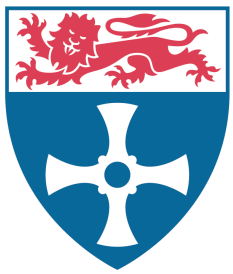
Compositional Techniques and Tools for Constructing and Analysing Boolean Networks



Hanin Abdulrahman

School of Computing
Newcastle University

This dissertation is submitted for the degree of
Doctor of Philosophy

March 2023

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. Throughout this thesis, I used plural pronouns, which does not indicate multiple authorship unless stated explicitly.

Hanin Abdulrahman
March 2023

Acknowledgements

First and foremost, I gratefully acknowledge the deepest gratitude to my supervisor, Jason Steggles, for his great guidance and wise supervision. He always made time for meetings to ensure that my work was proceeding smoothly. Without his great and continuing support and encouragement, this research would never have become a reality. I would also like to thank Hanadi Alkhudhayr for many interesting discussions on Boolean network composition.

During this journey, I could not have survived without my family. My husband KHALED and my two wonderful daughters TALA and SEBA, you gave me the strength to continue through your absolute love, support, understanding and patience. My parents Yahya and Horeyah, your unconditional love, continuous support and prayers influence my life. My sisters and brothers, Ebtihal, Rahaf, Yasser, Marwan and Amer, you have given me nothing but love and support throughout this work. I thank God for all the things that have happened to me in my life.

I would like to thank my friends who shared the happiness with me and supported me through all of the hard times on this journey: Aisha Blfageh, Aisha Alarfaj, Amal Khalifah, Badreyah Almutairy, Ebtisam Abdulqader, Hadeel Alateeq, Hanadi Alkhudhayr, Malak Alharbi and Shaimaa Bajouda.

Finally, I gratefully acknowledge that this research has been supported by Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia, through their scholarship programme, allowing me to further my studies experience.

Abstract

Boolean networks are an important qualitative modelling approach that are widely used in biological modelling. In particular, *attractor analysis* (i.e. finding key cyclic behaviour) has been a crucial tool for analysing biological systems. The practical application of Boolean networks is limited by the state space explosion problem. To address this, researchers have considered applying compositional analysis techniques to Boolean networks. Recently, a new compositional framework for constructing and analysing Boolean networks was developed at Newcastle University. This framework provides a foundation for both engineering Boolean networks and decomposing them to aid analysis. While the initial results for this framework are interesting, its practical application currently has some important limitations: the definition of a composition is too restrictive, and it lacks support for compositional attractor analysis.

In this thesis, we set out to address these practical limitations of the existing compositional framework for Boolean networks. We significantly strengthened and extended this compositional framework by developing a new general structure for compositions and by providing new results and techniques to compositionally identify the attractors of a Boolean network. Our attractor analysis approach is based on using *strongly connected components* to identify potential cyclic behaviour, taking into account the interference arising from a composition, and then merging these cyclic behaviour using an important new property called *interference alignment*. We began by identifying attractors in a composition involving a set of two BNs and a set of three BNs. However, it became clear that extending the results to multiple Boolean networks with multiple entities to be merged was constrained by the complexity of the existing compositional framework. Therefore, a new generalised version of a composition was developed and the compositional attractor analysis techniques were then extended to this new compositional approach.

To support the practical applications of our techniques of compositionally identifying the attractors, a prototype support tool was developed. This involved developing

a new algorithmic approach based on the underlying theoretical results we have developed. During the development of the tool, two small case studies were undertaken to gain insight into its practical application. The final tool was implemented, and its performance was evaluated by using a series of compositional tests. Moreover, the tool's performance was compared to the performance of three mature tools from the literature, and the results were very promising.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Qualitative modelling techniques [8, 9] have increasingly represented a key tool in modelling and analysing biological systems since they avoid the need for problematic quantitative data concerning reaction rates. Such techniques afford useful approaches to enabling development methods and tools for analysing and synthesising large biological systems such as *gene regulatory networks (GRNs)* [10, 9], *signal transduction networks (STNs)* [11, 12] and *metabolic pathways (MPs)* [13, 14]. A wide range of these approaches, including *Boolean networks (BNs)* [15, 16], *multi-valued networks (MVNs)* [17–19] and *Petri nets (PNs)* [20–22], have been successfully used to gain important insights into biological systems.

BNs are a qualitative modelling approach that has been widely used to model and analyse biological systems (e.g. [23–27, 2]). The modelled systems can be analysed by identifying their *attractors* (i.e. key cyclic behaviour) and this can provide important biological insights [28, 7]. BNs were first introduced by Kauffman (1969) to study the dynamical properties of GRNs [15, 16], then by Thomas (1973), who described the complex dynamics of GRNs by developing an asynchronous logical approach [29, 24]. In recent years, they have been successfully used to model and analyse GRNs in the context of expression pattern prediction [23], cellular states (e.g. proliferation, apoptosis and differentiation) [28] and evolution [30]. A wide range of biological regulatory networks have been modelled and analysed by BNs, such as the mammalian cell cycle [25], the expression of *Drosophila* segment polarity genes [23], the yeast cell cycle [31] and the differentiation of T-helper cells [32].

In structure, a BN is a qualitative formalism consisting of a set of entities (e.g. genes, proteins and mRNAs) whose states are either 0 (i.e. *inactive*) or 1 (i.e. *active*), and whose regulatory interactions among the entities are based on specified next-state Boolean functions. Transitions between states are normally either *asynchronous*, meaning that each entity updates its state separately or *synchronous*, meaning that all entities update their states simultaneously. A range of other update semantics, such as sequential and block-sequential asynchronous, have also been considered in the literature [33]. In this research, we focus on synchronous BNs, which are considered to be important and helpful in studying the dynamics of modelled biological systems, as they are simple to interpret [34] and are appealing for automated analysis, because there is a deterministic pathway through the state space. Furthermore, synchronous BNs have been well studied in the literature to date [15, 10, 35–37].

In a synchronous BN, any sequence of consecutive state transitions of a BN eventually converges to cycles called *attractors*. They represent an important dynamical property of modelled systems, which can subsequently be linked to biological phenomena. Attractors are widely considered in analysing biological systems (see for example [38–41]). They correspond to different cellular phenotypes, including *proliferation* (cell cycle), *apoptosis* (programmed cell death) and *differentiation* (execution of tissue-specific tasks) [42, 43, 28]. For example, in [44], attractors of the *Arabidopsis thaliana* flower organ specification GRN correspond to the wild-type gene expression patterns of inflorescence, sepal, petal, stamen and carpel primordial cells during the different stages of flower development.

Modelling and analysing large-scale regulatory systems using BNs is a key challenge due to the *state space explosion problem* [45]. For n entities in a BN, the state space has 2^n states. This exponential growth in states severely restricts size of the models from being practical to analyse. This has motivated many researchers to investigate new techniques and tools to address the state space explosion problem and practical analysis (e.g. [46, 47, 36, 48]).

1.2 Motivation

Developing new approaches to overcome the exponential growth of the state space problem is crucial to enabling the practical application of BNs for modelling and analysing complex systems [28, 7]. To address this limitation researchers have considered applying compositional and decompositional techniques to BNs (for example, see

[47, 36, 49–51]). The compositional approaches allow complex models to be constructed from smaller ones, such that their properties can be analysed without considering the entire model. This approach also allows the inference of the composed model's behaviour from the underlying BNs in order to address the limitations imposed by the state space explosion problem. It provides a potential way to decompose models by breaking a model into sub-models, and referring to its properties in light of distinct components which can accommodate the complexity of analysing and gaining insights into complex systems' dynamics.

An interesting new compositional framework for BNs was proposed by *Alkhudhayr* and *Steggles* [48, 49, 4]. The basic idea of that framework is composing synchronous BNs by merging entities using logical connectives (e.g. conjunction and disjunction). The compositional techniques in this framework allow the construction of BNs to aid in engineering large models, and from an analytical point of view, so far, the focus has been on behaviour preservation [49, 48, 4]. However, more work is required to facilitate its practical application. This thesis considers using this developed framework practically to engineer and analyse biological systems. We identified two limitations that need to be addressed in this framework. First, it does not support BN attractor analysis, which is a crucial point of interest in analysing biological systems [28, 7]. Second, the form of compositions is currently restricted to a sequential structure of a composition. Therefore, we focus on extending the existing framework by developing new techniques and tools to identify BN attractors compositionally, and generalising the concept of a composition to facilitate the engineering and analysing biological systems.

1.3 Aim

The aim of this PhD research is to:

Develop practical compositional techniques and tools to support engineering and analysing BNs by extending an existing compositional framework [48, 49, 4].

This project focuses on extending an existing compositional framework [48, 49, 4]. During this work, it became clear that there were two main limitations which needed to be addressed for this framework:

- Lack of techniques and tools for compositional attractors analysis.
  The existing framework provides analysis techniques based on behaviour preservation and does not provide facilities for compositionally identifying attractors.
- Restricted compositional structure.
  The current one is based on two BNs or a sequence of BNs being composed together and that places some restrictions and limits the practical application.

