints are listed in Table 6.1.

constraint	# models	# IGs	# building blocks		
			v_1	v_2	v_3
(a)	2197	512	8	8	8
(b)	441	176	8	8	8
(c)	336	154	8	8	8
(d)	784	233	8	8	8

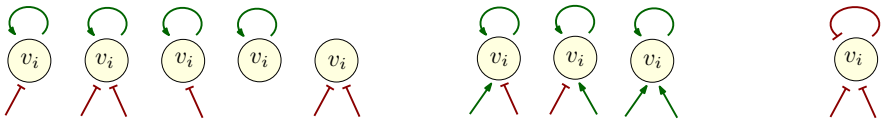
Table 6.1: Number of functional models, IGs and building blocks for condition (S1) or (S2), depending on the four constraints (a) - (d).

6.3.3 Comparison of the results

[Breindl et al., 2011] used the robustness measure to determine all functional GRNs of three TRs. A comparison with the results from the discrete modelling method is given in Section 6.3.3.1. Moreover, some building blocks for (S1) are the same as for (S2). They are compared in Section 6.3.3.2.

6.3.3.1 Discrete vs continuous methods

For (S1), the building blocks from the discrete and the continuous method are shown in Figure 6.5.



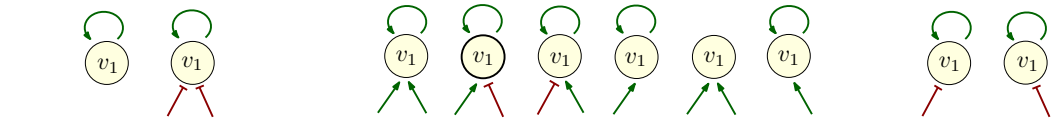
(a) discrete and continuous (b) discrete (c) continuous

Figure 6.5: (S1), building blocks for v_i, $i \in \{1, 2, 3\}$ by the discrete and the continuous method. (a) Both discrete and continuous. (b) Only discrete. (c) Only continuous. Links going to v_i from bottom left and right represent interactions from nodes v_j, $j \neq i$, and v_k, $i \neq k \neq j$.

The similarities and differences between the two approaches are further analysed below.

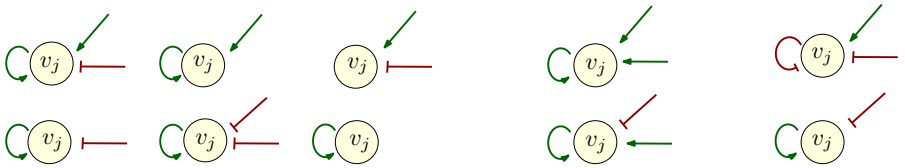
1. In the results obtained by the discrete method:
   (a) The *self-loop* of a component v_i, $i \in \{1, 2, 3\}$ is *always positive*.
   (b) The *interactions* between two different components *can be positive or negative*.
2. In the results obtained by the continuous method:
   (a) The self-loop of a component v_i, $i \in \{1, 2, 3\}$ *can be positive or negative*.
   (b) The *interactions* between two different components are *always negative*.

Similarly for (S2), the building blocks from the discrete and continuous method are shown in Figure 6.6 and 6.7.



(a) discrete and continuous (b) discrete (c) continuous

Figure 6.6: (S2), building blocks for v_1 by the discrete and the continuous method. (a) Both discrete and continuous. (b) Only discrete. (c) Only continuous. Links going to v_1 from below left and right represent interactions from v_2 and v_3.



(a) discrete and continuous (b) discrete (c) continuous

Figure 6.7: (S2), building blocks for v_j, $j \in \{2,3\}$ by the discrete and the continuous method. (a) Both discrete and continuous. (b) Only discrete. (c) Only continuous. Links entering v_j from top right is from v_1, and from right side is v_k, $k = \{2,3\}\backslash\{j\}$.

The differences between the building blocks for v_1 are as follows:

1. In the results obtained by discrete method:
   (a) If v_1 has no self-activation, it receives two activations from v_2 and v_3.
   (b) The *interactions* (v_2, v_1) and (v_3, v_1) *can be positive or negative*.
2. In the results obtained by the continuous method:
   (a) v_1 always has a self-activation.
   (b) The *interactions* between two different components are *always negative*.

The difference between the building blocks of v_2 and v_3 are the following:

1. In the results obtained by discrete method:
   (a) The *self-loop* of a component v_j, $i \in \{2,3\}$ is *always positive*.
   (b) The *interactions* between v_2 and v_3 *can be positive or negative*.
2. In the results obtained by the continuous method:
   (a) The *self-loop* a component v_j, $i \in \{2,3\}$ can be *positive or negative*.
   (b) The *interactions* between v_2 and v_3 are *always negative*.

6.3.3.2 Discrete methods on (S1) and (S2)

The stable states in (S1) 100, 001 and 010 are in a some way symmetric to the stable states 000, 101 and 110 in (S2). They have the same values in components x_{v_2} and x_{v_3}, but opposite value in x_{v_1}. The same number of functional IGs are obtained under both hypotheses $(S1)$ and $(S2)$. Moreover, the functional IGs for (S1) and (S2) have exactly opposite signs in the interactions

(v_1, v_2), (v_2, v_1), (v_1, v_3) and (v_3, v_1). In other words, if the signs of these four interactions in each functional IG for (S1) are changed from positive to negative, and from negative to positive, we get exactly the set of functional IGs for (S2). The reason lies in the symmetry of the two hypotheses.

All 512 functional IGs for $(S1)$ are summed up together with every present interaction. The *occurrence rate* of an interaction is defined as how many times it shows up among all 512 IGs, as shown in Figure 6.8a. The same process is made for the functional IGs of $(S2)$, as shown in Figure 6.8b.

(a) (S1) (b) (S2)

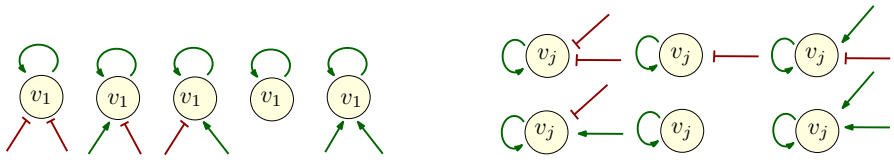
Figure 6.8: Sum of all 512 functional IGs for: (a) $(S1)$ and (b) $(S2)$. The weight of an interaction: occurrence rate. Arrows: activations. Blunt links: inhibitions. The thickness of lines denote the occurrence rates.

From Figure 6.8, the following can be observed.

- For (S1):
  1. The self-loop of each component is always positive. The occurrence rate is 0.875.
  2. For the interactions between different components, more inhibitions appear than activations. The occurrence rate of each inhibition is 0.5 and of each activation is 0.25.
- For (S2):
  1. The self-loop of each component is always positive. The occurrence rate is 0.875.
  2. For the interactions between v_1 and v_2, and between v_1 and v_3, more activations appear than inhibitions. The occurrence rate of each activation is 0.5 and of each inhibition is 0.25.
  3. For the interactions between v_2 and v_3, more inhibitions appear than activations. The occurrence rate of each inhibition is 0.5 and of each activation is 0.25.

  In summary, for both (S1) and (S2), only self-activations are observed. The occurrence rate of inhibitions in (S2) between v_2 and v_3 is higher than in (S1). The occurrence rate of activations in (S2) between v_1 and v_j with $j \in \{2, 3\}$ is also higher than in (S1).

The two sets of functional IGs for (S1) and (S2) have 180 IGs in common. Moreover, these 180 IGs are a full combination of 5 building blocks for v_1, and 6 building blocks for both v_2 and v_3, as shown in Figure 6.9.



(a) v_1 (b) v_j

Figure 6.9: The building blocks of the 180 IGs functional for both (S1) and (S2). (a) From v_1, the links entering v_1 from below left and right are from v_2 and v_3. (b) v_j, the links entering v_j from top right come from v_1, and from right come from v_k, $k = \{2, 3\}\backslash\{j\}$. Arrows: activations. Blunt links: inhibitions.

The sum graph of these 180 IGs is shown in Figure 6.10.

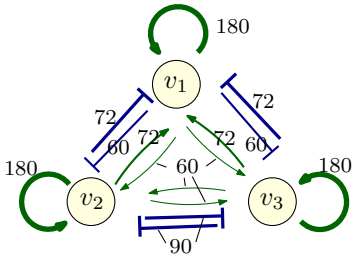


Figure 6.10: The sum of the 180 IGs. The numbers on the links denote the occurrence numbers of each interaction.

From Figure 6.9 and 6.10, the following can be observed.

1. Each component always has a self-activation.
2. v_1 gets the same amount of activations and inhibitions from the other two components.
3. v_1 has the same amount of activations and inhibitions towards the other two components.
4. There are more inhibitions than activations between v_2 and v_3.

6.3.4 Realizability

For each functional IG for (S1) and (S2), the realizability can be calculated. Because the position of (S1) and (S2) are symmetric in the state space, for each functional IG for (S1), there exists exactly one IG for (S2) such that the realizabilities of these two IGs are the same. No surprise that, the capacity measures for the two hypotheses are also the same, 0.0276.

For the functional IGs for (S1) and (S2), a classification is done based on the number of the functional and compatible parameters of each IG. Table 6.2 shows the number of the functional and compatible parameters of each functional IG for (S1) or (S2). In the third row, the number of IGs means that there are this amount of IGs which have this number of compatible and functional parameters.

# of funct_K	1	1	1	1	1	6	1	6	1	6	1	6	36	1	6	36	1	6	36	216
# of compt_K	1	2	4	8	9	9	18	18	36	36	81	81	81	162	162	162	729	729	729	729
# of IGs	1	9	27	27	9	3	54	18	81	27	27	18	3	81	54	9	27	27	9	1

Table 6.2: Number of K's for functional IGs for (S1) or (S2) (20 columns).

The numbers in Table6.2 are in common for (S1) and (S2). The reason is that for the stable states (S1) and (S2) are very symmetric and only differ in one component.

There are 180 IGs functional for both (S1) and (S2). IGs do not have the same number of functional parameters for both (S1) and (S2). In total, these 180 IGs have 1210 models for (S1) and also 1210 models for (S2). A classification is done on the number of the functional and compatible parameters of each IG, as shown in Table 6.3 for (S1) and 6.4 for (S2).

# of funct_K	1	1	1	1	6	1	6	1	6	1	6	36	1	6	36	1	6	36	216
# of compt_K	1	2	4	9	9	18	18	36	36	81	81	81	162	162	162	729	729	729	729
# of IGs	1	2	1	9	3	12	4	3	1	27	18	3	18	12	2	27	27	9	1

Table 6.3: Functional parameters for (S1) for the 180 IGs functional for both (S1) and (S2) (19 columns).

# of funct_K	36	1	6	6	1	6	1	6	36	1	6	1	6	36	1	6	36	1	6	36	216
# of compt_K	1	2	2	4	9	9	18	18	18	36	36	81	81	81	162	162	162	729	729	729	729
# of IGs	1	1	1	1	6	6	9	5	2	2	2	28	19	1	18	10	4	37	23	3	1

Table 6.4: Functional parameters for (S2) for the 180 IGs functional for both (S1) and (S2) (21 columns).

6.3.5 Logical analysis on functional models

To further analyse the functional models, the logical analysis method is applied on the functional IGs and the logical parameters.

Logical analysis on IGs. In order to find meaningful logical IG patterns, the functional IGs for the desired attractors are transformed into a sum of Boolean expressions. Here, the target dynamics includes (S1) and (S2). As introduced in Chapter 4, transforming an IG of 3 Boolean components into a Boolean expression, 18 Boolean variables are needed. The core of this logical analysis method is the Quine-McCluskey algorithm (using the software PyBoolNet [Klarner et al., 2016]) in order to compute the minimal DNF. However, 18 Boolean variables and 512 IGs is too much to get the prime implicants in a reasonable time. Alternative tools can be applied, but the number of prime implicants is still too high.

For this reason, the logical analysis method is applied on the building blocks of every component. A building block includes at most three incoming interactions. Thus 6 Boolean variables are needed for encoding. For example, the Boolean expressions of the building blocks of v_i for (S1) are shown in Figure 6.11.

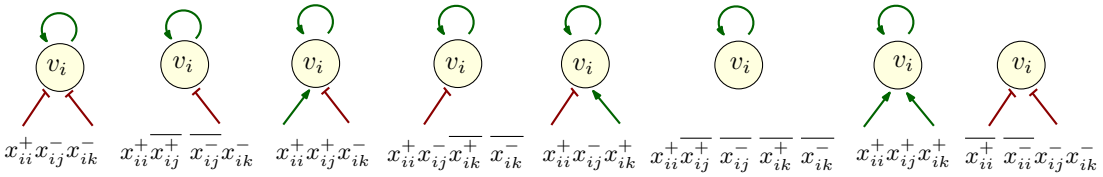


Figure 6.11: The Boolean expressions of all building blocks for (S1). Links going from bottom left and right are v_j and v_k, respectively. $i, j, k \in \{1, 2, 3\}$.

The sum of these Boolean expressions is given as input into PyBoolNet to get the minimal DNF. This minimal DNF provides a compact description of all building blocks. The minimal DNF of the 8 Boolean expressions in Figure 6.11 is a disjunction of 5 clauses. In Figure 6.12, they are transformed back to the form of graphs.

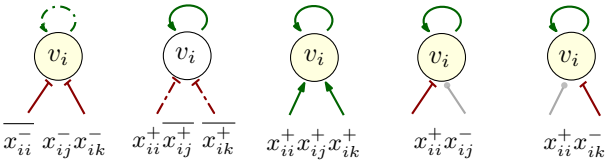
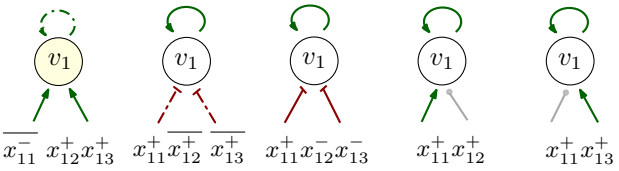


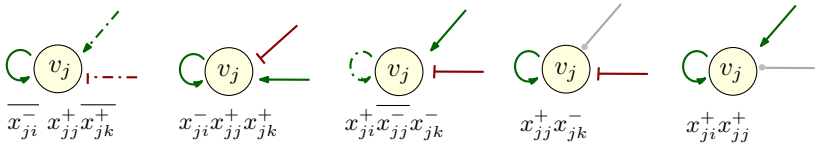
Figure 6.12: The minimal DNF of all building blocks from (S1). Arrows: activations, blunt links: inhibitions, dashed arrows: *non-inhibitions*, blunt dashed links: *non-activations*, dot-ended line: *any one of activation, inhibition and no influence*.