1.4 Overview of Objectives

Based on the aim and the ideas introduced in Section 1.3, the following objectives were identified:

1. Research
   1.1. To study BNs and produce a comprehensive introduction to BNs.
   1.2. To research the existing compositional framework with focus on its practical limitations.
   1.3. To research existing work on attractors identification and analysis.
2. Theory
   2.1. To develop new definitions and results for compositional attractor analysis based on the existing compositional framework (where a composition is restricted to two BNs).
   2.2. To generalise the existing compositional framework to support arbitrary compositions involving multiple BNs and entities.
   2.3. To extend the definitions and results of attractor analysis in *Objective 2.1* to apply them to the generalised composition structure.
3. Tools
   3.1. To develop techniques and algorithms for implementing the results of *Objective 2.1* and *Objective 2.3*.
   3.2. To develop a prototype support tool for applying the techniques and algorithms of *Objective 3.1*.
   3.3. To evaluate the practical performance of the developed techniques and tools.

1.5 Contributions

The following are the key contributions of this PhD project:

1. **Develop a novel compositional approach for attractor analysis**
   A significant contribution of this PhD thesis is the development of a new theoretical basis for compositionally identifying attractors in an arbitrary composition involving multiple BNs. This new approach is an important extension to the existing framework, and significantly different to other approaches that exist in the literature (see the discussion of related work in Section 2.5.3). The basic idea of this approach is based on analysing the *interference state graph* [49], a state graph that extends a BN's normal behaviour with additional dynamics that can occur due to interference from a composition. We identify the *strongly connected components (SCCs)* [52] associated with the interference state graphs for the underlying BNs, and then selectively merge their cyclic behaviour using an important new property called *interference alignment*.

   The initial work focused on a set of two BNs and a set of three BNs involved in a composition, and developed theoretical definitions and results to identify attractors of a composed model. We formally show that our approach is correct by proving that it is *sound* (all attractors found are present in the composed model) and *complete* (every attractor in the composed model is found using the approach).

   The initial work on attractor analysis was extended to consider an arbitrary composition consisting of multiple BNs merged on multiple entities. This has resulted in a new approach to defining a composition (see *Contribution 2*), and involved extending the results.

2. **Generalise the compositional definition to allow arbitrary compositions**
   Another significant contribution is the generalisation of the definition of a composition to allow arbitrary compositions based on an underlying graph structure. In association with that, we set new definitions and notations to support the generalised composition. This is very important in taking the step from identifying attractors in a composition consisting of two BNs to an arbitrary structure of a composition consisting of multiple BNs. In addition, this work facilitates the practical application of the existing framework to real biological systems. Importantly, the generalised definition simplifies the presentation of key definitions and results for the compositional framework.

3. **Develop a prototype support tool for attractor analysis**
   We develop algorithms for the new theoretical basis for attractor analysis (see *Contribution 1*). Initially, an algorithm was developed for an attractor identification in the composed model involving two BNs. Then, based on this algorithm we develop a new algorithm to support the identification of attractors in a composed model of multiple BNs based on the extended theoretical results. Our algorithmic approach uses a new notion of *aligned state tuples*, that is, a collection of states for the underlying BNs' SCCs, which are consistent with each other. We make a transitional step to a new aligned state tuple based on the new definition of an *interference aligned next state tuple*. We then prove that there is, at most, one interference aligned next state tuple for each aligned state tuple. We consider how to efficiently generate aligned state tuples, and maximise efficiency by ordering underlying BNs heuristically based on a new formula called *impact factor*. We attempt to generate a sequence of interference aligned next state tuples until a repeated aligned state tuple is reached, indicating that the set of interference aligned cyclic paths required to form attractors has been found. We also consider how to efficiently compute the interference aligned next state tuple to ensure that the algorithm developed is practical. We analyse the complexity of the key parts of the algorithm.

   We place a strong emphasis on ensuring that we have a support tool to automate our developed approach and assist our investigations and results. We use the developed algorithms for identifying attractors to implement a prototype support tool in Python. This tool takes BNs' state graphs in a DOT format [53] with details of composed entities and return composed model attractors. The tool development involved several iterations to improve its efficiency. Two case studies are considered to apply the techniques and tools developed. The first case study is based on a BN model of a regulatory network responsible for cell differentiation in the bacteria *Caulobacter crescentus* [5, 6]. The second case study is based on a BN model [7] for the signalling network underpinning *T cell large granular lymphocyte (T-LGL) leukemia* [54]. Although these models are small, they provide important initial insights into how we could apply our techniques to biological examples.

4. **Evaluate tools and techniques developed**
   We evaluate the developed tool using a series of performance tests by undertaking three experiments based on scalable test models. We utilise a set of nine submodels and using them to create a series of compositional tests. The first experiment looks at the general performance of the tool while the second one considers how the

performance is impacted by using models with more complex behaviour designed to stress the tool. In the third experiment, we attempt to increase the size of attractors between sizes five and 14 to observe how our tool performs. The results show that our tool performed better in the first experiment than in the second one. The third one suggests that the size of the attractors does not affect the performance of the tool. We compare the results of the first experiment to results obtained with three existing attractor identification tools *BoolNet* [55], *BNS* [35] and *BoolSim* [56]. The initial performance results were poor, and the tool underwent several improvements, as a result of the testing results. The final performance results show that our tool performs well in comparison with other tools, but the tests 16–20 indicate that further efficiency improvements must be considered in the future.

1.6 Awards, Presentations and Publications

During the research journey, the initial results of this project, along with some chapters of this thesis, have been presented in the following ways:

- The PhD Poster Presentation day for the School of Computing at Newcastle University, being awarded the prize for the best poster.
- Research work was presented at BCTCS & AlgoUK 2019 (35th British Colloquium for Theoretical Computer Science 2019 conference).
- A paper based on the work presented in Chapter 3 has been accepted to the international conference workshop BIBM 2021 Workshop, IWBNA (International Workshop on Biological Network Analysis and Integrative Graph-Based Approaches)[1].
- A journal paper based on the work presented in Chapter 4 and Chapter 5 is in preparation to be submitted to the journal *Biosystems*[2].

1.7 Thesis Structure

This thesis consists of six chapters which are organised as follows.

[1] https://ieeebibm.org/BIBM2021/
[2] https://www.sciencedirect.com/journal/biosystems

Chapter 2 Background
This chapter provides an overview of the qualitative modelling approaches. In particular, it focuses on the basic definitions for BNs, and summarises various existing tools which provide support for BNs. Following that, it explores their application in biology. Then, we discuss the importance of attractor analysis in biology, and its algorithms and techniques. Moreover, it introduces the key definitions and results of the computational framework developed at Newcastle University. Finally, we discuss related work, considering compositional and decompositional techniques for identifying attractors and for BNs.

Chapter 3 Identifying Attractors for Basic Compositions
The third chapter presents a new approach for identifying attractors in a composed model involving two BNs and the proofs of its *soundness* and *completeness*. Next, it formulates the algorithm and develops support tool for compositionally identifying attractors. Then, it illustrates the practical application of the developed tool support with a case study. Finally, it provides an overview of extending the approach into compositions of three BNs and a sequence of pairwise BNs based on the existing framework.

Chapter 4 Identifying Attractors for Generalised Compositions
This chapter introduces the general definition of a composition involving multiple BNs based on a graph structure. It presents the extended theoretical approach developed for compositionally identifying attractors based on the general definition of a composition, and this theoretical approach is illustrated by a composition example. We then prove the correctness of the approach by showing its *soundness* and *completeness*.

Chapter 5 Practical Application: Tool Support and Experimental Studies
Here we formulate an algorithm based on the theoretical results to compositionally identify attractors in an arbitrary structure of a composition. In addition, this chapter presents the algorithm for generating aligned state tuples and the idea of computing interference aligned next state tuples. Then, we discuss the implementation of the prototype tool support and its efficiency challenges. Furthermore, we illustrate the conducted experimental study, including its test models, experiments results and evaluation, and performance comparison to three existing tools. Finally, we illustrate the practical application of the developed tools and techniques using a biological case study.

Chapter 6 Concluding Remarks

The conclusion of this thesis provides a brief summary of the key results. We discuss what has been achieved, and some of the challenges we faced during this work. We review a number of interesting areas of future research that can be explored to take this work forward.

Chapter 2

Background

This chapter provides an overview of the background details required for this thesis. We start with an overview of the existing qualitative modelling approaches. Next, we introduce the background on BNs, which is the key formalism we use throughout this thesis. This section starts by introducing the BN's basic definitions. Subsequently, we summarise the existing tool support for BNs. It follows by presenting their applications in biology, in particular,the modelling of Gene Regulatory Networks (GRNs). Then, we establish the importance of attractor analysis in biology and explore their identification techniques. Following that, we introduce the background on the existing compositional framework, which we are going to use throughout this thesis. Finally, we review various works that are related to our work including compositional and decompositional techniques and attractor identification.

2.1 Qualitative Modelling Approaches

Modelling biological systems as networks provides valuable insights into the dynamical and structural properties of the modelled systems. One model used is quantitative modelling, such as the *differential equations model* [57], enabling the modelling of systems' execution precisely, but requiring a number of parameters that are not always available for many biological systems. The other approach involves qualitative modelling approaches that operate on a high-level abstraction of the system's dynamics. This relies on the interactions between systems' components, and represents them in graphs. Furthermore, the approach allows the prediction of important properties, even when some quantitative data of modelled systems are unknown [58]. Besides, it can cope with

the complex network, and simplifies the biological reality. However, it cannot be used to predict and explain the results of biological experiments that produce quantitative data. For a comprehensive study of qualitative and quantitative modelling approaches, see [59].

Qualitative modelling approaches have a wide range of techniques proposed in the literature to address the complexity of biological systems [60], such as *Boolean networks (BNs)* [15, 16, 29], *Multi-valued Networks (MVNs)* [61, 62], *Petri nets (PNs)* [20, 63, 64] and *Process Algebra (PA)* [65].

2.1.1 Boolean Networks (BNs)

BNs [15, 16, 29] represent a qualitative modelling approach that is widely used to model the interactions of biological systems. They were first introduced by Kauffman [15, 16] as a formalism for modelling natural complex systems. His work influenced Thomas [29, 66] to develop a sequential logical approach, based on Boolean formalising, for describing asynchronous logic of gene regulatory networks (GRNs). The dynamic interaction between entities is captured using next-state Boolean functions and these are applied either *synchronously* [16] (all entities update their state simultaneously) or *asynchronously* [67] (entities update their state independently).

BNs are based on using Boolean states (0 and 1) to model regulatory entities. They have been shown to be an important tool for modelling and analysing biological regulatory networks [31, 11, 68], and, in particular, attractor analysis (i.e. identifying key cyclic behaviour) can provide important insights [28]. In practice, the application of BN techniques is limited by the *state space explosion problem*, where the exponential growth of the state space severely limits the size of the model. This limitation has led to the development of compositional techniques to support BNs (for example, see [47, 36, 49]). An overview of the BN extension formalism can be be found in [69, 70]. GINsim [71, 3], BNS [35] and BoolNet [55] are examples of tools for analysing BNs in systems biology.

2.1.2 Multi-Valued Networks (MVNs)

MVNs [61, 62] are an extension of BNs, allowing the entity's state to be with in range of discrete values instead of just 0 or 1. Thus, they allow models different levels of abstraction in biological systems to represent gene expression and protein activity

levels. Furthermore, they capture a large number of interactions between biological components rather than merely activation and inhibition. For example, the active components can take several values corresponding to different levels of gene expression and protein activity, such as *high, medium and low*.

MVNs have been successfully applied to model biological systems, [70, 72, 19] and have been well studied in circuit design [61, 73]. However, they suffer from the *state space explosion problem*. Thus, in [74], an abstraction theory was developed to reduce the state space of the original model. Other studies abstract MVN models by reducing the number of regulatory entities (see for example [75, 76]). The above abstraction models preserve the underlying dynamics of the systems. GINsim [71, 3] is an example of tool support that provides an impressive array of techniques for modelling and analysing asynchronous MVN models.

2.1.3 Petri Nets (PNs)

PNs were introduced by Carl Adam Petri in 1962 [77]. The theory of PNs [20, 64] combines formal mathematical semantics and graphical representation for the modelling, design, analysis and verification of concurrent distributed systems. A PN is a bipartite graph consisting of *places* which contain *tokens* to indicate the presence of a resource, and *transitions*, which represent actions. The distribution of tokens represents the state of the PN, called a *marking*. The state space of a PN is, therefore, the set of all possible markings. Tokens can then be moved around the net by *firing transitions*. Thus, the behaviour of a system is simulated by using tokens.

PN representation of a system allows the analysing of its properties. Their behavioural properties [20], such as reachability, boundedness and liveness, are of interest when formally analysing PN models of biological systems. One of the PN structural properties based on the incidence matrix, aims to determine the invariants of the net [78, 79]. Furthermore, they are able to model a system both qualitatively and quantitatively [22]. As a consequence, they have been successfully applied in several domains, such as biological systems [80, 81], hardware design [82], cyber attacks [83, 84], and communication networks [85]. PNs have extensions formalism including *Coloured Petri nets (CPNs)* [86], *Stochastic Petri nets (SPNs)* [87] and *Time Petri nets (TPNs)* [88] which have been used to model metabolic pathways. Pathway Logic Assistant [89], Snoopy [90] and GreatSPN [91, 92] are examples of important tools for PNs used in biology.

2.1.4 Process Algebras (PAs)

PAs are formal modelling languages for concurrent computation, and have been widely used in the modelling and analysis of biological systems [93–96]. They provide tools for describing the communication and synchronisation between a collection of agents or processes which execute atomic actions. There is a wide variety of well-known process algebras, such as CCS [97], CSP [98], PCCS [99], π-calculus [100] and Beta-binders [101]. The classical ones are CCS and CSP, which are used for qualitative analysis rather than quantitative. These PAs have many extensions to capture more information about the system, such as probabilistic process algebras (e.g. PCCS) and stochastic process algebras (e.g. PEPA [102]).

In biological systems, PAs can analyse and describe a biological system, and reason about the interactions of the components. In particular, genes and proteins can be abstracted as processes, and their interactions can be modelled as actions. They have useful features that allow them to study biological systems. One such feature is that they allow the formal specification of a system. Furthermore, they support compositionality (i.e. defining the whole system from the properties of its subcomponents). In addition, they offer different levels of abstractions for a system. Their effective techniques can be used for simulation and reasoning about the possible behaviour. CADP [103] and CWB (Concurrency Workbench) [104] are examples of tools for process algebra.

The remaining part of the chapter will provide an in-depth discussion of BNs, which represent the focus of this research.

2.2 Boolean Network

Despite the conceptual simplicity of BNs, they have received much attention over the last few decades in Bioinformatics [105, 14, 8, 10, 25], circuit theory and computer science [106, 107], physics [108–110], geosciences [111, 112], social sciences [113–115] and robotics [116, 117]. In this section, we introduce the basic definitions for BNs required in the sequel, supported by illustrative examples. We then summarise a selection of existing tool support related to the Boolean dynamic of regulatory systems.

2.2.1 Basic Definitions

Boolean networks (BNs) [15, 16] represent a qualitative model that consists of a set of regulatory entities which have a Boolean value, where 1 indicates that an entity is active (e.g. a gene is expressed or a protein is present), and 0 indicates that an entity is inactive (e.g. a gene is not expressed or a protein is absent). The state of each entity is regulated by its *neighbourhood* entities (i.e. those entities which directly affect it). The behaviour of each entity is determined by applying a *next-state Boolean function* to the current state of its own neighbourhood (where an entity may or may not be a member of its neighbourhood).

A BN is a directed graph $\mathcal{BN} = (G, N, F)$, where $G = \{g_1, g_2, ..., g_n\}$ is a set of entities; $N = \{N(g_1), N(g_2), ..., N(g_n)\}$ is a finite set of neighbourhoods, where $N(g_i)$ contains the entities that regulate the behaviour of g_i; and $F = \{F(g_1), F(g_2), ..., F(g_n)\}$ represents the next-state functions for each entity, where $F(g_1) : \mathbb{B}^{|N(g_1)|} \rightarrow \mathbb{B}$, $\mathbb{B} = \{0, 1\}$.

We define a BN more formally as follows ([118, 18]).

Definition 1. *A* B*oolean Network $\mathcal{BN}$ is a tuple $\mathcal{BN} = (G, N, F)$ where:*
i) $G = \{g_1, ..., g_n\}$ is a non-empty, finite set of entities;
ii) $N = (N(g_1), ..., N(g_n))$ is a tuple of neighbourhoods, such that $N(g_i) \subseteq G$ is the neighbourhood of g_i; and
iii) $F = (F(g_1), ..., F(g_n))$ is a tuple of next-state functions, such that the function $F(g_i) : \mathbb{B}^{|N(g_i)|} \rightarrow \mathbb{B}$ defines the next state of g_i.

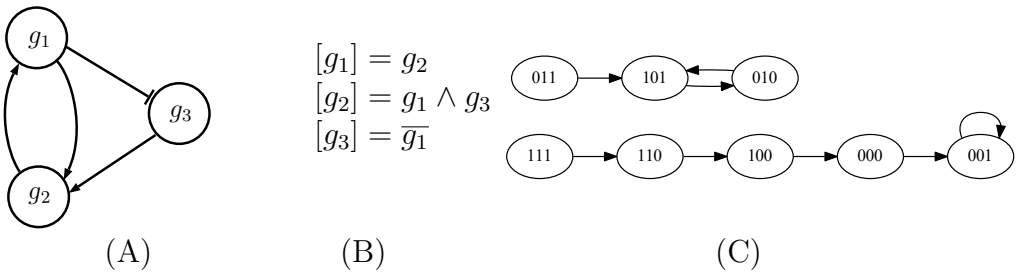


Figure 2.1 Example of a Boolean Network $\mathcal{BN}_{Ex}$ consisting of: (A) interaction graph; (B) next–state functions; and (C) state transition graph

A Boolean network $\mathcal{BN}$ consisting of n entities has a global state $(s_1 \ ... \ s_n)$, where s_i is a Boolean value representing the state of entity $g_i \in \mathcal{BN}$. The state space $S_{\mathcal{BN}} = \mathbb{B}^{|G|}$ of a Boolean network $\mathcal{BN}$ is defined to be the set of all possible global states. It is

clear that if $\mathcal{BN}$ has n entities, then $S_{\mathcal{BN}} = 2^n$. Transitions between global states are either *synchronous*, meaning that all entities update their state simultaneously, or *asynchronous*, meaning that individual entities update their own state separately [119]. While there is one synchronous updating scheme, several asynchronous updating schemes exist, such as the random order asynchronous updating scheme, the general asynchronous updating scheme and the deterministic asynchronous updating scheme [120].

We consider the synchronous update semantics here and we let $S_1 \xrightarrow{\mathcal{BN}} S_2$ represent a *(synchronous) update step* where $S_2 \in S_{\mathcal{BN}}$ is the global state that results from simultaneously updating the state of each entity g_i by applying its associated update function $F(g_i)$ to the appropriate neighbourhood of states from S_1. Note that for any global state there is only one possible next state under the synchronous update semantics.