The full combination of these 5 logical IG patterns is exactly the same set as the 512 functional IGs for (S1). In total there are $5^3 = 125$ combinations in terms of logical expressions, which is much less than 512.

Similarly, the minimal DNF of the 8 Boolean expressions from the building blocks for (S2) is a disjunction of 5 clauses. Figure 6.13 shows them in terms of graphs.



(a) v_1



(b) v_j, $j \in \{2, 3\}$

Figure 6.13: The minimal DNF of all building blocks from (S2). Arrows: activations, blunt links: inhibitions, dashed arrows: *non-inhibitions*, blunt dashed links: *non-activations*, dot-ended line: *any one of activation, inhibition and no influence*.

The full combination of these 5 logical IG patterns again gives exactly the same set of 512 functional IGs for (S2). In total there are $5^3 = 125$ combinations in terms of logical expressions, which is much less than 512.

Logical analysis of K's: (S1) One functional IG for (S1) has 216 functional K's, which is the maximum among all 512 IGs. This IG is shown in Figure 6.14. Moreover, if we look at the logical parameters for each component, the 216 K's are a full combination of 6 different $K(v_i, \cdot)$, for each $v_i \in \{v_1, v_2, v_3\}$. To keep the problem simple, the logical analysis method is applied on the logical parameters of an individual component, $K(v_i, \cdot)$. It is enough to consider v_1 because it has similar results in the logical parameters and structure of state transitions as component v_2 and v_3.

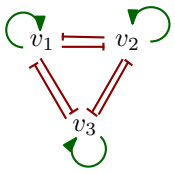


Figure 6.14: An IG which has 216 functional K's for $(S1)$.

The logical parameters $K(v_1, \cdot)$ in the case of 8 resources are transformed into a Boolean expression using 8 Boolean variables. The minimal DNF obtained from the sum of 6 different $K(v_1, \cdot)$'s contains 5 clauses. They are translated back to the form of logical parameters, as shown in Table 6.5. The results for $K(v_2, \cdot)$ and $K(v_3, \cdot)$ are not shown because they are similar to v_1.

$K(v_1,\cdot)\backslash$Res	$\emptyset$	$\{v_3\}$	$\{v_2\}$	$\{v_2,v_3\}$	$\{v_1\}$	$\{v_1,v_3\}$	$\{v_1,v_2\}$	$\{v_1,v_2,v_3\}$	f
$K^1(v_1,\cdot)$	0	0	0	0	0	0	0	1	1
$K^2(v_1,\cdot)$	0	0	0	0	0	1	1	1	1
$K^3(v_1,\cdot)$	0	0	0	1	0	0	1	1	1
$K^4(v_1,\cdot)$	0	0	0	1	0	1	1	1	1
$K^5(v_1,\cdot)$	0	0	0	1	0	1	0	1	1
$K^6(v_1,\cdot)$	0	0	0	1	1	1	1	1	1
$K^{p1}(v_1,\cdot)$	0	0	0	0	0	0	0	1	(K^1)
$K^{p2}(v_1,\cdot)$	0	0	0	–	0	1	1	1	(K^2+K^4)
$K^{p3}(v_1,\cdot)$	0	0	0	1	–	1	1	1	(K^4+K^6)
$K^{p4}(v_1,\cdot)$	0	0	0	1	0	–	1	1	(K^3+K^4)
$K^{p5}(v_1,\cdot)$	0	0	0	1	0	1	–	1	(K^4+K^5)

Table 6.5: For the IG in Figure 6.14 ((S1)), 6 $K(v_1, \cdot)$'s. The minimal DNF is shown as 5 $K^p(v_1, \cdot)$'s. "–": either 0 or 1.

The 5 clauses in the minimal DNF are just one less than the number of logical parameters. Combining two Boolean expressions by extracting the largest common factors, we can get a shorter description of 3 logical parameter patterns, as shown in Table 6.6. This description is

$$\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,(\overline{x_4}\,\overline{x_5}(\overline{x_6}\,\overline{x_7} + x_6x_7) + x_4\overline{x_5}(\overline{x_6}x_7 + x_6\overline{x_7}) + x_4x_6x_7)\,x_8.$$

$K(v_1,\cdot)\backslash$Res	$\emptyset$	$\{v_3\}$	$\{v_2\}$	$\{v_2,v_3\}$	$\{v_1\}$	$\{v_1,v_3\}$	$\{v_1,v_2\}$	$\{v_1,v_2,v_3\}$	f
Boolean variable	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
$K^1(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	$\overline{x_4}$	$\overline{x_5}$	$\overline{x_6}$	$\overline{x_7}$	x_8	1
$K^2(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	$\overline{x_4}$	$\overline{x_5}$	x_6	x_7	x_8	1
$K^3(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	x_4	$\overline{x_5}$	$\overline{x_6}$	x_7	x_8	1
$K^4(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	x_4	$\overline{x_5}$	x_6	x_7	x_8	1
$K^5(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	x_4	$\overline{x_5}$	x_6	$\overline{x_7}$	x_8	1
$K^6(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	x_4	x_5	x_6	x_7	x_8	1
$K^{p1}(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	$\overline{x_4}$	$\overline{x_5}$	$(\overline{x_6}\,\overline{x_7} + x_6x_7)$		x_8	(K^1+K^2)
$K^{p2}(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	x_4	$\overline{x_5}$	$(\overline{x_6}x_7 + x_6\overline{x_7})$		x_8	(K^3+K^5)
$K^{p3}(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	x_4		x_6	x_7	x_8	(K^4+K^6)

Table 6.6: For the IG in Figure 6.14 ((S1)), a short description of 6 $K(v_1, \cdot)$'s in terms of Boolean variables.

All 6 $K(v_1,\cdot)$ differ from each other in 4 resources. This can be directly related to the state transitions in direction of v_1 in the state space. Figure 6.15 shows the state transitions in direction of v_1 in the state space.

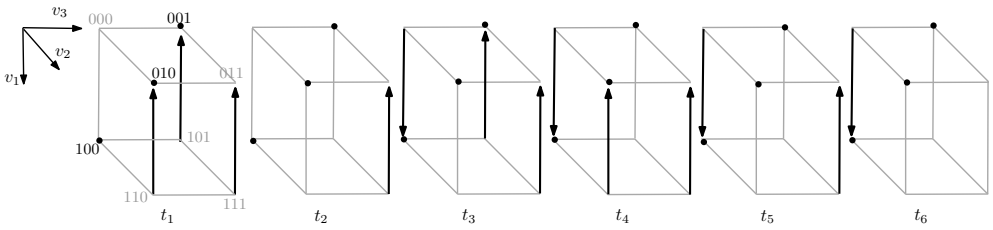


Figure 6.15: The 6 state transitions in direction of v_1, from all functional ASTGs of IG in Figure 6.14. Three black dots: (S1).

The incoming interactions of v_1 in Figure 6.14 include an activation $(v1, v1)$, and two inhibitions $(v2, v1)$ and $(v3, v1)$. The state transitions in Figure 6.15 can be tracked back to $K(v_1, \mathrm{Res}_{v_1}(\text{state}))$. For example, in state 101, $\mathrm{Res}_{v_1}(101) = \{v_2, v_3\}$, $K^i(v_1, \{v_2, v_3\}) = 0$ for $i \in \{1, 3\}$, $K^i(v_1, \{v_1, v_1\}) = 1$ for $i \in \{2, 3, 4, 6\}$. In state 000, $\mathrm{Res}_{v_1}(000) = \{v_2, v_3\}$, $K^i(v_1, \{v_2, v_3\}) = 0$ for $i \in \{1, 2\}$, $K^i(v_1, \{v_2, v_3\}) = 1$ for $i \in \{3, 4, 5, 6\}$. The structures in Figure 6.15 correspond one by one to the $K(v_1,\cdot)$'s in Table 6.5.

Logical analysis on K's: (S2) Similarly, there is an IG functional for $(S2)$ which has 216 functional K's. These 216 K's are a full combination of 6 $K(v_i,\cdot)$'s, for each $v_i \in \{v_1, v_2, v_3\}$. This IG is shown in Figure 6.16 and the minimal DNF of its functional $K(v_i,\cdot)$'s for each component is analysed. Because v_2 is similar to v_3 both in the logical parameters and the structure of state transitions, only the results of v_1 and v_2 are presented in Table 6.7 and 6.9.

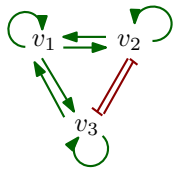


Figure 6.16: An IG which has 216 functional K's for $(S2)$.

$K(v_1,\cdot)\backslash$Res	$\emptyset$	$\{v_3\}$	$\{v_2\}$	$\{v_2,v_3\}$	$\{v_1\}$	$\{v_1,v_3\}$	$\{v_1,v_2\}$	$\{v_1,v_2,v_3\}$	f
$K^1(v_1,\cdot)$	0	0	0	0	0	1	1	1	1
$K^2(v_1,\cdot)$	0	0	0	1	0	1	1	1	1
$K^3(v_1,\cdot)$	0	0	0	1	1	1	1	1	1
$K^4(v_1,\cdot)$	0	0	1	1	0	1	1	1	1
$K^5(v_1,\cdot)$	0	1	0	1	0	1	1	1	1
$K^6(v_1,\cdot)$	0	1	1	1	1	1	1	1	1
$K^{p1}(v_1,\cdot)$	0	1	1	1	1	1	1	1	1
$K^{p2}(v_1,\cdot)$	0	–	0	1	0	1	1	1	1
$K^{p3}(v_1,\cdot)$	0	0	–	0	0	1	1	1	1
$K^{p4}(v_1,\cdot)$	0	0	0	–	0	1	1	1	1
$K^{p5}(v_1,\cdot)$	0	0	0	1	–	1	1	1	1

Table 6.7: For the IG in Figure 6.16, 6 functional $K(v_1,\cdot)$'s for (S2), 5 $K(v_1,\cdot)^p$'s from the minimal DNF. "–": either 0 or 1.

All 6 $K(v_1,\cdot)$ differ from each other in 4 resources. Again, this can be directly related to the state transitions in direction of v_1 in the state space. Figure 6.17 shows the state transitions in direction of v_1 according to IG (Figure 6.16) and the 6 different $K(v_1,\cdot)$.

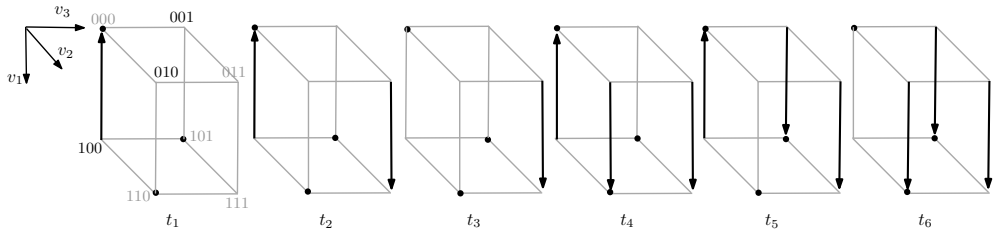


Figure 6.17: The 6 state transitions in direction of v_1, from all functional ASTGs of IG in Figure 6.16. Three black dots: stable states of (S2).

There are 5 clauses obtained in the minimal DNF by using direct encoding and PyBoolNet on 6 $K(v_1,\cdot)$. The parameters in Table 6.7 are very similar with the $K(v_1,\cdot)$'s for (S1). Combining two Boolean expression by extracting the largest common factors, we can get a short description of these logical parameters, as shown in Table 6.8.

$K(v_1,\cdot)\backslash$Res	$\emptyset$	$\{v_3\}$	$\{v_2\}$	$\{v_2,v_3\}$	$\{v_1\}$	$\{v_1,v_3\}$	$\{v_1,v_2\}$	$\{v_1,v_2,v_3\}$	f
Boolean variable	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
$K^1(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	$\overline{x_4}$	$\overline{x_5}$	x_6	x_7	x_8	1
$K^2(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	x_4	$\overline{x_5}$	x_6	x_7	x_8	1
$K^3(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	x_4	x_5	x_6	x_7	x_8	1
$K^4(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	x_3	x_4	$\overline{x_5}$	x_6	x_7	x_8	1
$K^5(v_1,\cdot)$	$\overline{x_1}$	x_2	$\overline{x_3}$	x_4	$\overline{x_5}$	x_6	x_7	x_8	1
$K^6(v_1,\cdot)$	$\overline{x_1}$	x_2	x_3	x_4	x_5	x_6	x_7	x_8	1
$K^{p1}(v_1,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$		x_5	x_6	x_7	x_8	(K^1+K^2)
$K^{p2}(v_1,\cdot)$	$\overline{x_1}$	$(\overline{x_2}\overline{x_3}+x_2x_3)$		x_4	$\overline{x_5}$	x_6	x_7	x_8	(K^3+K^6)
$K^{p3}(v_1,\cdot)$	$\overline{x_1}$	$((\overline{x_2}x_3+x_2\overline{x_3})$		x_4	$\overline{x_5}$	x_6	x_7	x_8	(K^4+K^5)

Table 6.8: For the IG in Figure 6.16, a short description of 6 $K(v_1,\cdot)$'s in terms of Boolean variables, for (S2).