As an example, consider a Boolean network $\mathcal{BN}_{Ex} = (G_{\mathcal{BN}_{Ex}}, N_{\mathcal{BN}_{Ex}}, F_{\mathcal{BN}_{Ex}})$ defined in Figure 2.1, which contains three entities $G_{\mathcal{BN}_{Ex}} = \{g_1,g_2,g_3\}$. Figure 2.1. (A) shows the interaction graph of the entities and directed edges that specify the neighbourhoods for activation or inhibition. For example, g_1 activates g_2, whereas g_1 inhabits g_3. The neighbourhood of $N(g_2)$ is $\{g_1, g_3\}$ and the next-state function is $[g_2] = g_1 \wedge g_3$ as represented in Figure 2.1. (B), where the notation $g_1 \vee g_2$, $g_1 \wedge g_2$, and $\overline{g_1}$ are used to illustrate the logical operators OR, AND, and NOT, respectively. Given a global state such as 101 (denoting that entity $g_1 = 0$, $g_2 = 1$, and $g_3 = 1$) we can apply the next–state functions to make a synchronous update step $101 \xrightarrow{\mathcal{BN}_{Ex}} 010$.

Given $\mathcal{S}_{\mathcal{BN}}$, a sequence of global states from some initial state in a model is called a *trace*. This sequence is infinite under a synchronous update since they must ultimately enter a cycle which we refer to as the *attractor cycle* [16, 29]. One such a trace of $\mathcal{BN}_{Ex}$ in Figure 2.1 is

$$111 \xrightarrow{\mathcal{BN}_{Ex}} 101 \xrightarrow{\mathcal{BN}_{Ex}} 010 \xrightarrow{\mathcal{BN}_{Ex}} 101 \xrightarrow{\mathcal{BN}_{Ex}} 010 \xrightarrow{\mathcal{BN}_{Ex}} 101 \xrightarrow{\mathcal{BN}_{Ex}} ...$$

which falls in the attractor $101 \xrightarrow{\mathcal{BN}_{Ex}} 010 \xrightarrow{\mathcal{BN}_{Ex}} 101 \xrightarrow{\mathcal{BN}_{Ex}} 010 \xrightarrow{\mathcal{BN}_{Ex}} ....$ We denote an attractor by using this notation

$$[101, 010, 101]$$

It should be noted that as the attractor is a cycle this attractor can be also represented by $[010, 101, 010]$. We represent a trace $\sigma(111)$ as

$$\sigma(111) = \langle 111, 101, 010, 101 \rangle$$

which specifies a trace up to the first repeated state. We let $Tr(\mathcal{BN})$ refer to a set of traces in a BN that capture all the behaviour under synchronous semantics. For example, $\mathcal{BN}_{Ex}$ (Figure 2.1) has $|\mathcal{S}_{\mathcal{BN}Ex}| = 8$ traces as the following:

$\sigma(011) = \langle 011, 101, 010, 101 \rangle$ $\quad$ $\sigma(110) = \langle 110, 100, 000, 001, 001 \rangle$
$\sigma(101) = \langle 101, 010, 101 \rangle$ $\quad$ $\sigma(100) = \langle 100, 000, 001, 001 \rangle$
$\sigma(010) = \langle 010, 101, 010 \rangle$ $\quad$ $\sigma(000) = \langle 000, 001, 001 \rangle$
$\sigma(111) = \langle 111, 101, 010, 101 \rangle$ $\quad$ $\sigma(001) = \langle 001, 001 \rangle$

Attractor cycles capture the key behaviour of a BN and are important in biological modelling, where they can be associated with biological phenomena (such as different cellular types [28]). Thus, being able to identify the attractors of a BN is an important analytic step. As mentioned above, traces must ultimately enter an attractor. There are two attractors, as depicted in the state transition graph in Figure 2.1. (C): $[010, 010]$, which is called a *point attractor* and $[101, 010, 101]$, which is called a *cyclic attractor*. The number of states presented in the cycle is called a *period*. For example, the period of this cyclic attractor $[101, 010, 101]$ is two.

A *state transition graph* $SG_{\mathcal{BN}} = (S_{\mathcal{BN}}, \xrightarrow{\mathcal{BN}})$ is a graphical presentation of BN dynamics which can be synchronous or asynchronous updates. An example of a state transition graph under synchronous semantics is given in Figure 2.1. (C) for the Boolean network $\mathcal{BN}_{Ex}$. Global states are represented by nodes and, edges denote transactions between global states. A *path* α is a set of infinite or finite steps $S_1 \xrightarrow{\mathcal{BN}} S_2 \xrightarrow{\mathcal{BN}} ....$ We use

$$\alpha = \langle S_1, S_2, S_3, \ldots \rangle$$

such that $S_i \in S_{\mathcal{BN}}$ and $S_i \xrightarrow{\mathcal{BN}} S_{i+1}$, for $i \in \mathbb{N}$, to represent an infinite steps path in $SG_{\mathcal{BN}}$. One such path in Figure 2.1 is $\langle 011, 101, 010, 101, ... \rangle$. Under synchronous semantics, such paths are deterministic and infinite. We let $Path(SG(\mathcal{BN}))$ represent the set of all such (infinite) paths. It can be seen that infinite paths and traces are equivalent under the synchronous update semantics. A path comes from a graph theory perspective, and a trace comes from the standard notion of executing the model. For

example, we have the infinite path $\langle 000, 000, ... \rangle \in Path(SG(\mathcal{BN}))$ is equivalent to the trace $\sigma(000) = \langle 000, 000 \rangle \in Tr(\mathcal{BN})$.

As mentioned above, in an asynchronous update semantic, each node updates its status independently, and there are one or more possible states for each one. Thus, multiple pathways can exist in the asynchronous dynamics, which leads to non-determinism. As an example, consider the state transition graph under an asynchronous update semantics represented in Figure 2.2. There are three possible next–states for the global state 101 by applying next–state functions: 001 if we apply $[g_1]$; 111 if we apply $[g_2]$; and 100 if we apply $[g_3]$. It assumes that at each time step, only one entity could be updated [56]. Point attractors could occur in synchronous and asynchronous update semantics [56]; however, cyclic attractors rarely exist in asynchronous models [67]. The asynchronous update semantics could reveal a *complex attractor*, which is formed by overlapping loops [121]. It can be seen that the cyclic attractor $[101, 010, 101]$ in the synchronous model of the state transition graph in Figure 2.1. (C) has been disappeared the asynchronous dynamics, as shown in Figure 2.2, but it has several point attractors such as $[000, 000]$ and $[111, 111]$.

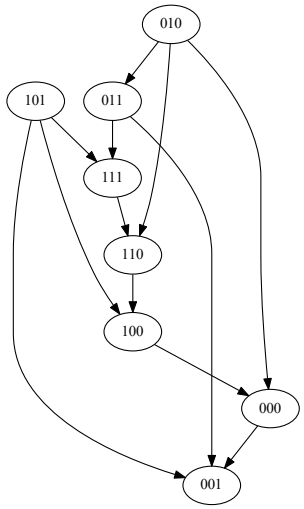


Figure 2.2 State transition graph represents the asynchronous update semantics of $\mathcal{BN}_{Ex}$ in Figure 2.1.

It has been argued that asynchronous updating is more representative of biological systems [67]. However, asynchronous updating is complex to analyse, as it captures too much behaviour that might be unrealistic in practice. In contrast, a synchronous updating has a deterministic pathway, so it is simple to model and analyse. Furthermore,

a substantial amount of research has been conducting regarding synchronous BNs (see for example [35, 36, 122, 123, 37]).

2.2.2 Existing Tool Support for BNs

BNs have proven to be useful tools to analyse and simulate regulatory networks in a qualitative manner [124]. Thus, numerous BN support tool have been produced to help scientists to gain insight into the dynamical behaviour of regulatory networks. In this section, we briefly review some of the existing software support for constructing and analysing models of biological systems related to BNs.

GinSim

GinSim [71, 125, 3] is a Java software program for the qualitative modelling, analysis and simulation of GRNs based on the logical formalism developed by Thomas and colleagues [17, 70]. It leans on both Logical Regulatory Graphs (LRG), which model regulatory networks and State Transition Graphs (STG), which represent their dynamical behaviour. GinSim is a user-friendly program that enables editing, simulating and analysis of the resulting behaviour. The graph editor allows the user to specify the regulatory graph's nodes and interactions, and to visualise it as a directed graph. For each node, the maximal level and logical parameter can be defined. Once the genes and interactions are specified, the simulation on the LRG results in an STG under both the synchronous and asynchronous update semantics, as well as priority classes. GinSim allows modification of the defined LRG in many different ways, such as reducing LRG, reversing synchronous dynamics, defining perturbations of regulatory graphs and compositing LRGs. The core of the analysis of an STG consists of attractors and the stationary states identification, and determines the strongly connected components, as well as the paths going through specific states. Static analysis for the LRG includes circuit analysis, stable search analysis, trapspace search, and highlighting the local graph. GinSim implements algorithms to decompose the complex regulatory graph into elementary circuits and analyse their functionality (see [126] for more details). It provides a feature to compose identical individual logical modules to obtain models of inter-cellular regulation [50]. GinSim allows the export of the LRG to other modelling formats (e.g. BoolSim [56], Truth table and Petri net) and allows the export of graph structures to GraphViz [127], BioLayout [128] and Cytoscape [129].

One of GinSim's important features is that it provides a large number of functions for Boolean and multi-valued logical models. In particular, to overcome the state space explosion problem, they focus on a specific part of the STG by allowing the user to choose a depth-first or a breadth-first search algorithm to explore the state transition graph by specifying the maximal depth or limit for the nodes during the search. Furthermore, an efficient algorithm computes all stable states using a Multi-valued Decision Diagram (MDD). In addition, the size of a STG can be reduced by grouping a set of states that belong to the same strongly connected components into hyper-nodes. Interestingly, to increase STG compression, they introduce another acyclic graph, called a hierarchical transition graph (HTG), which is based on merging linear chains of states (in addition to cycles) into single nodes [130]. The resulting graph conserves the important behavioural properties (e.g. attractors), but does not fully preserve reachability properties. In addition, they provide an option to reduce the regulatory graph by specifying the nodes to be reduced and recomputing their functions. Although they provide many solutions to address the state space explosion problem, their applications are limited to the size of the model.

BoolNet

BoolNet [55] is an *R* package for constructing, analysing and visualising synchronous, asynchronous and probabilistic BNs. It includes reconstructing networks from time series and natural-language statements. Its functions also include generating various types of random networks, conducting robustness analysis via perturbation and conducting Markov chain simulations. BoolNet supports several methods to identify attractors of synchronous, asynchronous and probabilistic BNs. An exhaustive search of all state-space or a heuristic search starting from several predefined or random states are methods to find synchronous attractors. To calculate potential attractor states and probabilities of reaching certain states for synchronous and probabilistic networks, Markov chain simulations are used [69].

BoolNet includes the positive feature of providing some interesting tools for attractor analysis such as providing a new random walk algorithm to identify complex asynchronous attractors. Another interesting feature of the tool is its ability to integrate well with existing modelling tools, such as BioTapestry [131] and Pajek [132]. Moreover, the transition of attractors and a network wiring diagram can be visualised. To ensure high performance for time-critical algorithms such as attractor identification, these algorithms were implemented in ANSI C, which uses bit vectors. One disadvantage of

BoolNet is that it requires programming skills. Another limitation is that this tool does not provide any compositional or decompositional support. BoolNet is accessible from [1].

BooleSim

BooleSim [133] is an open-source tool that can be run in a browser to simulate and manipulate a synchronous BN. This tool was developed in Google's Summer of Code 2012 (GSoC'12). It can be used to model and simulate GRN and signal transduction networks. In addition, it supports the importing and exporting BNs in different formats, including Python BooleanNet, R BoolNet and jSBGN formats [134]. These formats are text-based and identify the nodes and their rules. When the user imports the network, a BN graph is displayed using a force-directed graph layout algorithm from the D3.js JavaScript library [2]. The imported or constructed model can be simulated and manipulated by virtue of its on-click functionality and the visualisation of a given network's dynamic properties. BooleSim allows inline editing of the functions of the network and visualising of time series. During simulation, nodes change colour in the interface and stop when a steady state appears. Once the system enters a cyclic attractor, the simulation stops when the user clicks the simulate/pause button.

Unlike BoolNet, BooleSim is a highly interactive web-based tool, accessible online, and requires no installation, downloading or basic programming skills. However, for appropriate rendering, it does require a recent version of a browser that is compatible with HTML5. One notable drawback is that it is difficult for the user to know exactly the states involved in each attractor and it needs time to identify their sequence transitions because they are represented as a grid or as an animation. Furthermore, this online tool cannot visualise the state space. Generally, the tool is not affected when the number of entities exceeds 20. BooleSim is open-source software and exists in the GitHub repository at [3], and its online version is available at [4].

VisiBool

VisiBool [135] is a Java application for modelling, organising, simulating and visualising BNs. VisiBool introduces optional time delays for synchronous BNs in the form of

[1] https://CRAN.R-project.org/package=BoolNet
[2] (http://d3js.org/)
[3] https://github.com/matthiasbock/BooleSim
[4] https://rumo.biologie.hu-berlin.de/boolesim/

temporal predicates to model latency periods that might be more than one step between gene expression. BNs can be constructed from scratch; loaded from different options that exist in VisiBool files in XML format, BoolNet format (i.e. a text-based format) or SBML format (i.e. based on the SBML-qual package); and saved in any of those formats. VisiBool has a network panel to display a BN model as a graph representation, where its nodes represent the regulatory factors and edges represent nodes' dependencies. The network is represented as a graph in five different layouts [136–138], with regular entities as the graph's nodes. The user is allowed to add or delete nodes and update their Boolean functions via a text editor or through a tree-based graph representation. VisiBool allows the running of several simulators in parallel. The simulator is based on initial states that can be set by the user to compute attractors. VisiBool searches for attractors in synchronous models and for steady state attractors in asynchronous models for different experimental setups like knock-out experiments.

One of the tool's advantages is its capacity to be run on different operating systems, including OS X, Windows and Linux. In addition, the tool does not require the installation of a specific version of Java, because the required JRE is included in the executable file, and can interact with different BN tools because it is supported by different file formats. However, the edges cannot be moved easily, and to make a connection with other nodes, the manual function has to be entered. Furthermore, the response time for some actions (e.g. adding a new node or showing a simulation) increases as the number of nodes increases, especially when greater than 20. ViSiBooL (Java 8) is freely available at [5].

PNST

PNST (Peckham's Network Support Tool) [139] is a toolkit that supports generating and visualising BNs and multi-valued networks. Users can construct and visualise a network wiring diagram and its synchronous and asynchronous behaviour, including traces, attractors and a state transition graph. The tool applies a state clustering technique and asynchronous priority-based networks to address the state space explosion problem. It was initially developed as part of an MSc dissertation project. Since then, it has been expanded to incorporate the compositional framework results and techniques developed at Newcastle University [48, 49, 4]. PNST automates the compositional construction for two BNs and supports projection, alignments checks and compatibility checks

[5] http://sysbio.uni-ulm.de/?Software:ViSiBooL

during the composition. PNST allows the importing of JSON Based Graph-Exchange Formate (jSBGN) files, and supports exporting to Latex and GraphViz.

PNST is a user-friendly program that allows the generation, manipulation and virtualisation of networks. It provides important tools to analyse the behaviour of a BN. The user can either visualise traces and attractors in the state transition graph or see them in the form of text. In addition, attractors can be highlighted in both text and graph format. Besides, the analysis results can be exported into Latex or GraphViz. The tool allows comparison of the results of the compatibility, alignment and weak alignment by displaying the two networks compared, and through the settings can be used to display the missing behaviour or errors in the system. However, the tool performance is limited by the size of a BN, especially when the number of nodes is more than 10 for opening a BN or more than 13 nodes when merging two BNs.

Table 2.1 Summary of available tools and their features

Features/Tool Support	GinSim	BoolNet	BooleSim	VisiBool	PNST
Graphical User Interface	✓		✓	✓	✓
Synchronous updates	✓	✓	✓	✓	✓
Asynchronous updates	✓	✓		✓	✓
Construction	✓	✓	✓	✓	✓
Visualisation	✓	✓	✓	✓	✓
Attractors Identification	✓	✓	✓	✓	✓
Composition	✓				✓
Decomposition	✓				
Solutions for the state space explosion problem	✓	✓			

Summary

In summary, the reviewed tools provide all the essential required functions to construct, simulate and visualise BNs. However, GinSim produces advanced features, including the ability to compose multiple LRGs and algorithms for decomposition. Further, it addresses the state space explosion problem by compressing and specifying the search limit of the STC and using MDD. BooleSim, VisiBool and PNST are easy-to-use tools and do not require basic programming skills, unlike BoolNet. Attractors identification is one of the main features on which previous tools have focused; however, some of them use special algorithms to increase the performance in regard to finding the attractors. GinSim uses MDD to compute stable states, and BoolNet implements attractors

identification algorithms in ANSI C. Other tools such as *Genetic Network Analyser* [140], *SQUAD* [141], *CellNetAnalyzer* [142], and *Jimena* [143], have implemented algorithms for attractors identification. Table 2.1 provides a summary of the main functions of the tools described here.

2.3 Boolean Networks and Biology

Modelling approaches such as *Boolean network models* [15, 16], *multi-valued logical models* [61, 62] and *Petri nets* [20, 64] provide qualitative dynamic descriptions of system behaviours. These approaches have been widely used to model large systems such as biological networks. Among these models, BNs are very common in biology. They have been successfully applied in modelling gene regulatory and signal transduction networks [144, 145, 23, 54]. Using BNs to analyse human signalling networks associated with diseases allows prediction of therapeutic targets [145, 7]. Besides, BNs have been successfully used to identify specific regulatory interactions in gene regulatory networks [146]. In addition, Boolean models have been used in reverse engineering, which focuses on analysing and constructing biological networks (e.g. inferring regulatory interactions from gene expression [147–149]).

The BN model is among the most common models for GRNs. One of the BNs' properties is their simple nature, yet they are able to capture the complex dynamics of GRNs. The dynamics of GRNs can be described by the state transitions in the BN, and the activation and inhibition of the gene interactions are modelled as a set of Boolean functions.