$K(v_2,\cdot)\backslash$Res	$\emptyset$	$\{v_3\}$	$\{v_2\}$	$\{v_2,v_3\}$	$\{v_1\}$	$\{v_1,v_3\}$	$\{v_1,v_2\}$	$\{v_1,v_2,v_3\}$	f
$K^1(v_2,\cdot)$	0	0	0	0	0	0	0	1	1
$K^2(v_2,\cdot)$	0	0	0	0	0	1	1	1	1
$K^3(v_2,\cdot)$	0	0	0	1	0	0	1	1	1
$K^4(v_2,\cdot)$	0	0	0	1	0	1	1	1	1
$K^5(v_2,\cdot)$	0	0	0	1	0	1	0	1	1
$K^6(v_2,\cdot)$	0	0	1	1	0	1	1	1	1
$K^{p1}(v_2,\cdot)$	0	0	0	0	0	0	0	1	1
$K^{p2}(v_2,\cdot)$	0	0	–	1	0	1	1	1	1
$K^{p3}(v_2,\cdot)$	0	0	0	–	0	1	1	1	1
$K^{p4}(v_2,\cdot)$	0	0	0	1	0	–	1	1	1
$K^{p5}(v_2,\cdot)$	0	0	0	1	0	1	–	1	1

Table 6.9: For the IG in Figure 6.16, 6 functional $K(v_2,\cdot)$'s. 5 $K^p(v_2,\cdot)$'s from the minimal DNF, for (S2). "$-$": either 0 or 1.

Similarly, the state transitions in direction of v_2 according to IG (Figure 6.16) and the 6 different $K(v_2,\cdot)$ are shown in Figure 6.18

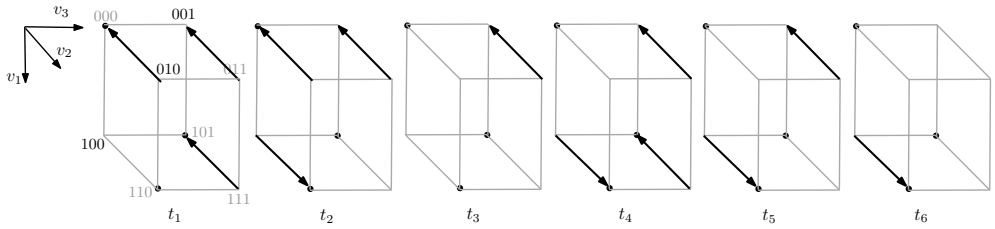


Figure 6.18: The 6 state transitions in direction of v_2, from all functional ASTGs of IG in Figure 6.16. Three black dots: stable states of (S2).

In Table 6.9, there are 5 $K^p(v_2,\cdot)$'s obtained in the mDNF by using direct encoding and PyBoolNet on 6 $K(v_2,\cdot)$. By combining two Boolean expression by extracting the largest common factors, we can get a shorter description of these logical parameters, as shown in Table 6.10.

$K(v_1,\cdot)\backslash$Res	$\emptyset$	$\{v_3\}$	$\{v_2\}$	$\{v_2,v_3\}$	$\{v_1\}$	$\{v_1,v_3\}$	$\{v_1,v_2\}$	$\{v_1,v_2,v_3\}$	f
Boolean variable	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
re-order	$\emptyset$	$\{v_3\}$	$\{v_1\}$	$\{v_1,v_2,v_3\}$	$\{v_2\}$	$\{v_2,v_3\}$	$\{v_1,v_3\}$	$\{v_1,v_2\}$	
	x_1	x_2	x_5	x_8	x_3	x_4	x_6	x_7	
$K^1(v_2,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_5}$	x_8	$\overline{x_3}$	$\overline{x_4}$	$\overline{x_6}$	$\overline{x_7}$	1
$K^2(v_2,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_5}$	x_8	$\overline{x_3}$	$\overline{x_4}$	x_6	x_7	1
$K^3(v_2,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_5}$	x_8	$\overline{x_3}$	x_4	$\overline{x_6}$	x_7	1
$K^4(v_2,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_5}$	x_8	$\overline{x_3}$	x_4	x_6	x_7	1
$K^5(v_2,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_5}$	x_8	$\overline{x_3}$	x_4	x_6	$\overline{x_7}$	1
$K^6(v_2,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_5}$	x_8	x_3	x_4	x_6	x_7	1
$K^{p1}(v_2,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_5}$	x_8	$\overline{x_3}$	$\overline{x_4}$	$(\overline{x_6}\overline{x_7}+x_6x_7)$		(K^1+K^2)
$K^{p2}(v_2,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_5}$	x_8	$\overline{x_3}$	x_4	$(\overline{x_6}x_7+x_6\overline{x_7})$		(K^3+K^5)
$K^{p3}(v_2,\cdot)$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_5}$	x_8		x_4	x_6	x_7	(K^4+K^6)

Table 6.10: For the IG in Figure 6.16, a short description of 6 $K(v_2,\cdot)$'s in terms of Boolean variables, for (S2).

Structure comparison The structures of the state transitions in direction of v_1 for (S1) are very similar to those for (S2). The symmetry of the two groups is related to the different incoming interactions of the IG. The regulations from v_2 and v_3 to v_1 for (S1) are activations, while for (S2) there are inhibitions. As can be seen from Figure 6.19, the structures of the v_1-transitions for (S2) are like a "flip" in the dimension of v_1.

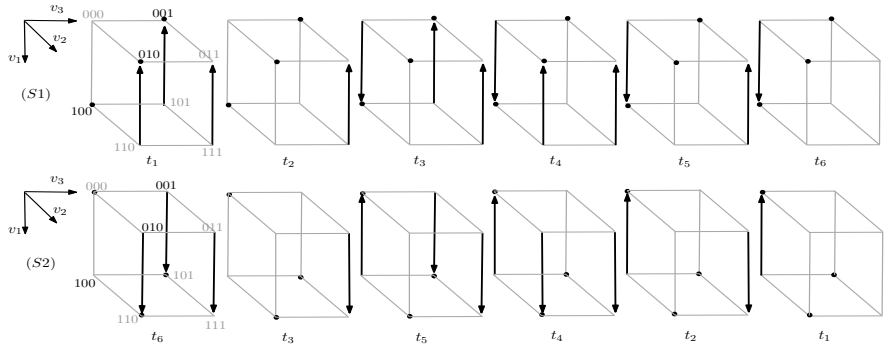


Figure 6.19: Symmetric structures of the state transitions in direction of v_1 for (S1) from all functional ASTGs of IG in Figure 6.14, and for (S2) from all functional ASTGs of the IG in Figure 6.16.

Moreover, the structures of the state transitions in direction of v_1 and v_2 for (S2) are also very similar. v_1 has only positive incoming interactions, while v_2 has one inhibition from v_3. From Figure 6.20 for (S2), the structures of v_2-transitions are like a "rotation" to the back by 180 degrees plus a"flip" in the dimension of v_2.

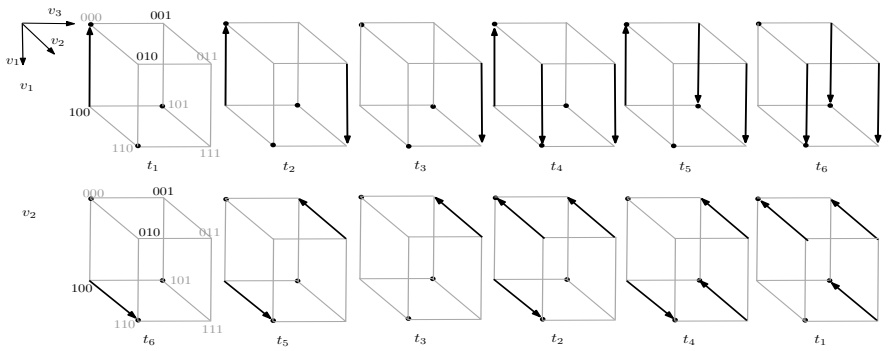


Figure 6.20: Symmetric structures of the state transitions in direction of v_1 and v_2 for (S2), from all functional ASTGs of the IG in Figure 6.16.

6.4 Conclusion and discussion

This chapter explored the structures that are functional for multistabilities. In this specific application, multistabilities are modelled by two hypotheses on three stable states within a system of 3 regulatory components. Without prior knowledge, exhaustive searching is a first choice to spot all solutions under certain required criteria which includes the way of modelling the regulatory system and in which situation is the desired multistabilities satisfied. Both continuous and discrete modelling approaches were applied to obtain all possible regulatory patterns related with these multistabilities.

[Breindl et al., 2011] found a group of GRNs which are able to realise the required three stable steady states. A GRN is modelled by differential equations with only qualitative information from the topology of the GRN. The three stable steady states are treated as a forward invariant set for the system. For each network structure, a robustness measure is defined and transformed into an optimisation problem. A GRN is counted as functional only when there exists a feasible solution to this optimisation problem. For the hypothesis (S1), 206 weakly connected GRNs are found to be functional. For each transcription factor, 6 building blocks are found. A similar process was applied for the hypothesis (S2). 242 weakly connected GRNs are found to be functional, where 4 building blocks are found for one transcription factor, and 8 for the other two.

The two discrete modelling workflows were applied to reveal the relations between multistability and all possible functional GRNs. The multistabilities were modelled as three stable states in a system of three Boolean components. Equivalent initial setups to those in [Breindl et al., 2011] were applied. Both hypotheses (S1) and (S2) are transformed into stable states in a Boolean state space. The forward modelling workflow searches functional IGs from enumerated models whose ASTGs are able to generate the required stable states. For both hypotheses (S1) and (S2), 512 IGs were found where for each component there are 8 building blocks. The reverse engineering workflow was applied as well. It enumerates all graphs on the Boolean state space which have the required stable states. The generalised Lorenz Algorithms are applied to

infer all functional models. At the end, the same set of functional models is found as with the forward modelling workflow.

The reasons for finding more GRNs by the discrete than by the continuous method from [Breindl et al., 2011] can be the following.

1. *The parameters space.* For an IG, all compatible logical parameter functions enable us to see all possible models of this IG. However, in the continuous method, in case of multiple incoming regulations only multiplication was allowed. Therefore, certain types of parameters in the discrete method are not considered in [Breindl et al., 2011].

2. *Modelling the required dynamics.* The three stable steady states in continuous modelling were treated as forward invariant sets. This does not give information whether other stable steady states are allowed. In our discrete modelling method, we allow the existence of other steady states and these can even be isolated. There are some ASTGs which are functional in our approach, but not considered in [Breindl et al., 2011].

Further analysis of these IGs including realizability and the capacity measure show how much those IGs can realise the required dynamics. The logical analysis method is applied on the building blocks for each component and on some logical parameters. More compact descriptions can be obtained. As a result, a few logical IG patterns are found for the two hypotheses. Moreover, the logical parameter patterns are highly related with the state transitions in direction of the corresponding dimension in the state space. The symmetry properties between these structures of the state transitions may lead to other ways of reducing the search space.

Chapter 7

Application: structures for single-stripe pattern

A. Turing in the 1950's described biological pattern formation as a morphogen phenomenon [Turing, 1952]. Later, Wolpert formally developed the concept of morphogen gradient and proposed the famous "French flag" model for the stripe-forming in the embryogenesis of the fruit fly *Drosophila melanogaster* [Wolpert, 1996]. Experimental evidence shows that cells receive *positional information* which instructs them to develop different shapes and structures [Wolpert, 1996]. In the early stages of the embryogenesis of *Drosophila melanogaster*, the forming of single stripes is a crucial spatial phenomenon. Here, the concentration gradient of the transcription factors Bicoid and Dorsal plays a significant role [Jaeger et al., 2004], similar to the role of Activin in the development of the vertebrate neural tube in the frog *Xenopus* [Green, 2002].

A graded signal called *morphogen gradient* leads to gene expression changes in cells and cell differentiation in a concentration-dependent way [Kitano, 2004, Ashe and Briscoe, 2006]. Some morphogen gradients are increasing from the lowest concentration along the cell development, and some are decreasing from high concentration to low. Figure 7.1 illustrates a decreasing morphogen gradient.

It is not clear yet how the morphogen should be interpreted and how it relates to the robustness and precision of biological patterns. Various studies explored the possible mechanisms for morphogen interpretation [Ashe and Briscoe, 2006, Cotterell and Sharpe, 2010, Schaerli et al., 2014, Seeger, 2012]. For example, the mechanisms can be binding-site affinity, combinatorial input, the feed-forward loop, positive feedback, cross repression and reciprocal repressor gradient etc.

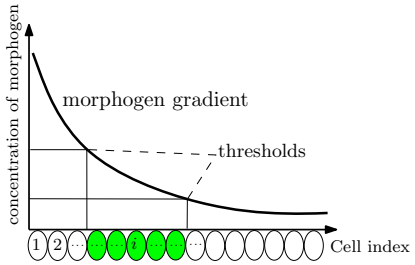


Figure 7.1: The morphogen gradient. Vertical axis: concentration level of the morphogen signal. Horizontal axis: the spatial cell lines receiving different concentration of the morphogen signal.

For decades, the process of stripe-forming by the morphogen gradient in the early embryo development of *Drosophila melanogaster* has been an important topic in developmental biology. [Wolpert, 1968] proposed the famous French Flag problem to model this problem. Later, [Cotterell and Sharpe, 2010] studied a simplified version of the French Flag problem using partial differential equations. They came up with an atlas of GRNs that are able to realise a single stripe. 3-component GRNs were used to model the underlying regulatory system of single stripe-forming: an input gene which receives the morphogen gradient signal, an output gene whose concentration level will show a

single stripe pattern along the cell line, and a third gene representing other factors that affect the stripe-forming. The concentration of the output/target gene is *low-high-low* in a single stripe under the *low-medium-high* levels of the morphogen gradient. At the end, [Cotterell and Sharpe, 2010] identified six distinct regulatory mechanisms to implement the morphogen on a single stripe.

[Seeger, 2012] studied the single stripe-forming problem using discrete modelling and model checking. The morphogen gradient and the single stripe pattern are modelled in the Thomas formalism. The dynamics of the single stripe is then translated into computational tree logic (CTL) and attractor logic for model checking. Two Boolean components are used to model the morphogen gradient, therefore in total 5 components are needed for modelling the stripe-forming system. A Boolean variable with five predecessors has $2^5 = 32$ resources. There are more than 4 billion possible choices for the logical parameter function, which makes it impractical to search through all of them. To deal with a component carrying 5 incoming interactions, [Seeger, 2012] reduced the running time by slicing the scale based on the input layer and stopping search once reaching a certain criteria. Thousands of regulatory networks which can generate a stripe with a certain set of parameters were obtained. Comparing these results with [Cotterell and Sharpe, 2010], five mechanisms out of six were found to be able to form a single stripe. The sixth mechanism was not considered in Seeger's work because the input gene is the same as the output.