2.3.1 Gene Regulatory Networks (GRNs)

Proteins are the main players in cells, consisting of chains of amino acids, and attending to various tasks essential for the survival of cells. The information that is required to produce proteins is encoded in the genome, the entirety of genes located in DNA (deoxyribonucleic acid). The process of *gene expression* to produce proteins has two stages, *transcription* and *translation*, and it is highly regulated at different levels. During the transcription stage, information from a gene is transcribed into *messenger RNA (mRNA)*, which is used to produce protein in the translation step. This process is highly regulated and can be different in a wide range, allowing the organism to adapt to external influences and environmental changes, and survive by maintaining

the metabolic processes [150]. For more details about GRNs and gene expression, see [1, 151].

GRNs describe how cells control gene expression to produce proteins that are essential for cellular function. This regulatory network consists of a set of genes and their regulators, which are connected by physical or/and regulatory interactions that control specific cell functions.

The regulation of gene expression occurs at different levels. A sample of a GRN adapted from [1] is depicted in Figure 2.3, consisting of four genes X, Y, Z_1 and Z_2. As an example of regulation, $ProteinX$ binds to an operator O, and thus has a negative influence on the transcription rates of genes X and Y. *Operator* is a DNA area with binding sites for *repressor*, a molecule that inhibits gene expression.

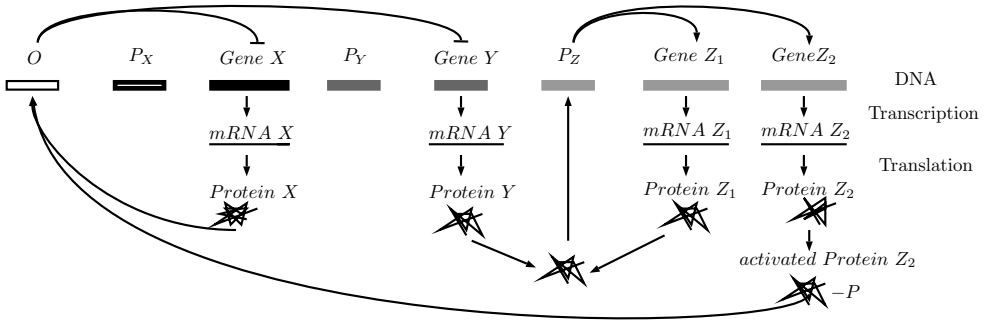


Figure 2.3 Sample regulatory network consisting of four genes X, Y, Z_1 and Z_2. Regulation of gene expression occurs at different levels, adapted from [1].

2.3.2 Modelling GRNs Using BNs

Biologists use GRNs to understand the relationships between genes that have similar phenotypes. In addition, these networks can be employed to understand how information flows in a biological system, and to model gene expression under different conditions [152].

GRNs represent as directed graphs in computer models. Models such as *Boolean networks*, *differential equation models*, *Petri nets* and *Bayesian networks* are examples of computer models that have been used in the literature to model GRNs (see [15, 16, 1, 10, 8, 153]). In the BN model, genes (nodes) are treated as binary variables; therefore, there can be only two states of expression: active or inactive (Figure 2.4 represents the BN model of the fission yeast cell cycle, in which directed edges represent positive interactions (activation), and flat edges represent negative interactions (inhibition)). A

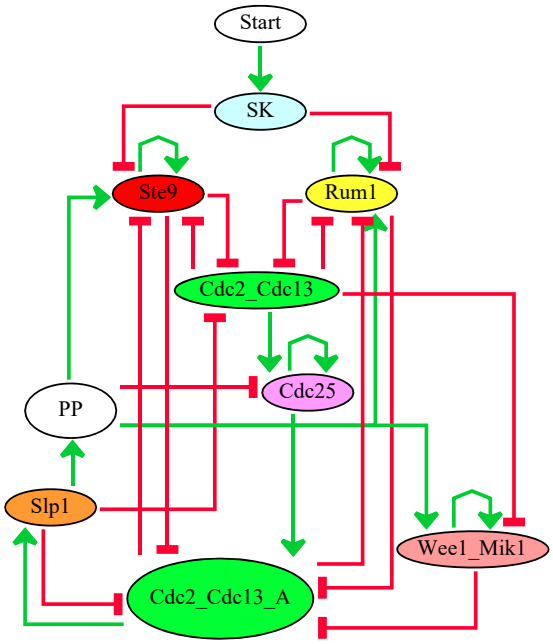


Figure 2.4 A logical model of the fission yeast cell cycle (based on [2] and transcribed using GINsim [3]).

study of the dynamical behaviour of BNs has provided insights into the dynamics of real cellular systems. Many biologically relevant phenomena can be explained by this Boolean model, such as homeostasis, multistationarity, differentiation, hysteresis and epigenesis [42, 70].

Modelling GRNs using BNs were initially proposed by Stuart Kauffman in 1969 [15], from which point it gained great attention in the literature (see [154, 16, 155–157] for reviews). For example, a BN for the *budding yeast cell cycle* has been used to identify a four node core regulatory circuit underlying cell cycle regulation [158]. Analysing the Boolean model of the *T-LGL survival signalling network* serves to identify the therapeutic targets of a disease [7]. The Boolean model of the regulation of *host immune responses* provides new insights into the virulence, pathogenesis, and host adaptation of disease-causing microorganisms [145]. The topology of the regulatory interactions in the Boolean model of the *Drosophila* segment polarity genes predicts the expression pattern [23]. As a result, BNs provide reliable results for different organisms, as illustrated by these examples.

2.3.3 Attractor Analysis

The attractors in GRNs usually correspond to the activation states of components that relate to cellular phenotypes [56, 31]. In biology, analysing attractors can help in comparing experimental data with systems dynamics. In the 1960s, Kauffman claimed that attractors can be revealed in a complex network of GRNs under specific conditions [159, 15]. In the same vein, Kauffman found that attractors correspond to the gene expression profiles for each cell type [15, 16].

Analysing attractors in BN models has received significant attention amongst biologists for several reasons. For one, such analysis can identify potential therapeutic strategies for treating cancer in a simplified protein network, which is referred to as p53 [40]. For another, attractors of the Boolean model of oestrogen transcriptional regulation represent the proliferative or antiproliferative state within cells [160]. Beyond that, attractors help in studying cell differentiation and plasticity in the CD4+ T-cell regulatory network [161]. For more examples of attractor analysis in biology, see [162, 40, 160, 161, 163, 164].

Detecting attractors in large BN models is challenging. It needs to consider 2^n states when the network has n nodes. Thus, attractor detection has been well studied in the literature. Although an approach to simulating the activation and inhibition for each initial condition has been proposed in [23, 140, 142], it may not find all possible attractors, as the initial states are chosen at random. By comparison, another technique applied to detect attractors is based on a binary decision diagram (BDD) data structure, [165, 166, 141, 56], which compactly represent the state space of the BN and support the efficient analysis of system dynamics. Even so, the size of the BDD structure is based on the Boolean operations and the order of the variables, which it is exponential to the order in the worst case. Algorithms based on a SAT-based bounded model checking support another approach to find attractors [167, 35], namely by using the SAT-solver to identify path p in the state transition graph. The algorithms unfold transitions by p times steps by generating a propositional formula and solve it by a SAT solver to identify the valid paths in the STG. In each step, a new variable in the propositional formula is used to represent a state of an entity in a BN. The process is repeated for larger number of p steps until all attractors are detected, which can increase the computation complexity owing to the large number of entities and steps. However, SAT-based model checking is more efficient than the BDD-based approach because it can detect attractors without searching the entire state space [168].

To address the state space explosion problem for large BNs that standard methods could have some limitations due to it, decompositional and compositional approaches were proposed. These approaches will be discussed in detail in the related work in Section 2.5.3.

The above shows that analysing attractors is necessary for our compositional framework, and we aim to address this area in our thesis.

2.4 Boolean Network Composition

Analysing and understanding the dynamics of large-scale biological systems is a central challenge for biologists. Consequently, the modularity advantage of biological networks has encouraged several researchers to propose compositional approaches for analysing the behaviour of the network occurring in separate network modules. Over the past decade, most research has focused on how to reveal attractors by composing the sub-networks. For instance, Guo et al. [51] proposed an algorithm to find attractors by composing the local attractors of the strongly connected components of the decomposed parts. In the same vein, [47] proposed the aggregation algorithm, which can reveal attractors with little computational cost. Many compositional approaches focus on analysing BNs [169, 51, 47]; in contrast, the formal compositional framework developed by *Alkhudhayr* and *Steggles* [48, 49, 4] supports construction models and develops results regards the preservation of behaviour (compatibility) of BNs under composition. However, as mention earlier, it has no support for attractor analysis, and its composition structure is too limited. More details about composition techniques will be discussed in related work in section 2.5.

In the following sections, we introduce the compositional framework developed by *Alkhudhayr* and *Steggles* [48, 49, 4] which was used as the basis for our thesis.

2.4.1 Key Definitions and Results

The formal compositional framework is based on composing synchronous BNs by merging entities using logical connectives, such as conjunction and disjunction. Note that there are 16 Binary Boolean operators [204] that can be chosen. However, this compositional framework restricted the Boolean operator used to be idempotent

Boolean operators (AND, OR, transfer and complement) since the idempotency is needed for the following results.

In the following we focus on using conjunction, but the results are easily applied it disjunction for example [48, 49, 4].

See Figure 2.5 for a pictorial example of the idea behind the compositional framework.