In Section 7.1 we study the single stripe problem using our discrete modelling workflows in order to explore possible functional GRNs. The single stripe is modelled by attractors which can include stable states and cyclic attractors. The morphogen gradient signal is modelled by two Boolean components like in [Seeger, 2012]. Section 7.2 presents the results together with a first analysis. The structural properties of these functional IGs are further studied using the logical analysis method in Section 7.3. Finally, we give a conclusion and discussion in Section 7.4.

7.1 Discrete modelling workflows: stripe-forming GRNs

The discrete modelling workflows start with a hypothesis on the desired property of a regulatory system. This desired property usually corresponds to some attractors in the dynamics. The number of components in the regulatory system and the state space need to be determined before representing the desired property by some attractors. Both the enumeration of all possible models or all possible ASTGs with the desired property is taking a lot of time and memory. A full enumeration of a 5-component system would take too much time. For this reason, our goal is to model the system with up to 4 components.

For modelling the *low-medium-high* levels of the morphogen gradient, a multi-valued component seems natural. But, to keep the same setup as [Cotterell and Sharpe, 2010], the morphogen gradient should only regulate the input gene. Section 7.1.1 will present the modelling of the single stripe pattern and the morphogen gradient.

7.1.1 Modelling setup

Single stripe pattern [Cotterell and Sharpe, 2010] defined a single stripe as a continual changing variable along a line of 32 cells. These 32 parallel cells receive a fixed decreasing morphogen gradient signal to the target gene. The output gene shows a single stripe pattern if the following 3 regions are continually observed: a region of low concentration level for at least

2 consecutive cells, followed by a region of high level for a maximum of 16 successive cells, and a region of low level for at least 2 consecutive cells.

[Seeger, 2012] modelled the single stripe in the following way. When the morphogen input is *low*, the required output gene will be *low* after enough time, eventually. For the *medium* morphogen input, the output gene will stay at *high level*, eventually. For the *high* level of morphogen input, the output gene will eventually stay at *low level*.

The gene names from [Seeger, 2012] are applied in this study where the input gene is coloured in red and the target gene in green. Let rr be the input gene and gg be the output gene. The concentration of the gene product can be low or high, denoted by 0 or 1, respectively. The output gene is supposed to show the *off-on-off* pattern under the *low-medium-high* level input signal, as shown in Figure 7.2. The concentration level of gg *eventually* reaches a low (or high) level is specified as: there is at least an attractor where $x_{gg} = 0$ (or 1) for all states. This attractor can be a stable state or a cyclic attractor in the ASTG.

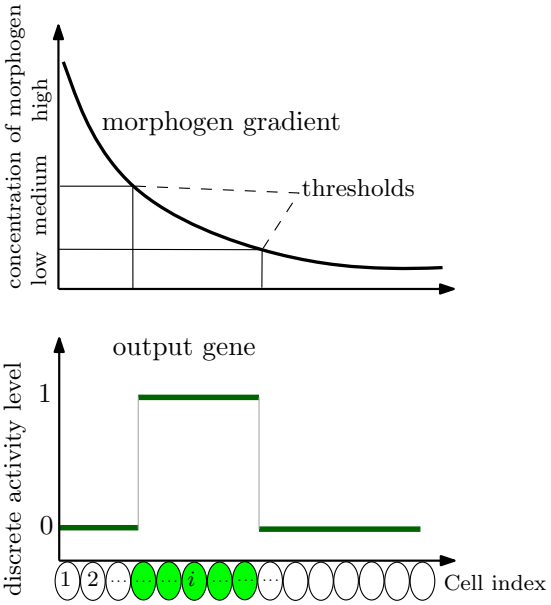


Figure 7.2: The concentration level of the output gene is *high* when the morphogen signal is at *medium* level, and *low* in other two levels.

Definition 7.1 (Single stripe pattern) Consider a GRN including the morphogen gradient signal, an input gene accepting the signal and an output gene. The ASTG of the GRN shows a *single stripe pattern*, if the following appears.

1. When the morphogen signal is at *low* or *high* level, there exists at least one attractor where the output gene is *stable* at *low* level.
2. When the morphogen signal is at *medium* level, there exists at least one attractor where the output gene is *stable* at *high* level.

An IG is called a stripe-forming IG, if it can have a compatible model that generates the single stripe pattern in its ASTG, as in Definition 7.2.

Definition 7.2 (Stripe-forming IG) An IG I (with an input component and an output component) is called a *stripe-forming IG* if, in the ASTG generated by a compatible model $M = (I, K)$, the output component shows a single stripe pattern.

Modelling the morphogen input A morphogen gradient is shown either as an increasing or a decreasing concentration of a gene product. Here we choose an increasing morphogen gradient. This does not change the set of functional models for the single stripe. To achieve *low-medium-high* levels of the morphogen gradient in a discrete setting, at least three distinct concentration ranges are needed. A direct choice is to use a multi-valued component representing the morphogen input m. Let the concentration levels of m be $x_m \in \{0, 1, 2\}$.

Gene rr is receiving the morphogen input and gene gg is the output. We have $x_{rr}, x_{gg} \in \{0, 1\}$, for rr and gg are Boolean. Therefore, the state space X is $\{0, 1, 2\} \times \{0, 1\}^2$. Let the

order of components in a state x be $(x_m x_{rr} x_{gg})$. Let $0--$ represent a subspace including the states with $x_m = 0$, and "$--$" denoting the subspace $\{0,1\}^2$ for x_{rr} and x_{gg}. The morphogen gradient m only regulates the input gene rr. Gene gg can be regulated by rr and itself. Therefore, there are no state transitions between the subspaces $X^{m_0} = \{0\}\times\{0,1\}^2$, $X^{m_1} = \{1\}\times\{0,1\}^2$ and $X^{m_2} = \{2\} \times \{0,1\}^2$.

Take three states from these three subspaces with the same values for x_{rr} and x_{gg}. Let $M = (I, K)$ be some model with $V = \{m, rr, gg\}$ and $m \in \mathrm{Pre}(rr)$. The state transitions in the rr-dimension of the three states are defined by the logical parameters $K(rr, \mathrm{Res}_{rr}(\text{state}))$.

morphogen level	x_m	x	$\mathrm{Res}_{rr}(x)$	$K(rr, \mathrm{Res}_{rr}(x))$
low	0	$(0x_{rr}x_{gg})$	ω	$K(rr, \omega)$
medium	1	$(1x_{rr}x_{gg})$	ω_1	$K(rr, \omega_1)$
high	2	$(2x_{rr}x_{gg})$	ω_2	$K(rr, \omega_2)$

Table 7.1: Logical parameters $K(rr, \mathrm{Res}_{rr}(\text{state}))$ under three states of different morphogen levels.

Assume that m activates the input gene rr. As long as no multiple edges are allowed in the modelling, $\vartheta(m, rr)$ can be either 1 or 2.

1. If $\vartheta(m, rr) = 1$, we get $m \notin \mathrm{Res}_{rr}(0x_{rr}x_{gg}) = \omega$ and $\omega_1 = \omega_2 = \omega \cup \{m\}$. Thus, $K(rr, \omega_1) = K(rr, \omega_2)$.

2. If $\vartheta(m, rr) = 2$, we get $m \notin \omega_1 = \omega$ and $\omega_2 = \omega \cup \{m\}$. Thus, $K(rr, \omega) = K(rr, \omega_1)$.

Though x_m can be constant at three different levels, m can only give one influence to rr. Depending on $\vartheta(m, rr)$, the state transitions in two subspaces are isomorphic, either X^0 and X^1, or X^1 and X^2. In this way, the single stripe pattern is only possible if the ASTG contains two kinds of attractors with x_{gg} constantly being at 0 and 1. This is not a natural setup for a model to form a three-regional single stripe.

In the continuous approach [Cotterell and Sharpe, 2010], m also only regulates rr, but all three levels of effect on gg can be achieved. Multiple edges with m regulating rr on both thresholds 1 and 2 would be needed. However, this is not allowed in our modelling framework. Therefore, the strategy of using a multi-valued component to model the morphogen gradient is not applied. [Seeger, 2012] applied two Boolean components to model the morphogen gradient, three values 00, 01 and 10 are used to denote the levels *low*, *medium* and *high* respectively. There is no restriction on gg if the morphogen components are 11.

The state space X for using two components for the morphogen signal is $\{0,1\}^4$. A state of the system is described by the vector $x = (x_{m_0} x_{m_1} x_{rr} x_{gg})$. $00--$ represents a subspace where $x_{m_0} x_{m_1} = 00$, and "$--$" means x_{rr} and x_{gg} in $\{0,1\}^2$. From the setup, the morphogen components m_0 and m_1 only regulate themselves and the input gene rr. Gene gg can only be regulated by rr and itself. Therefore, for any model satisfying this setup, in the corresponding ASTG, there do not exist state transitions between the four subspaces $X^{00} = \{00\} \times \{0,1\}^2$, $X^{01} = \{01\} \times \{0,1\}^2$, $X^{10} = \{10\} \times \{0,1\}^2$, and $X^{11} = \{11\} \times \{0,1\}^2$.

Now we show that this set up is suitable for generating the single stripe pattern and that it does not cause multiple edges. Let $M = (I, K)$ be a model with $V = \{m_0, m_1, rr, gg\}$ and let ω denote the resource of rr in state $00x_{rr}x_{gg}$. The logical parameters $K(rr, \mathrm{Res}_{rr}(\text{state}))$ for the different states are listed in Table 7.2.

morphogen level	$(x_{m_0}x_{m_1})$	x	$\mathrm{Res}_{rr}(x)$	$K(rr, \mathrm{Res}_{rr}(x))$
low	(00)	$(00x_{rr}x_{gg})$	ω	$K(rr, \omega)$
medium	(01)	$(01x_{rr}x_{gg})$	$\omega \cup \{m_1\}$	$K(rr, \omega \cup \{m_1\})$
high	(10)	$(10x_{rr}x_{gg})$	$\omega \cup \{m_0\}$	$K(rr, \omega \cup \{m_0\})$
other	(11)	$(11x_{rr}x_{gg})$	$\omega \cup \{m_0, m_1\}$	$K(rr, \omega \cup \{m_0, m_1\})$

Table 7.2: Logical parameters $K(rr, \mathrm{Res}_{rr}(\text{state}))$ for states with different morphogen levels.

The components m_0 and m_1 are fixed to activate the input gene rr. A model $M = (I, K)$ is more realistic if it satisfies the observability and the Snoussi-conditions. In order to get a single stripe in the ASTG of M, the following two requirements are needed for the logical parameters $K(rr, \cdot)$.

1. (m_0, rr) and (m_1, rr) are observable. In $M = (I, K)$, there exists resources $\zeta^1 \subseteq \mathrm{Pre}(rr)\backslash\{m_0\}$ and $\zeta^2 \subseteq \mathrm{Pre}(rr)\backslash\{m_1\}$, such that

   (a) $K(rr, \zeta^1) < K(rr, \zeta^1 \cup \{m_0\})$,

   (b) $K(rr, \zeta^2) < K(rr, \zeta^2 \cup \{m_1\})$.

2. (m_0, rr) and (m_1, rr) satisfy the Snoussi-condition. In $M = (I, K)$, for all resources $\zeta^1 \subseteq \mathrm{Pre}(rr)\backslash\{m_0\}$, $\zeta^2 \subseteq \mathrm{Pre}(rr)\backslash\{m_1\}$, it holds that

   (a) $K(rr, \zeta^1) \leq K(rr, \zeta^1 \cup \{m_0\})$,

   (b) $K(rr, \zeta^2) \leq K(rr, \zeta^2 \cup \{m_1\})$.

For the four resources of rr $\mathrm{Res} \in \{\omega, \omega \cup \{m_0\}, \omega \cup \{m_1\}, \omega \cup \{m_0, m_1\}\}$, it is possible that for some $K(rr, \mathrm{Res}) \in \{0, 1\}$, the two requirements above can be satisfied.

Summary The single stripe pattern is modelled by a set of attractors with the target gene gg being *off-on-off* along a *low-medium-high* morphogen gradient. The latter is modelled by three cases $00, 01, 10$ of two Boolean components m_0 and m_1 that activate rr.

Figure 7.3 shows the setup of the method. Each stripe-forming IG contains two Boolean component m_0 and m_1 (the morphogen gradient), an input gene rr which is activated by m_0 and m_1, and an output gene gg. The regulatory interactions between the components rr and gg are not fixed.

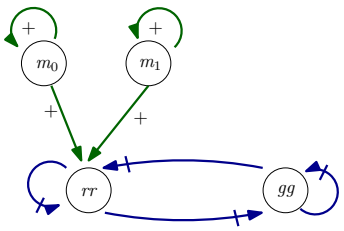


Figure 7.3: The structure of a possible IG. Arrows denote activations, and arrows with blunt dash denote one of the three: *positive, negative, no influence*.

7.1.2 Forward modelling workflow

After fixing the target ASTG attractors and the initial setup of the stripe-forming IGs, the forward modelling workflow can be applied as follows.

1. Enumerate all possible IGs on $\{rr, gg\}$ with morphogen inputs m_0 and m_1 activating rr. Between rr and gg, there are 4 possible interactions, each of which can be an activation, an inhibition, or no influence at all. Therefore, there are $3^4 = 81$ possible IGs.

2. For each IG, enumerate all compatible logical parameter functions. These can be obtained as all the combinations of the logical parameters of each component. For a component u and a resource ω, we have $K(u, \omega) \in \{0, 1\}$. For all $2^{|\mathrm{Pre}(u)|}$ resources, the logical parameters of u belong to $\{0, 1\}^{2^{|\mathrm{Pre}(u)|}}$. This gives $2^{2^{|\mathrm{Pre}(u)|}}$ possible values. All logical parameters $K(u, \cdot)$ have to be checked whether they are compatible with the IG. The number of compatible logical parameters for different cases of a Boolean component v is shown below.

# predecessor	# resources	# $K(u,\cdot)$	# compatible $K(u,\cdot)$
$\lvert\mathrm{Pre}(u)\rvert$	$2^{\lvert\mathrm{Pre}(u)\rvert}$	$2^{2^{\lvert\mathrm{Pre}(u)\rvert}}$	
1	2	4	1
2	4	16	2
3	8	256	9
4	16	65536	114

   At the end of this step, $6,916$ compatible models are found.