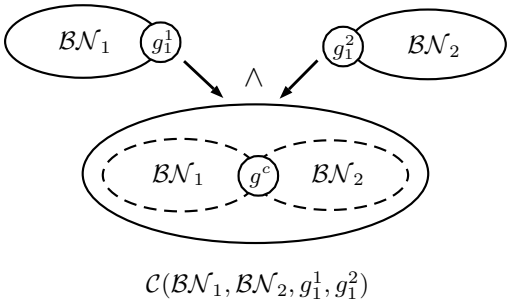


Figure 2.5 Pictorial representation of composing two Boolean networks on entities $\mathcal{BN}_1$ and $\mathcal{BN}_2$ to form a new Boolean network $\mathcal{C}$ by merging entities $g_1^1 \in \mathcal{BN}_1$ and $g_1^2 \in \mathcal{BN}_2$ into a new entity g^c (based on [4]).

In the sequel, let $\mathcal{BN}_1 = (G_1, F_1, N_1)$ and $\mathcal{BN}_2 = (G_2, F_2, N_2)$ be Boolean networks, where $G_1 = \{g_1^1, g_2^1, ..., g_n^1\}$ and $G_2 = \{g_1^2, g_2^2, ..., g_m^2\}$ are disjoint sets, for some $n, m \in \mathbb{N}$. For simplicity, we assume the entities g_1^1 and g_1^2 are always used in the composition of $\mathcal{BN}_1$ and $\mathcal{BN}_2$.

We formally define the composition of two Boolean networks $\mathcal{BN}_1$ and $\mathcal{BN}_2$ by using conjunction to merge the behaviour of entities (see Figure 2.5).

Definition 2. *(Composition) Let $\mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$ denote the Boolean network constructed by composing $\mathcal{BN}_1$ and $\mathcal{BN}_2$, which must have disjoint entities, on entities g_1^1 and g_1^2 defined as follows:*
1. Entities: *the finite set of entities $G = (G_1 \setminus \{g_1^1\}) \cup (G_2 \setminus \{g_1^2\}) \cup \{g^c\}$, where g^c denotes the new entity created by merging g_1^1 and g_1^2.*
2. Neighbourhood: *for any entity $h_i \in G$, the neighbourhood $N(h_i)$ is defined as follows:*

$$N(h_i) = \begin{cases} N_1(h_i)[g_1^1/g^c], & \text{if } h_i \in G_1 \\ N_2(h_i)[g_1^2/g^c], & \text{if } h_i \in G_2 \\ N_1(g_1^1)[g_1^1/g^c] \cup N_2(g_1^2)[g_1^2/g^c], & \text{if } h_i = g^c \end{cases}$$

where $S[f/e]$ represents set S with all occurrences of element f replaced by e.
3. Functions: *for any $h_i \in G$, the next-state function $F(h_i)$ is defined:*

$$F(h_i) = \begin{cases} F_1(h_i), & \text{if } h_i \in G_1 \\ F_2(h_i), & \text{if } h_i \in G_2 \\ \mathcal{F}, & \text{if } h_i = g^c \end{cases}$$

where $\mathcal{F} : \mathbb{B}^{|N(g^c)|} \to \mathbb{B}$ is defined using four cases as follows:
i) If $g_1^1 \notin N_1(g_1^1)$ and $g_1^2 \notin N_2(g_1^2)$, where $N_1(g_1^1) = \{l_1, \ldots, l_p\}$ and $N_2(g_1^2) = \{l'_1, \ldots, l'_q\}$, then

$$\mathcal{F}(l_1, \ldots, l_p, l'_1, \ldots, l'_q) = F_1(g_1^1)(l_1, \ldots, l_p) \wedge F_2(g_1^2)(l'_1, \ldots, l'_q);$$

ii) If $g_1^1 \in N_1(g_1^1)$ and $g_1^2 \notin N_2(g_1^2)$, where $N_1(g_1^1) = \{g_1^1, l_1, \ldots, l_p\}$ and $N_2(g_1^2) = \{l'_1, \ldots, l'_q\}$, then

$$\mathcal{F}(g^c, l_1, \ldots, l_p, l'_1, \ldots, l'_q) = F_1(g_1^1)(g^c, l_1, \ldots, l_p) \wedge F_2(g_1^2)(l'_1, \ldots, l'_q);$$

iii) If $g_1^1 \notin N_1(g_1^1)$ and $g_1^2 \in N_2(g_1^2)$, where $N_1(g_1^1) = \{l_1, \ldots, l_p\}$ and $N_2(g_1^2) = \{g_1^2, l'_1, \ldots, l'_q\}$, then

$$\mathcal{F}(g^c, l_1, \ldots, l_p, l'_1, \ldots, l'_q) = F_1(g_1^1)(l_1, \ldots, l_p) \wedge F_2(g_1^2)(g^c, l'_1, \ldots, l'_q);$$

iv) If $g_1^1 \in N_1(g_1^1)$ and $g_1^2 \in N_2(g_1^2)$, where $N_1(g_1^1) = \{g_1^1, l_1, \ldots, l_p\}$ and $N_2(g_1^2) = \{g_1^2, l'_1, \ldots, l'_q\}$, then

$$\mathcal{F}(g^c, l_1, \ldots, l_p, l'_1, \ldots, l'_q) = F_1(g_1^1)(g^c, l_1, \ldots, l_p) \wedge F_2(g_1^2)(g^c, l'_1, \ldots, l'_q).$$

Note, in finding the functions, it should be observed that if the composed entity is part of the input function, this is handled matching by updating the neighbourhood.

In the sequel, we let g^c denote the new entity created by merging g_1^1 and g_1^2. We assume that a global state $S \in \mathcal{S}_\mathcal{C}$ in the composed model $\mathcal{C} = \mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$ has the form $S = (s\ s_2^1\ \ldots\ s_n^1\ s_2^2\ \ldots\ s_m^2) \in \mathcal{S}_\mathcal{C}$, where s is the state of the new merged entity g^c and s_j^i is the state of entity g_j^i.

To illustrate the composition approach, consider composing $\mathcal{BN}_{Ex1}$ and $\mathcal{BN}_{Ex2}$ (Figure 2.6) on entities $g_1^1 \in \mathcal{BN}_{Ex1}$ and $g_1^2 \in \mathcal{BN}_{Ex2}$ represented by $\mathcal{C}(\mathcal{BN}_{Ex1}, \mathcal{BN}_{Ex2}, g_1^1, g_1^2)$ (see Figure 2.7). The new merged entity g^c is produced by composing the next-state

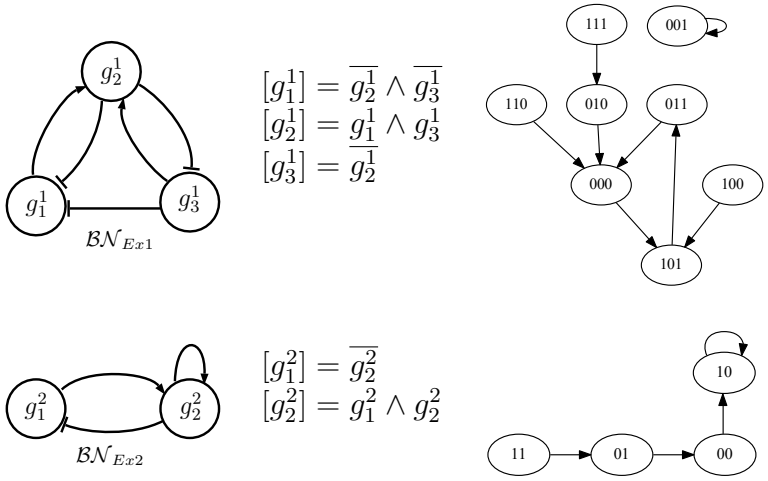


Figure 2.6 The interaction graph, next–state functions and state graphs for two example Boolean Networks $\mathcal{BN}_{Ex1}$ and $\mathcal{BN}_{Ex2}$.

functions of g_1^1 and g_1^2 using conjunction resulting in

$$[g^c] = (\overline{g_2^1} \wedge \overline{g_3^1}) \wedge \overline{g_2^2}$$

2.4.2 Behaviour Preservation

The compositional approach allows the inference of the behaviour of the composed model from the underlying BNs to address the limitations imposed by the state space explosion problem. Thus, they introduce a notion of *compatibility* which considers what it means to preserve the behaviour for each BN in the composed model. The following definitions formalise the idea of behaviour preservation under the underlying BNs.

Alkhudhayr and Steggles begin by defining a *projection operator* [49, 4], which extracts the state and traces from a composed BN.