3. All enumerated compatible models are checked whether their ASTGs contain a single stripe pattern.

4. Functional model analysis. The realizability is calculated. Building blocks of each component are found in all functional IGs. The logical analysis method is applied for interesting patterns.

7.1.3 Reverse engineering workflow

The reverse engineering workflow includes three steps: enumerating all graphs based on a state space with the single stripe information, model inference with the generalised Lorenz Algorithms and functional model analysis.

Here are the details for the first step. The single stripe information is modelled by a sequence of attractors. These attractors are not fixed but described by a property of x_{gg}. Since m_0 and m_1 only have self-regulations, in any ASTG for such IGs, there do not exist state transitions between the 4 subspaces corresponding to the different values of $x_{m_0}x_{m_1}$: $00 \times \{0, 1\}^2$, $01 \times \{0, 1\}^2$, $10 \times \{0, 1\}^2$ and $11 \times \{0, 1\}^2$. In other words, the ASTG is a graph with 4 isolated subgraphs. Moreover, because gg receives no direct regulations from m_0 and m_1, the state transitions in direction of gg in all 4 subgraphs of such an ASTG are isomorphic to each other.

According to Proposition 3.30 on ASTG construction in Section 3.1.3, the ASTGs in a state space can be constructed by choosing from the eligible row structures for each component. Thus, the enumeration of all graphs based on the state space $\{0, 1\}^4$ is executed for each component with the prior knowledge that the state transitions in direction of gg in all 4 subgraphs of such an ASTG being isomorphic.

The whole process of the reverse engineering workflow is detailed below.

1. Enumerate all graphs based on $X = \{0,1\}^4$ containing a single stripe pattern in two steps:

   (a) Construct all sub-ASTGs based on a subspace $X_{rr,gg} = \{0,1\}^2$. It is the same as in Example 3.1.4, there exists 196 ASTGs which can carry a compatible model.

      i. There are 84 ASTGs carrying attractors where gg is stable at *low* level, denoted by $\mathcal{T}_0$.

      ii. There are 84 ASTGs carrying attractors where gg is stable at *high* level, denoted by $\mathcal{T}_1$.

   (b) Construct graphs based on $X = \{0,1\}^4$ by combining sub-ASTGs on the four subspaces.

      i. For the sub-ASTGs in subspace $00 \times \{0,1\}^2$, choose a sub-ASTG from $\mathcal{T}_0$.

      ii. For the sub-ASTGs in subspace $01 \times \{0,1\}^2$, choose a sub-ASTG from $\mathcal{T}_1$.

      iii. For the sub-ASTGs in subspace $10 \times \{0,1\}^2$, choose a sub-ASTG from $\mathcal{T}_0$.

      iv. For the sub-ASTGs in subspace $11 \times \{0,1\}^2$, choose any one of the 196 sub-ASTGs.

      v. The state transitions in direction of gg in all four subspaces are isomorphic, because there is no regulation from m_0 and m_1 to gg. $\mathcal{T}_0$ and $\mathcal{T}_1$ have 7 gg-slices along rr in common.

   Moreover, there are only 14 different rr-slices along gg in each subspace. In total, $14^4 \times 7 = 268,912$ graphs based on X are enumerated, denoted by $\mathscr{G}_X$.

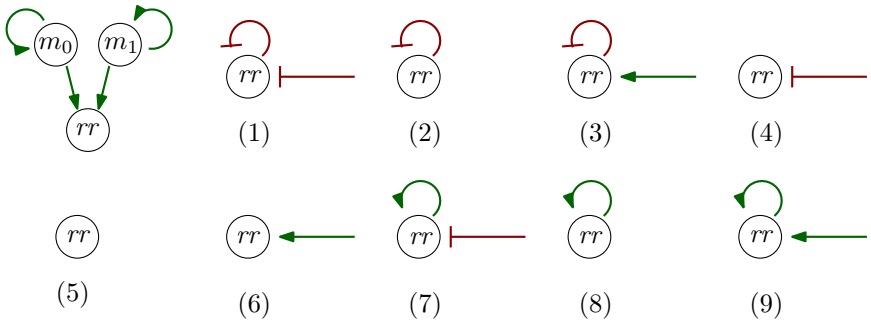
2. Model inference. The *generalised algorithm Observable-Snoussi-Model* is applied for each enumerated graph based on X. Thus, all functional models for the single stripe pattern can be obtained.

3. Functional model analysis. This step is the same as in the forward modelling workflow.

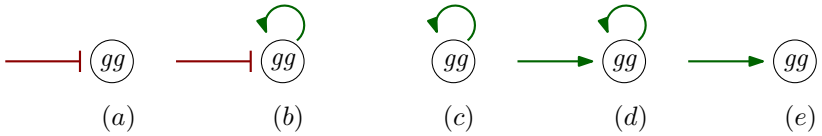
7.2 Results and analysis

7.2.1 Results of the forward modelling workflow

Using the forward modelling workflow, 1520 stripe forming models are obtained which contain 37 different IGs. In every IG, m_0 and m_1 are activating themselves and rr. Figure 7.4 shows the building blocks of rr and gg consisting of the incoming interactions to the component in these stripe-forming IGs.



(a) Building blocks of rr.



(b) Building blocks of gg.

rr	gg
$\{(1),\ldots,(9)\}$	$\{(b),(c),(d)\}$
$\{(7),(8),(9)\}$	$\{(a),(b),(c),(d),(e)\}$
$\{(1),(4)\}$	$\{(a)\}$
$\{(3),(6)\}$	$\{(e)\}$

(c) Stripe-forming IGs.

Figure 7.4: Building blocks from rr and gg from 37 stripe-forming IGs. Blunt edges represent inhibition and arrows denote activation. (a) Two morphogen inputs activate rr, and 9 building blocks of rr. (b) 5 building blocks of gg. (c) 37 stripe-forming IGs represented by 4 groups of building blocks of rr from (a) and of gg from (b), *i.e.*,, combining one building block from the left column under rr and one from the right column under gg gives a stripe-forming IG.

7.2.2 Results of the reverse engineering workflow

From all ASTGs carrying a single stripe, 227 stripe-forming IGs are obtained from the reverse engineering workflow, which can be listed as follows.

Case	(m_0, rr)	(m_1, rr)	# of IGs
1	$(m_0, rr) \notin E$	$(m_1, rr) \notin E$	27
2	$\varepsilon(m_0, rr) = +/-$	$(m_1, rr) \notin E$	21
3	$(m_0, rr) \notin E$	$\varepsilon(m_1, rr) = +/-$	21
4	$\varepsilon(m_0, rr) = +$	$\varepsilon(m_1, rr) = -$	21
5	$\varepsilon(m_0, rr) = -$	$\varepsilon(m_1, rr) = +$	21
6	$\varepsilon(m_0, rr) = -$	$\varepsilon(m_1, rr) = -$	37
7	$\varepsilon(m_0, rr) = +$	$\varepsilon(m_1, rr) = +$	37

Table 7.3: Seven cases among all 227 functional IGs which are obtained by the reverse engineering workflow. For example, case 1 means there are 27 IGs which contain no interactions (m_0, rr) and (m_1, rr).

The 37 IGs in Case 7 with m_0 and m_1 activating rr are the same as the 37 IGs obtained by

the forward modelling workflow in Section 7.2.1. Moreover, there are in total 37 sub-IGs on components rr and gg among all 227 functional IGs. Case 1 is not considered because there are no regulations from the two morphogen components to rr. All sub-IGs on $\{rr, gg\}$ in Case 1 to 6 are contained in those of Case 7.

7.2.3 Five rules for all functional IGs

Only the IGs combined from the building blocks of rr and gg in Figure 7.4c are stripe-forming. The interactions of the morphogen components are the same for all IGs. Therefore, we focus on the remaining part of the IG, the so-called *sub-IG* including only interactions from rr and gg. To characterise the functional IGs, we can use the following 5 *rules*.

Proposition 7.3 Consider an IG with four Boolean components, including 2 morphogen components m_0 and m_1, an input gene rr and an output gene gg. The components m_0 and m_1 can only activate themselves and the input gene rr. Rule 1 and 2 have to be always satisfied. An IG is stripe-forming if it satisfies one of the Rules 3, 4 and 5.

1. gg has no self-inhibition.
2. gg has at least one incoming interaction.
3. An IG is stripe-forming if gg has a self-activation.
4. An IG is stripe-forming if rr has a self-activation and gg satisfies Rule 1 and 2.
5. An IG is stripe-forming, if there is a positive feedback loop between rr and gg and gg satisfies Rule 1.

A detailed analysis on how these rules are related with the stripe-forming ability is given below with respect to the subspaces of X.

a) Rule 1, $\{gg\}$ *has no self-inhibition.* Suppose, gg has a self-inhibition. Then, for a compatible model, there always exists a gg-row in the ASTG such that $\delta(gg, \tau^{gg}) = (1, -1)$

   in all four sub-ASTGs. The transitions of a gg-row are either $(1, 0)$ or $(0, -1)$, but will never be of *pos*-type $(0, 0)$. The gg-slices along rr are isomorphic in the four sub-ASTGs, x_{gg} in the attractors in four sub-ASTGs are the same.

   This means, if Rule 1 does not hold, no single-stripe pattern can be generated.

b) Rule 2, $\{gg\}$ *has at least one incoming interaction*. This rule is added for the following two reasons:

   (a) An output gene is supposed to have at least an incoming regulation. Otherwise, it is trivial that there is no impact from the morphogen signal.

   (b) To carry a single stripe pattern in the ASTG, the incoming regulation can be from rr or itself.

   With gg self-activating, it is always possible that in every sub-ASTG, there are two attractors. All states in one attractor are with $x_{gg} = 0$, and in the other one with $x_{gg} = 1$, which forms exactly a single stripe pattern.

c) Rule 3, *an IG is stripe-forming if gg has a self-activation.* If gg has a self-activation, there always exist suitable parameters which enable the existence of two attractors in every sub-ASTG, one with $x_{gg} = 0$ and one with $x_{gg} = 1$.

d) Rule 4, *an IG is stripe-forming if rr has a self-activation and gg satisfies Rule* 1 *and* 2. Because rr has a self activation, there always exists at least one parameter such that in each sub-ASTG, there is a rr-row τ^{rr} with $\delta(rr, \tau^{rr}) = (0, 0)$. For the other three possible interactions, (rr, gg) and (gg, rr) can be either positive or negative or no influence, gg can have a self-activation or no self-regulation. For all 5 building blocks from gg, with suitable parameters, in all four sub-ASTGs, there always exist two attractors, one with $x_{gg} = 0$ and one with $x_{gg} = 1$.

e) Rule 5, *an IG is stripe-forming, if there is a positive feedback loop between rr and gg and gg satisfies Rule* 1. The interactions between rr and gg can be two positive or two negative interactions. So that there always exist suitable logical parameters to enable two attractors in every sub-ASTG, one with $x_{gg} = 0$ and one with $x_{gg} = 1$. For example, the logical parameters on rr and gg can be constructed as follows:

   (a) If $|\mathrm{Pre}(gg)| = 1$, there is only one possible logical parameter for $K(gg, \cdot)$. If $\mathrm{Pre}(gg) = \{rr\}$, $K(gg, \emptyset) = 0$ and $K(gg, \{rr\}) = 1$. If $\mathrm{Pre}(gg) = \{gg\}$, $K(gg, \emptyset) = 0$ and $K(gg, \{gg\}) = 1$.

   (b) If $\mathrm{Pre}(gg) = \{rr, gg\}$, there exists two possible logical parameters for $K(gg, \cdot)$.

ω	$K^1(gg, \omega)$	$K^2(gg, \omega)$
$\emptyset$	0	0
$\{gg\}$	0	1
$\{rr\}$	0	1
$\{rr, gg\}$	1	1

   From the possible logical parameters for gg, there always exist two attractors, one with $x_{gg} = 0$ and one with $x_{gg} = 1$. Therefore, if an IG satisfies ***Rule*** 5, then this IG is stripe-forming.

   Moreover, if there is a negative feedback loop between rr and gg, then in each sub-ASTG, there always exists a cyclic attractor in which x_{gg} is oscillating, either between $\{(x_{m_0}x_{m_1}00), (x_{m_0}x_{m_1}10), (x_{m_0}x_{m_1}01), (x_{m_0}x_{m_1}01)\}$ or between $\{(x_{m_0}x_{m_1}x_{rr}0), (x_{m_0}x_{m_1}x_{rr}1))\}$.

   Now we want to show how the 37 IGs can be recovered from the 5 rules.

   1. Rule 1 and 2 will always be satisfied.

   2. Rule 3. If there is a self-activation on gg, then all IGs under the same context are stripe-forming. Together with three possible choices (activation, inhibition, no influence) for the other three possible interactions (rr, rr), (rr, gg), (gg, rr), there are $3^3 = 27$ stripe-forming IGs. This rule holds for the combinations of building blocks $\{(1), \ldots, (9)\}$ for rr and $\{(b), (c), (d)\}$ for gg.

   3. Rule 4. If there is no self-loop on gg and there is a self-activation on rr, then (gg, rr) can be $+$, $-$ or no influence. (rr, gg) is either positive or negative, in total we get $3 \times 2 = 6$ stripe-forming IGs. This rule holds for the combinations of building blocks $\{(7), (8), (9)\}$ for rr and $\{(a), (e)\}$ for gg.

   4. Rule 5. If there is neither a self-activation on gg, nor on rr, then (rr, rr) can be only self-inhibition or have no self-loop, while (rr, gg) and (gg, rr) have to be either both positive, or both negative, which leads to $2 \times 2 = 4$ stripe-forming IGs. This rule holds for the combinations of building blocks $\{(1), (4)\}$ and (a), resp. $\{(3), (6)\}$ and (e).

7.2.4 Anti-stripe pattern

These 37 IGs are not only functional for a single stripe, but also for a so-called *anti-stripe*. An *anti-stripe* is symmetric to the single stripe, and is defined below.

1. If the morphogen signal is at *low* or *high* level, gg is *on* and eventually stays at a *high* concentration level. In the ASTG, there exists at least one attractor with $x_{gg} = 1$.
2. If the morphogen signal is at *medium* level, gg is *off* and eventually stays at a *low* concentration level. In the ASTG, there exists at least one attractor with $x_{gg} = 0$.

Moreover, a single stripe and an anti-stripe pattern can coexist in the same ASTG. A stripe-forming IG can be also an anti-stripe-forming IG with the same logical parameter function. Using the same forward modelling workflow, we add a criterion to check for an anti-stripe pattern in the ASTGs from all enumerated models. The result is that the anti-stripe-forming IGs are exactly the same as the stripe-forming IGs.

A self-activation corresponds to a positive circuit at a component. Each of these 37 IGs has at least one positive circuit on rr, or on gg, or between rr and gg.

These positive circuits are crucial for these regulatory systems to generate multiple stable states which verifies the Thomas conjecture about the positive circuits [Thomas and D'Ari, 1990]. Related research on mutistationarity can be found in [Remy et al., 2006, Remy and Ruet, 2006, Remy et al., 2008, Richard, 2009, Thomas and Kaufman, 2001, Didier and Remy, 2012].

In 4 of these 37 IGs, gg receives no regulation from rr. This means that the morphogen components have no impact on gg. Every compatible model of these 4 IGs can generate ASTGs that can carry both single stripe and anti-stripe patterns. Every ASTG from the models of these 4 IGs has four isomorphic gg-slices along rr in four subgraphs. Figure 7.5 shows these 4 sub-IGs and the example of a gg-slice along rr.

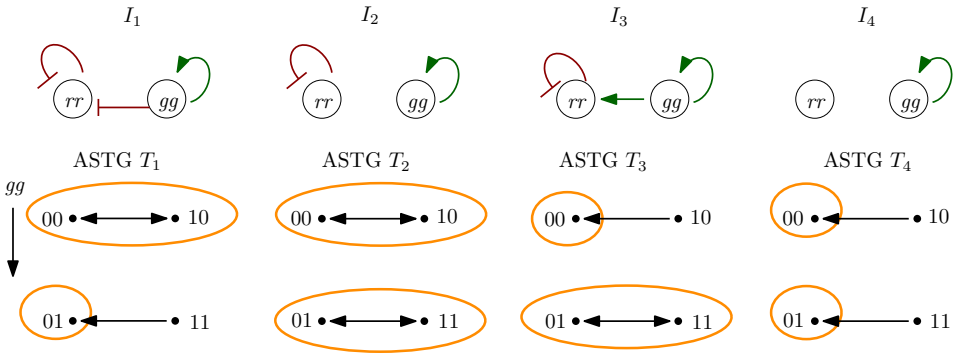


Figure 7.5: 4 sub-IGs where rr does not regulate gg in the IGs that are both stripe-forming and anti-stripe-forming. Below each sub-IG is a gg-slice along rr in all subspaces from one ASTG of the above IG. The logical parameters for the 4 sub-IGs are: for I_1 and I_3, $K(rr, \{rr, gg\}) = 1$, $K(gg, \{gg\}) = 1$; for I_2, $K(rr, \{rr\}) = 1$, $K(gg, \{gg\}) = 1$; for I_4, $K(gg, \{gg\}) = 1$, other unspecified logical parameters under other resources are by default 0.

The appearance of these 4 IGs in the results can have two reasons.

- One reason lies in the discrete modelling method. If an ASTG on $\{0, 1\}^4$ has isomorphic state transitions in the four subspaces $x_{m_0}x_{m_1} \times \{0, 1\}^2$, then in any compatible model of this ASTG, the IG contains isolated components. Moreover, the state transitions in one

dimension depend only on the incoming interactions of the component and its logical parameters. Therefore, in a fixed state space, the state transitions in one dimension do not effect the transitions in the other dimensions.

- Another reason lies in the way of modelling the single-stripe pattern. It does not exclude other attractors. An anti-stripe coexists with a single stripe in the same ASTG. For an array of multiple cells under the morphogen gradient signal, there are the following possibilities for the output gene: x_{gg} is always *on*, or x_{gg} is always *off* along the whole cell line, or x_{gg} shows a stripe *off-on-off*, or an anti-stripe *on-off-on*.

7.2.5 Realizability and the capacity on stripe-forming IGs

There are 9 IGs where rr does not regulate gg (building blocks: (1) to (9) for rr combined with (c) for gg). In the remaining 28 IGs, one IG has the highest realizability, as shown in Figure 7.6a. Out of these 9 IGs, 4 have a realizability 1. They all have a self-regulation on rr but no regulation from rr to gg. The ASTGs generated by all compatible models of these 4 IGs always carry a single stripe pattern. Besides the two morphogen components activating rr, they can be combined with one of the building blocks $\{(1), (3), (7), (9)\}$ for rr and $\{(c)\}$ for gg, as shown in Figure 7.6b.

Figure 7.6: (a) An IG has 228 compatible models, 210 of which can generate a single stripe, $\mathcal{R}_I^A = 0.9211$. (b) All 4 IGs have 114 compatible parameters which are all functional, $\mathcal{R}_I^A = 1$. Arrows denote activation, arrows with a dash denote either activation or inhibition.

In the remaining 28 stripe-forming IGs, some IGs have the same number of functional and compatible parameters, as shown in Table 7.4.

# funct. K's	210	162	114	66	18	9	96	50	22	2	14	9	4	2	5	1	2	1
# compt. K's	228	228	228	228	228	228	114	114	114	114	18	18	18	18	9	9	4	4
$\mathcal{R}_I^A$	0.92	0.71	0.5	0.29	0.08	0.04	0.84	0.44	0.19	0.02	0.78	0.5	0.22	0.11	0.56	0.11	0.5	0.25
# IGs	1	1	2	2	1	1	1	1	2	2	2	2	2	2	2	2	1	1

Table 7.4: Realizability of 28 stripe forming IGs (rounded to 2 digits).

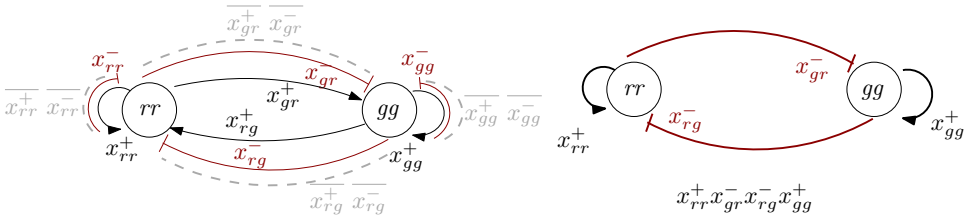
7.3 Logical analysis of functional IGs

The logical analysis method is applied on the resulting stripe-forming IGs in order to get a compact description. Let the stripe-forming ability be represented by a Boolean function f, which takes the value *TRUE* for the 37 stripe-forming IGs.

As introduced in Chapter 4, an IG can be be transformed into a Boolean expression which is a conjunction of Boolean variables representing its interactions. This transformation will be

done for all 37 IGs.

All 37 IGs have in common the two morphogen components activating themselves and rr. To keep the encoding short and simple, only the interactions related within rr and gg are encoded by Boolean variables. Therefore, using partial encoding, only 8 Boolean variables are needed for the 8 possible interactions between two components, as shown in Figure 7.7.



(a) 8 Boolean variables (b) An example

Figure 7.7: (a) 8 Boolean variables needed. (b) An example.

The disjunction of the 37 Boolean expressions is given to the Software PyBoolNet [Klarner et al., 2016], where the Quine-McCluskey algorithm is applied to obtain the minimal DNF. As a result, there are five conjunctive clauses in the minimal DNF, which can be transformed back into sub-IGs, as shown in Figure 7.8.

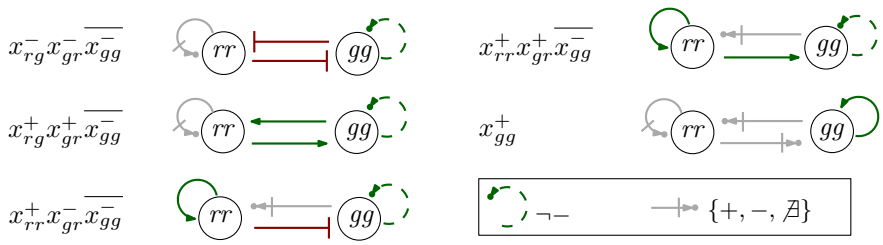


Figure 7.8: The 5 conjunctive clauses in the minimal DNF and the corresponding graphical sub-IGs. Arrows: activations, blunt arcs: inhibitions; dashed arrows ending with a dot: *non negative*, $\neg-$; dashed arrows ending with a dash and a dot: one of activation, inhibition or no influence, $\{+,-,\not\exists\}$.

Together with the two morphogen components activating rr, these five conjunctive clauses are a complete description of all 37 stripe-forming IGs. Moreover, they agree with the 5 rules in Proposition 7.3.

7.4 Conclusion and discussion

This chapter explored the functional structures for single stripe-forming using the discrete modelling workflows. The single stripe is a simplified version of the morphogen interpretation problem. It is modelled by a sequence of attractors where the target output gene is stable at *off-on-off* level for *low-medium-high* levels of the morphogen gradient signal. The morphogen gradient is modelled by two self-activating Boolean components $\{0,1\}^2$, where 00, 01 and 10 denote *low*, *medium* and *high* levels of the morphogen gradient, respectively. It is supposed that the morphogen gradient can only regulate the input gene.

Using the forward modelling workflow, from the all possible IGs and their compatible parameters, 1520 functional models are found that contain 37 IGs. Five rules about the regulations between the input and output genes have been extracted from these results. There is at least one positive feedback loop in each single-stripe-forming IG which can include a self-activation on the input gene and/or the output gene, and/or a positive feedback circuit between these two genes. All single-stripe-forming IGs have no self-inhibition on the output gene.

In Section 7.1.3, the stripe-forming IGs are explored through the reverse engineering workflow. From the theory on ASTG construction, the enumeration of all graphs based on the state space $\{0,1\}^{|V|}$ can be done separately in each dimension.

With the prior knowledge about the single stripe pattern, all possible state transitions for one gene are enumerated. After that, these state transitions in the different dimensions are combined to get all graphs based on the state space with the single stripe pattern prescribed.

As a result, the same set of functional IGs is found as with the forward modelling workflow. The results from both workflows can validate each other and guarantee that all possible stripe-forming IGs have been discovered.

Interestingly, every stripe-forming IG is also anti-stripe-forming, which corresponds to the pattern *off-on-off*. There can be two reasons for the appearance of this anti-stripe pattern. One is that in the discrete modelling method, the state transitions in one dimension are independent from the other dimensions. The other is the possibility of other attractors besides the single stripe pattern.

Using the logical analysis method on the 37 single stripe-forming IGs, the minimal DNF as a compact description is obtained using the Quine-McCluskey algorithm provided by the software PyBoolNet. 5 clauses are found in the minimal DNF, which agree with the 5 rules extracted before.

As future work, other hypotheses can be applied for modelling the single-stripe pattern. The single-stripe pattern can be modelled with a wider range of *on*. Three components instead of two might be considered for the system. The morphogen gradient signal can be allowed to regulate multiple input genes. If the complexity of the parameter space can be reduced to a reasonable level, multi-valued modelling of each component may provide more structural information.

Chapter 8

Summary and discussion

8.1 Summarising the story

Based on the Thomas formalism and discrete modelling, gene regulatory networks (GRNs) can be modelled by interaction graphs (IGs) and logical parameter functions. The expression level of a gene is an integer from $0, 1$ to its maximal activity level. A vector of the expression levels of all components is called a state of the regulatory system. In a particular state, a gene is regulated depending on present activators and absent inhibitors, the so-called regulatory resources. The logical parameter functions define the rules of the regulatory system, *i.e.*, the tendency of each component under all possible regulatory resources. An IG together with a logical parameter function is called a model. The dynamics of a model is given by the asynchronous state transition graph (ASTG), which describes all possible system behaviours. The most interesting behaviours of GRNs are multistationarity and homeostasis. In the ASTGs these behaviours correspond to attractors, which can be stable states or cyclic attractors. The topic of this thesis is exploring the relations between the structure and the dynamics of GRNs.

A circuit in the IG is called positive (negative) if the number of negative interactions in it is even (odd). The Thomas conjectures state that positive circuits are necessary for multistationarity and negative circuits are necessary for homeostasis. Much work has been done on studying the possible types of attractors for positive and negative circuits.

Going from GRN structures to dynamics is straightforward in terms of a model and the corresponding ASTG. Of particular interest are so-called reverse engineering methods, which aim at model inference from given dynamic information. In 2011, T. Lorenz proposed two reverse engineering algorithms which infer from a given ASTG a model satisfying specific model conditions [Lorenz, 2011]. The algorithm *Visibility-Model* can infer a model which satisfies the visibility condition and has only necessary interactions. The algorithm *Observability-Snoussi-Model* can infer a model that satisfies the observability condition and the Snoussi-conditions as much as possible. Two auxiliary algorithms are *Logical-Parameter*, which assigns a proper logical parameter function to a given IG and input ASTG, and *Activity-level*, which assigns a threshold value to a given interaction.

The original Lorenz algorithms [Lorenz, 2011] expect as input an ideal ASTG with complete information, and the constructed model will regenerate the dynamics of the input. However, if the input is not a valid ASTG, this is not possible. Chapter 3 introduces three necessary and sufficient conditions for a graph based on the state space to be a valid ASTG. These include the *asynchronicity*, *unitarity* and the u*-hypercube* condition. Furthermore, the *compatible*

model condition for an ASTG is proposed. A model is more realistic if it is compatible, *i.e.*, it satisfies both the observability and the Snoussi-conditions. In Chapter 3, the Lorenz algorithms are generalised for dealing with general input based on the proposed ASTG conditions. The *generalised algorithm Visibility-Model* will infer a model which satisfies the visibility condition if the input is a valid ASTG, and report an error, if this is not the case. The *generalised algorithm Observability-Snoussi-Model* will output a compatible model if the input is an ASTG satisfying all 4 ASTG conditions, and errors will be reported otherwise.

In Chapter 4, inspired by the classical concept of robustness for the ability of a structure to maintain a specific function under perturbation, a new measure called *realizability* is proposed for the context of discrete modelling. Given some desired dynamical properties, the realizability of an IG is defined as the ratio of all possible functional models over all compatible models. The capacity measure is defined as the ratio of all possible functional models from all functional IGs over all compatible models, which tells how possible the desired properties can be realised.

Two discrete modelling workflows are proposed in Chapter 4. The overall goal is to find all underlying GRNs for some desired dynamical properties, which are modelled as attractors in a predefined state space. Further analysis of the functional models may then allow one to uncover interesting structural patterns. The forward modelling workflow starts from brute force enumeration of all possible models on a set of components and searches for those models whose ASTG admits the required attractors. The reverse engineering workflow starts from enumerating all possible graphs on a state space where the required attractors are prescribed, and obtains all models from the set of enumerated ASTG using the generalised Lorenz algorithms. The set of functional models obtained by both workflows for the same setup and target attractors should be the same. The last step in both workflows is to analyse the set of functional models.

Usually, there are multiple possible GRNs which can realise the same desired function. A classical way is to identify building blocks for each component within the whole set of functional IGs. In Chapter 4, a new logical method for analysing a set of functional IGs is also proposed. Every interaction in an IG is represented by a Boolean variable. An IG can be encoded by a conjunction or product of these variables (and their negation), a set of IGs by a disjunction of these products. The Quine-McCluskey-algorithm is applied on the sum of all these expressions to achieve a minimal disjunctive normal form (DNF). At the end, each clause in the minimal DNF is translated back into an interaction pattern. All functional IGs can be recovered from these interaction patterns. The same logical method can be also applied on a set of logical parameters to obtain a minimal description.

8.2 Contributions and applications

On the methodological side, the main contributions of this thesis include the following:

1. Explaining, implementing, and generalising Lorenz algorithms.
2. Proposing three necessary and sufficient ASTG conditions for a graph based on a state space being a valid ASTG, and a fourth condition for an ASTG being able to have a compatible model.
3. Developing two discrete modelling workflows to explore all GRN structures functional for some desired dynamic properties.
4. Applying the logical analysis method on a set of functional IGs and logical parameters

in order to get a compact description of the whole set and to identify interesting logical patterns.

On the application side, different kind of attractors have been analysed. Chapter 5 studied all potential 3-node IGs that are able to reproduce the cycle from a simplified 3-node MAPK cascade signalling network [Thobe et al., 2014]. Eight different IGs are found in the end, where a core motif and three interaction-pairs cover all cases. While the logical analysis method based on the DNF does not lead to a more compact representation in this particular example, conjunctive normal forms (CNF) provide an interesting alternative, which could be further explored in future research.

Chapter 6 explored all potential 3-node IGs able to produce two types of multiple steady-states related to two hypotheses on cell differentiation in a paper by [Breindl et al., 2011], where a continuous modelling approach is used. The results show that for each hypothesis, there are 512 functional IGs and 8 different incoming-interaction patterns for each component. Any combination of these 8 incoming-interaction patterns forms a functional IG. The results from our discrete method are compared with those from the continuous method. The capacity measure of all IGs and the realization measure of the required attractors is computed, which shows that the two different setups of attractors have the same realization measure. 512 IGs and 18 Boolean variables would be needed for logical encoding, which is too big for the current available software to compute the minimal DNF. Therefore, the logical analysis method is applied only on the 8 incoming-interaction patterns. The results show that there are 5 interactions-patterns for each component under both hypotheses, which can be freely combined and cover the whole set of functional IGs. Additionally, the logical analysis method has been applied on a set of logical parameter functions.

Chapter 7 studied all possible IGs that are able to generate a sequence of attractors for different levels of an input signal, which is one way of modelling the single-stripe phenomenon during the development of the *Drosophila* embryo, the so-called morphogen interpretation [Cotterell and Sharpe, 2010]. Two Boolean components were used for modelling three levels of the morphogen gradient. An input gene receives the activating input signals. An output gene is supposed to exhibit the single-stripe pattern by showing low expression when the input signal is low resp. high, and high expression for medium input. Due to high computational cost, 2-node IGs are applied instead of the three nodes in the literature. The expected attractors can be one or more steady states, or a cyclic attractor with the output gene getting stable at the required level depending on the input signal. Using the reverse engineering workflow, 37 functional IGs were found, which contain 9 different incoming-interaction patterns on the input gene and 5 on the output gene. With the logical analysis method, there are 5 IG patterns obtained from the Boolean encoding of these IGs, which characterise the functional IGs for the single-stripe phenomenon.

8.3 Discussion

There are several limitations of this work. First, the discrete modelling workflows have been applied only on 2- and 3-node GRNs. Both in the forward modelling and the reverse engineering workflow, the double exponentially increasing enumeration space prevents the method from being directly applied to larger systems. Second, the work does not start from experimental data, so that the steps of selecting, curating and discretizing data were skipped. Instead, we started directly from well-studied biological phenomena where the desired dynamic properties were

easy to model by attractors, and the results could be easily compared with. Emphasis was on ASTGs and the generalised Lorenz algorithms for finding a model with a minimal number of interactions in the IG. Moreover, modelling the desired dynamic features by attractors in the ASTGs was done manually. Thus, different hypotheses about the desired functionality require manually selecting the corresponding target attractors.

There are different promising ways for reducing the enumeration space. In the forward modelling workflow, the enumeration of all parameters for an IG can be done separately for each component. In the reverse engineering workflow, all graphs based on the state space with the prescribed attractors can be enumerated component-wise. Therefore, the desired dynamic properties can be decomposed into prior information on each component. With this prior information, only some of the logical parameters enable the desired state transitions for each component. Similarly, the desired attractors on each component can be applied as filters to avoid brute force enumeration.

Another promising direction is to make use of symmetry properties of the graph-theoretical structures representing both IGs and ASTGs. The functional IGs for two similar hypotheses on a desired dynamical behaviour may be similar as graphs. It may be possible to find a transformation so that from a known set of functional IGs for one hypothesis, one can obtain the set of functional IGs for the other hypothesis, without using the discrete modelling workflows. Additionally, those models whose IGs have the same set of edges but with different signs show symmetric graph-theoretical structures in the ASTGs. Each IG can represent a symmetry class of similar IGs, and each ASTG as a graph on the state space can represent a symmetry class of ASTGs. Therefore, a library of IG structures and ASTG structures can be built up for much faster and easier research on GRNs with a small number of components.

Bibliography

[Ashe and Briscoe, 2006] Ashe, H. L. and Briscoe, J. (2006). The interpretation of morphogen gradients. *Development*, 133(3):385–394.

[Batt et al., 2010] Batt, G., Page, M., Cantone, I., Goessler, G., Monteiro, P., and de Jong, H. (2010). Efficient parameter search for qualitative models of regulatory networks using symbolic model checking. *Bioinformatics*, 26:i603–i610.

[Becker et al., 2016] Becker, K., Gebser, M., Schaub, T., and Bockmayr, A. (2016). Answer set programming for logical analysis of data. In *Workshop on constraint-based methods for bioinformatics (WCB'16)*, page 15.

[Bérenguier et al., 2013] Bérenguier, D., Chaouiya, C., Monteiro, P. T., Naldi, A., Remy, E., Thieffry, D., and Tichit, L. (2013). Dynamical modeling and analysis of large cellular regulatory networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 23(2):025114.

[Bernot et al., 2004] Bernot, G., Comet, J.-P., Richard, A., and Guespin, J. (2004). Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347.

[Breindl et al., 2010] Breindl, C., Schittler, D., Waldherr, S., and Allgöwer, F. (2010). A robustness measure for the stationary behavior of qualitative gene regulation networks. *In Proceedings of the 11th International Symposium on Computer Applications in Biotechnology, Leuven, Belgium*, pages 36–41.

[Breindl et al., 2011] Breindl, C., Schittler, D., Waldherr, S., and Allgöwer, F. (2011). Structural requirements and discrimination of cell differentiation networks. *IFAC Proceedings Volumes*, 44(1):11767–11772.

[Chaouiya and Remy, 2013] Chaouiya, C. and Remy, E. (2013). Logical modelling of regulatory networks, methods and applications. *Bulletin of Mathematical Biology*, 75(6):891–895.

[Chaouiya et al., 2008] Chaouiya, C., Remy, E., and Thieffry, D. (2008). Petri net modelling of biological regulatory networks. *Journal of Discrete Algorithms*, 6(2):165–177.

[Chappell et al., 2011] Chappell, W. H., Steelman, L. S., Long, J. M., Kempf, R. C., Abrams, S. L., Franklin, R. A., Bäsecke, J., Stivala, F., Donia, M., Fagone, P., et al. (2011). Ras/Raf/MEK/ERK and PI3K/PTEN/Akt/mTOR inhibitors: rationale and importance to inhibiting these pathways in human health. *Oncotarget*, 2(3):135–164.

[Comet et al., 2013] Comet, J.-P., Noual, M., Richard, A., Aracena, J., Calzone, L., Demongeot, J., Kaufman, M., Naldi, A., Snoussi, E., and Thieffry, D. (2013). On circuit functionality in boolean networks. *Bulletin of Mathematical Biology*, 75(6):906–919.

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

[Cotterell and Sharpe, 2010] Cotterell, J. and Sharpe, J. (2010). An atlas of gene regulatory networks reveals multiple three-gene mechanisms for interpreting morphogen gradients. *Molecular Systems Biology*, 6(1):425.

[Crama and Hammer, 2011] Crama, Y. and Hammer, P. L. (2011). *Boolean functions: theory, algorithms, and applications*. Cambridge University Press.

[De Jong, 2002] De Jong, H. (2002). Modeling and simulation of genetic regulatory systems: a literature review. *Journal of Computational Biology*, 9(1):67–103.

[Didier and Remy, 2012] Didier, G. and Remy, E. (2012). Relations between gene regulatory networks and cell dynamics in boolean models. *Discrete Applied Mathematics*, 160(15):2147–2157.

[Glass, 1977] Glass, L. (1977). *Combinatorial Aspects of Dynamics in Biological Systems*, pages 585–611. Springer US, Boston, MA.

[Green, 2002] Green, J. (2002). Morphogen gradients, positional information, and Xenopus: interplay of theory and experiment. *Developmental Dynamics*, 225(4):392–408.

[Hill, 1910] Hill, A. V. (1910). The possible effects of the aggregation of the molecules of hmoglobin on its dissociation curves. *The Journal of Physiology*, 40:i–vii.

[Ingalls, 2013] Ingalls, B. P. (2013). *Mathematical modeling in systems biology: an introduction*. MIT press.

[Jaeger et al., 2004] Jaeger, J., Blagov, M., Kosman, D., Kozlov, K. N., Myasnikova, E., Surkova, S., Vanario-Alonso, C. E., Samsonova, M., Sharp, D. H., Reinitz, J., et al. (2004). Dynamical analysis of regulatory interactions in the gap gene system of Drosophila melanogaster. *Genetics*, 167(4):1721–1737.

[Jamshidi, 2013] Jamshidi, S. (2013). *Comparing discrete, continuous and hybrid modelling approaches of gene regulatory networks*. PhD thesis, Freie Universität Berlin.

[Kitano, 2002] Kitano, H. (2002). Systems biology: a brief overview. *Science*, 295(5560):1662–1664.

[Kitano, 2004] Kitano, H. (2004). Biological robustness. *Nature Reviews Genetics*, 5(11):826–837.

[Klarner, 2015] Klarner, H. (2015). *Contributions to the analysis of qualitative models of regulatory networks*. PhD thesis, Freie Universität Berlin.

[Klarner et al., 2012a] Klarner, H., Siebert, H., and Bockmayr, A. (2012a). Time series dependent analysis of unparametrized Thomas networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(5):1338–1351.

[Klarner et al., 2012b] Klarner, H., Streck, A., Šafránek, D., Kolčák, J., and Siebert, H. (2012b). Parameter identification and model ranking of Thomas networks. In *Computational methods in systems biology*, pages 207–226. Springer.

[Klarner et al., 2016] Klarner, H., Streck, A., and Siebert, H. (2016). Pyboolnet: a python package for the generation, analysis and visualization of boolean networks. *Bioinformatics*, 33(5):770–772.

[Lorenz, 2011] Lorenz, T. (2011). Vergleich von zwei- und mehrwertigen Modellen bioregulatorischer Netzwerke. Diplomarbeit, Freie Universität Berlin.

[Lorenz et al., 2013] Lorenz, T., Siebert, H., and Bockmayr, A. (2013). Analysis and characterization of asynchronous state transition graphs using extremal states. *Bulletin of Mathematical Biology*, 75(6):920–938.

[McCluskey, 1956] McCluskey, E. J. (1956). Minimization of boolean functions. *Bell System Technical Journal*, 35(6):1417–1444.

[Melliti et al., 2015] Melliti, T., Noual, M., Regnault, D., Sené, S., and Sobieraj, J. (2015). Asynchronous dynamics of boolean automata double-cycles. In *Unconventional Computation and Natural Computation - 14th International Conference, UCNC 2015, Auckland, New Zealand, August 30 - September 3, 2015, Proceedings*, pages 250–262.

[Mendoza et al., 2011] Mendoza, M. C., Er, E. E., and Blenis, J. (2011). The Ras-ERK and PI3K-mTOR pathways: cross-talk and compensation. *Trends in Biochemical Sciences*, 36(6):320 – 328.

[Orton et al., 2005] Orton, R. J., Sturm, O. E., Vyshemirsky, V., Calder, M., Gilbert, D. R., and Kolch, W. (2005). Computational modelling of the receptor-tyrosine-kinase-activated mapk pathway. *Biochemical Journal*, 392(2):249–261.

[Ostrowski et al., 2016] Ostrowski, M., Paulevé, L., Schaub, T., Siegel, A., and Guziolowski, C. (2016). Boolean network identification from perturbation time series data combining dynamics abstraction and logic programming. *Biosystems*, 149:139–153.

[Palinkas, 2015] Palinkas, A. (2015). *Integrated modelling of metabolic and regulatory networks*. PhD thesis, Freie Universität Berlin.

[Perkins et al., 2010] Perkins, T., Wilds, R., and Glass, L. (2010). Robust dynamics in minimal hybrid models of genetic networks. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 368(1930):4961–4975.

[Petrick, 1959] Petrick, S. (1959). On the minimization of boolean functions. In *Proceedings of the international conference information processing, Paris*, pages 422–423.

[Quine, 1952] Quine, W. V. (1952). The problem of simplifying truth functions. *The American Mathematical Monthly*, 59(8):521–531.

[Remy et al., 2016] Remy, E., Mossé, B., and Thieffry, D. (2016). *Boolean dynamics of compound regulatory circuits*, pages 43–53. Springer International Publishing, Cham.

[Remy and Ruet, 2006] Remy, E. and Ruet, P. (2006). On differentiation and homeostatic behaviours of boolean dynamical systems. *Lecture Notes in Computer Science*, 4230:153–162.

[Remy et al., 2006] Remy, É., Ruet, P., and Thieffry, D. (2006). Positive or negative regulatory circuit inference from multilevel dynamics. *Positive Systems*, pages 263–270.

[Remy et al., 2008] Remy, É., Ruet, P., and Thieffry, D. (2008). Graphic requirements for multistability and attractive cycles in a boolean dynamical framework. *Advances in Applied Mathematics*, 41(3):335–350.

[Richard, 2009] Richard, A. (2009). Positive circuits and maximal number of fixed points in discrete dynamical systems. *Discrete Applied Mathematics*, 157(15):3281 – 3288.

[Richard, 2010] Richard, A. (2010). Negative circuits and sustained oscillations in asynchronous automata networks. *Advances in Applied Mathematics*, 44(4):378 – 392.

[Richard and Comet, 2007] Richard, A. and Comet, J.-P. (2007). Necessary conditions for multistationarity in discrete dynamical systems. *Discrete Applied Mathematics*, 155(18):2403–2413.

[Ruet, 2017] Ruet, P. (2017). Negative local feedbacks in boolean networks. *Discrete Applied Mathematics*, 221:1–17.

[Schaerli et al., 2014] Schaerli, Y., Munteanu, A., Gili, M., Cotterell, J., Sharpe, J., and Isalan, M. (2014). A unified design space of synthetic stripe-forming networks. *Nature Communications*, 5.

[Schittler et al., 2010] Schittler, D., Hasenauer, J., Allgöwer, F., and Waldherr, S. (2010). Cell differentiation modeled via a coupled two-switch regulatory network. *Chaos*, 20(4).

[Seeger, 2012] Seeger, M. (2012). Morphogene interpretation in boolean regulatory networks. Bachelor thesis, Freie Universität Berlin.

[Snoussi, 1989] Snoussi, E. H. (1989). Qualitative dynamics of piecewise-linear differential equations: a discrete mapping approach. *Dynamics and Stability of Systems*, 4(3-4):565–583.

[Soulé, 2003] Soulé, C. (2003). Graphic requirements for multistationarity. *ComPlexUs*, 1(3):123–133.

[Streck, 2016] Streck, A. (2016). *Toolkit for reverse engineering of molecular pathways via parameter identification*. PhD thesis, Freie Universität Berlin.

[Streck et al., 2015] Streck, A., Lorenz, T., and Siebert, H. (2015). Minimization and equivalence in multi-valued logical models of regulatory networks. *Natural Computing*, 14(4):555–566.

[Thieffry, 2007] Thieffry, D. (2007). Dynamical roles of biological regulatory circuits. *Briefings in Bioinformatics*, 8(4):220–225.

[Thobe, 2017] Thobe, K. (2017). *Logical modeling of uncertainty in signaling pathways of cancer systems*. PhD thesis, Freie Universität Berlin.

[Thobe et al., 2014] Thobe, K., Streck, A., Klarner, H., and Siebert, H. (2014). *Model integration and crosstalk analysis of logical regulatory networks*, pages 32–44. Springer International Publishing, Cham.

[Thomas, 1973] Thomas, R. (1973). Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42(3):563 – 585.

[Thomas, 1981] Thomas, R. (1981). On the relation between the logical structure of systems and their ability to generate multiple steady states of sustained oscillations. *Springer Series Synergetics*, pages 180–193.

[Thomas and D'Ari, 1990] Thomas, R. and D'Ari, R. (1990). *Biological feedback*. CRC press.

[Thomas and Kaufman, 2001] Thomas, R. and Kaufman, M. (2001). Multistationarity, the basis of cell differentiation and memory. II. Logical analysis of regulatory networks in terms of feedback circuits. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 11(1):180–195.

[Thormählen, 2016] Thormählen, T. (2016). Lecture of technical computer science I, interactive demonstrations, 5.3 Quine-McCluskey algorithm. `http://www.mathematik.uni-marburg.de/~thormae/lectures/ti1/code/qmc/`.

[Turing, 1952] Turing, A. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 237(641):37–72.

[Videla et al., 2015] Videla, S., Guziolowski, C., Eduati, F., Thiele, S., Gebser, M., Nicolas, J., Saez-Rodriguez, J., Schaub, T., and Siegel, A. (2015). Learning boolean logic models of signaling networks with ASP. *Theoretical Computer Science*, 599:79–101.

[Wagner, 2013] Wagner, A. (2013). *Robustness and evolvability in living systems*. Princeton University Press.

[Winter et al., 2011] Winter, J. N., Jefferson, L. S., and Kimball, S. R. (2011). ERK and Akt signaling pathways function through parallel mechanisms to promote mTORC1 signaling. *American Journal of Physiology-Cell Physiology*, 300(5):C1172–C1180.

[Wolpert, 1968] Wolpert, L. (1968). The french flag problem: a contribution to the discussion on pattern development and regulation. *Towards a Theoretical Biology*, 1:125–133.

[Wolpert, 1996] Wolpert, L. (1996). One hundred years of positional information. *Trends in Genetics*, 12(9):359–364.

Nomenclature

Δ symmetric difference, page 13

δ state transition function, page 7

$\delta(u, \tau^u)$ state transitions of a u-row in direction of u, page 13

$\delta(u, x)$ state transition function of a component u in a given state x, page 7

$\mathrm{dest}(u, \tau^u)$ destination values of a u-row in direction of u, page 40

$\mathrm{dest}(u, x^i)$ destination value of a state x^i in a u-row, page 40

$\mathbf{e}^u$ the u-th unit vector in X, page 8

$\mathcal{C}^A$ the capacity measure of an IG I for a desired property A , page 73

$\mathcal{G}_X = (X, \mathcal{E})$ a graph based on a state space X, page 40

$\mathcal{R}_I^A$ realizability of an IG I functional for some desired properties A, page 72

$\mathcal{R}_{I,v}^A$ realizability of a component of an IG I functional for some desired properties A , page 73

$\mathrm{Suc}(u)$ the predecessor set of node u, page 7

$\mathscr{X}_u$ u-hypercube, t is an integer, page 42

$\mathscr{X}_u^{c_v}$ complement of $\mathscr{X}_u$ in the v-th dimension, page 48

$\mathrm{Pre}(u)$ the predecessor set of node u, page 7

$\mathrm{Res}_u(x)$ the resource for u in a state x, page 7

τ^u a u-row, page 13

$\{\tau^u\}_{elg}^t$ set of eligible u-row structures of a threshold value , page 55

A an attractor, can be a stable state or cyclic attractor, page 8

$F^+(x, \vartheta),\ F^-(x, \vartheta)$ step functions, page 5

$F_m^+(x, \vartheta), F_m^-(x, \vartheta)$ Sigmoid curves, Hill functions, page 4

$H(u)$ the set of all u-hypercubes in X, page 42

$I = (V, E, \varepsilon, \vartheta, \max)$ an interaction graph, V: components, E: edges, ε: signs of E, ϑ: thresholds of E, $\max$: maximal activity levels of V , page 7

K logical parameter function, page 7

$K(u, \omega)$ logical parameter of a component u with resource ω, page 7

$M = (I, K)$ a model, page 7

S transitions among all states, page 8

$T_M = (X, S)$ asynchronous state transition graph of a model M, page 8

X a state space, page 7

x a state, page 7

$X^{u,\omega}$ set of states for a component under a resource , page 45

X_i $X_i = \{0, \ldots, \max_i\}$, page 42

List of Figures

List of Tables

Zusammenfassung

Ein Schlüsselthema der Systembiologie ist das Verstehen der komplexen Beziehungen zwischen molekularen Netzwerkstrukturen, dynamischen Eigenschaften und biologischer Funktion. In diesem Zusammenhang sind genregulatorische Netzwerke (GRN), welche die regulatorischen Interaktionen zwischen Genen und ihren Produkten beschreiben, von grundlegender Bedeutung. Das allgemeine Ziel dieser Dissertation ist die Erforschung der Beziehungen zwischen der Struktur und der Dynamik genregulatorischer Netzwerke. Dies geschieht in einem diskreten Modellierungsrahmen unter Verwendung des Thomas-Formalismus. Ein GRN wird durch ein diskretes Modell dargestellt, das einen Interaktionsgraphen (IG) und eine logische Parameterfunktion umfasst, welche die regulatorischen Interaktionen charakterisieren. Die Dynamik des GRN wird durch einen asynchronen Zustandsübergangsgraphen (ASTG) modelliert, bei dem Zustände nur durch asynchrone und unitäre Updates geändert werden können. Im Jahr 2011 schlug T. Lorenz zwei Rückwärtsinferenzalgorithmen vor, um aus einem gegebenen ASTG Modelle mit spezifischen Eigenschaften herzuleiten.

Im ersten Teil der Dissertation liegt der Schwerpunkt auf der Erläuterung, Implementierung und Verallgemeinerung der Lorenz-Algorithmen. Um allgemeine Eingaben zu behandeln, werden drei notwendige und hinreichende Bedingungen vorgeschlagen, um ASTGs in der Menge aller Graphen auf einem gegebenen Zustandsraum zu charakterisieren. Darüber hinaus wird eine vierte Bedingung hergeleitet, die notwendig und hinreichend dafür ist, dass ein ASTG ein realistisches Modell zulässt. Diese vier ASTG Bedigungen bilden die Grundlage für eine Verallgemeinerung der Lorenz-Algorithmen und mehrere Anwendungen.

Multistationarität und Homöostase sind zwei wichtige dynamische Eigenschaften von hoher biologischer Relevanz, die durch Attraktoren im ASTG dargestellt werden können. Im zweiten Teil der Dissertation werden zwei diskrete Modellierungsabläufe entwickelt, um all diejenigen GRN zu untersuchen, die in der Lage sind, eine gegebene Funktionalität zu realisieren. Der Vorwärtsablauf beinhaltet die Aufzählung aller möglichen Modelle und die Suche nach denjenigen Modellen, deren ASTG die gewünschten Eigenschaften aufweist. Der Rückwärtsablauf beginnt mit der Aufzählung der Graphen im Zustandsraum, die die dynamischen Eigenschaften erfüllen, und leitet daraus unter Verwendung der verallgemeinerten Lorenz-Algorithmen alle zugehörigen Modelle her. Zur Analyse der erhaltenen funktionalen IG wird ein logisches Analyseverfahren entwickelt, das IG durch Boolesche Ausdrücke codiert und daraus durch Minimierung Boolescher Funktionen eine kompakte Darstellung gewinnt. Dieses logische Analyseverfahren kann auch auf die logischen Parameter angewandt werden.

Im letzten Teil der Dissertation werden die diskreten Modellierungsabläufe angewandt, um den Raum der GRN zu untersuchen, die einige typische dynamische Verhaltensweisen von biologischem Interesse umsetzen. Drei Fallstudien werden vorgestellt. Die erste betrifft Homöostase in einer vereinfachten MAPK-Kaskade, die zweite Multistationarität in der Zelldifferenzierung und die dritte die Bildung einfacher Streifen in der Embryonalentwicklung der Fruchtfliege *Drosophila melanogaster*.

Acknowledgements

My most sincere, deepest and heartfelt gratitude is for my supervisor, Prof. Dr. Alexander Bockmayr, for his unlimited support and guidance in every step of my PhD, for his great patience and tolerance so that I could take my time to learn and try different topics, for his thoroughness, encouragement, for the countless time committed in our discussion and proofreading.

Next, I want to thank Dr. Élisabeth Remy, HDR for kindly accepting to become a reviewer of the thesis.

I would like to thank Therese Lorenz, for her valuable advice and proofreading the mathematics in the thesis, for all those interesting and enlightening discussions. I have enjoyed working with her and benefited from her solid mathematics knowledge.

I also want to thank all present and former colleagues in the work groups, *Mathematics in Life Sciences* and *Discrete Biomathematics*, for creating this friendly and supportive research atmosphere, namely Alexander, Heike, László, Shahrad, Hannes, Aljoscha, Arne, Alexandra, Kirsten, Firdevs, Annika, Yaron, Therese, Katinka, Matilde, Steffen, Adam, Robert, Lín, Neveen and Markus. I want to thank Shahrad, Hannes, Aljoscha, Kirsten, Firdevs, Katinka, Annika and Alexandra for all the helpful tips and discussions. I want to thank those with whom I shared the same office, László, Marco, Annika, Yaron, Therese and Neveen. Moreover, I want to thank Katja Geiger and Ekaterina Engel for all the help and especially in many documents. Because of you all, Berlin becomes my another hometown.

I am very grateful for the four years scholarship from China Scholarship Council which allows me to start and focus on the PhD study, and for additional funding and support from Freie Universität Berlin through Dahlem Research School.

Thanks to all my friends who keep me supported and connected all through my years of PhD. Thanks to Raphael, Sunil and Lín Liú for proofreading a few chapters. Special thank to Tiěyuán, Wénxiù, Shūjūan, Zhìguǎng for all the time encouragement. To Liú Píngpíng, Lóng Tāo, Hé Jìng and Zhìmín Xiāo, Jiā̀lu, Lín Rén, Zhèngx̄ing, Hán, J̌ingwēi, Hào, Yuè, Wěi, Xǔe, Guóx̄ing, Zhènghào, Wáng Biāo, Yàn L̆i and Sunil who contributed on my well-being during my PhD study.

I want to thank Zhìyǒng Liáng who has joined me in Berlin, consistently supported me and renewed my understanding of life.

Last, I want to express my very deep appreciation to my parents, my brother and uncle Déqiáng Lěng, for the life time unconditional everlasting trust, support and love.

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich alle Hilfsmittel und Hilfen angegeben habe und versichere, auf dieser Grundlage die Arbeit selbständig verfasst zu haben. Die Arbeit wurde nicht schon einmal in einem früheren Promotionsverfahren eingereicht.

_____

Ling Sun

Berlin, August 2017