Definition 3. *(Projections) Let* $\mathcal{C} = \mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$ *be the BN constructed by composing* $\mathcal{BN}_1$ *and* $\mathcal{BN}_2$ *(where* $G_1 = \{g_1^1, g_2^1, ..., g_n^1\}$*,* $G_2 = \{g_1^2, g_2^2, ..., g_m^2\}$*), on entities* g_1^1 *and* g_1^2*. Let* $\mathcal{S}_\mathcal{C}$ *denote the global state space for* $\mathcal{C}$*, and assume that global states have the following entity ordering* $(g^c\ g_2^1\ ...\ g_n^1\ g_2^2\ ...\ g_m^2)$*. Let* $S = (s\ s_2^1\ \ldots\ s_n^1\ s_2^2\ \ldots\ s_m^2) \in \mathcal{S}_\mathcal{C}$ *be an arbitrary global state. Then we define the projection operators*

$$\mathcal{P}_{\mathcal{BN}_1} : \mathcal{S}_\mathcal{C} \rightarrow \mathcal{S}_{\mathcal{BN}1}, \text{ and } \mathcal{P}_{\mathcal{BN}_2} : \mathcal{S}_\mathcal{C} \rightarrow \mathcal{S}_{\mathcal{BN}2}$$

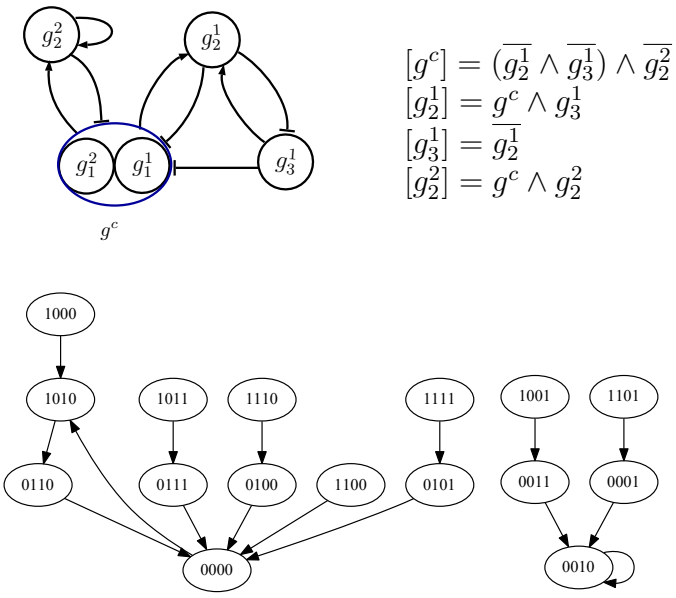


Figure 2.7 Composed model $\mathcal{C}(\mathcal{BN}_{Ex1}, \mathcal{BN}_{Ex2}, g_1^1, g_1^2)$ resulting from composing $\mathcal{BN}_{Ex1}$ with $\mathcal{BN}_{Ex2}$ on g_1^1 and g_1^2 using conjunction.

by $\mathcal{P}_{\mathcal{BN}_1}(S) = (s\ s_2^1\ \ldots\ s_n^1)$*, and* $\mathcal{P}_{\mathcal{BN}_2}(S) = (s\ s_2^2\ \ldots\ s_m^2)$ *We can extend the projection operators to traces* $\sigma = \langle S_1, S_2, ... \rangle \in Tr(\mathcal{C})$ *by*

$$\mathcal{P}_{\mathcal{BN}_1}(\sigma) = \langle \mathcal{P}_{\mathcal{BN}_1}(S_1), \mathcal{P}_{\mathcal{BN}_1}(S_2), ... \rangle, \text{ and}$$

$$\mathcal{P}_{\mathcal{BN}_2}(\sigma) = \langle \mathcal{P}_{\mathcal{BN}_2}(S_1), \mathcal{P}_{\mathcal{BN}_2}(S_2), ... \rangle$$

and let $\mathcal{P}_{\mathcal{BN}_1}(Tr(\mathcal{C}))$ *and* $\mathcal{P}_{\mathcal{BN}_2}(Tr(\mathcal{C}))$ *represent the sets of projected traces derived by projecting each trace in* $Tr(\mathcal{C})$*.*

To illustrate the idea of projection operator, consider the composed model $\mathcal{C}\,(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$, where the entities order is $(g^c, g_2^1, g_3^1, g_2^2)$ as depicted in Figure 2.7. Then,

$$\mathcal{P}_{\mathcal{BN}_{Ex1}}(\langle 0011, 0010, 0010 \rangle)) = \langle 001, 001, 001 \rangle$$

Notably, the projected trace may not be a proper trace in its corresponding BN, i.e. $Tr(\mathcal{BN}_1) \not\subseteq \mathcal{P}_{\mathcal{BN}_1}(\mathcal{C})$.

For any $\mathcal{BN}$ with entities $G = \{g_1, g_2, ...g_n\}$ and global state $S = (s_i\ ...\ s_n) \in \mathcal{S}_{\mathcal{BN}}$, they define the *projected state* on any entity $g_i \in \mathcal{BN}$ by $\mathcal{P}_{g_i}(S) = s_i$. Then $\mathcal{P}_{g_i}(\sigma)$ denotes the *projected trace* on trace $\sigma = \langle S_1, S_2, ... \rangle \in Tr(\mathcal{BN})$ defined by $\mathcal{P}_{g_i}(\sigma) = \langle \mathcal{P}_{g_i}(S_1), \mathcal{P}_{g_i}(S_2), ... \rangle$.

In [49, 4], Alkhudhayr and Steggles are interested in situations where composing two BNs preserves their underlying behaviour and denote a notion of *compatibility* to formalise this.

Definition 4. *(Compatibility) Let* $\mathcal{C} = \mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g, g')$ *be the new Boolean network constructed by composing* $\mathcal{BN}_1$ *and* $\mathcal{BN}_2$ *on entities* g *and* g'*. Then we say that* $\mathcal{BN}_1$ *and* $\mathcal{BN}_2$ *are* compatible *on* g *and* g' *iff* $Tr(\mathcal{BN}_1) \subseteq \mathcal{P}_{\mathcal{BN}_2}(Tr(\mathcal{C}))$ *and* $Tr(\mathcal{BN}_2) \subseteq \mathcal{P}_{\mathcal{BN}_1}(Tr(\mathcal{C}))$*.*

Compatibility ensures that an individual BN's behaviour is preserved under the composition (i.e. $Tr(\mathcal{BN}_1) \subseteq \mathcal{P}_{\mathcal{BN}_2}(Tr(\mathcal{C}))$). In particular, any behaviour exhibited in the individual BN must be presented in the composed model. However, the composed model could exhibit more behaviour which could not exist in the underlying BNs. The problem here is how to verify the compatibility without referencing the whole behaviour of the composed model and so it is limited by the state-space explosion problem. Therefore, they develop the notion of *alignment* which is a property that can predict whether an individual BN's behaviour is preserved under the composition by focusing on underlying BNs' behaviour only.

The *alignment* property is the exact matching of the state transitions when we map two projected traces on an entity in certain states, and is defined as follows.

Definition 5. *(Alignment) Let* $\mathcal{C} = \mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g, g')$ *be the new Boolean network constructed by composing* $\mathcal{BN}_1$ *and* $\mathcal{BN}_2$ *on entities* g *and* g'*. We say that* $\mathcal{BN}_1$ *and* $\mathcal{BN}_2$ *are aligned on* g *and* g' *iff*

$$\mathcal{P}_g(Tr(\mathcal{BN}_1)) \subseteq \mathcal{P}_{g'}(Tr(\mathcal{BN}_2))$$

In order to analyse the behaviour of the composed model they define how to merge global states and paths from the models underlying a composition (taken from [49, 4]).

Definition 6. Let $\mathcal{C} = \mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$, and $S^1 = (s_1^1 \ \ldots \ s_n^1) \in \mathcal{S}_{\mathcal{BN}1}$ and $S^2 = (s_1^2 \ \ldots \ s_m^2) \in \mathcal{S}_{\mathcal{BN}2}$. Then we define $S^1 \wedge_{g_1^1, g_1^2} S^2 \in \mathcal{S}_{\mathcal{C}}$ by merging the state of g_1^1 with g_1^2

$$S^1 \wedge_{g_1^1, g_1^2} S^2 = ((s_1^1 \wedge s_1^2) \ s_2^1 ... \ s_n^1 \ s_2^2 \ ... \ s_m^2)$$

For any paths $\alpha_1 = \langle S_1^1, S_2^1, ... \rangle \in Path(SG(\mathcal{BN}_1))$ and $\alpha_2 = \langle S_1^2, S_2^2, ... \rangle \in Path(SG(\mathcal{BN}_2))$ we define

$$\alpha_1 \wedge_{g_1^1, g_1^2} \alpha_2 = \langle S_1^1 \wedge_{g_1^1, g_1^2} S_1^2, \ S_2^1 \wedge_{g_1^1, g_1^2} S_2^2, \ldots \rangle$$

In a slight abuse of notation, we use $S_1 \wedge S_2$ and $\alpha_1 \wedge \alpha_2$ to represent $S_1 \wedge_{g_1^1, g_1^2} S_2$ and $\alpha_1 \wedge_{g_1^1, g_1^2} \alpha_2$ respectively, when the entities involved in the composition are clear from the context.

The *Alignment* property is a sufficient property to ensure the compatibility of the underlying BNs, but it is not a required condition for compatibility. Consequently, they extend the alignment property to fully characterise the compatibility. To do so, they develop an important notion, *interference*, which occurs of merged entities in the composition by extending the behaviour of the state transition graph.

2.4.3 Interference State Graph

One important notion developed in the existing work is that the interference for merged entities can be occur when two BNs are composed, resulting in new behaviour. To illustrate this, consider the global state 1001 in the composed model depicted in Figure 2.7. Then, $\mathcal{BN}_{Ex1}$ and $\mathcal{BN}_{Ex2}$ (Figure 2.6) want to make the following transitions: $100 \xrightarrow{\mathcal{BN}_{Ex1}} 101$ and $11 \xrightarrow{\mathcal{BN}_{Ex2}} 01$. The merged entities g_1^1 in $\mathcal{BN}_{Ex1}$ would like to transition from 1 to 1 and g_1^2 would like to transition from 1 to 0. The conjunction operator has been used for the composition, and therefore g^c will transition from 0 to $0 \wedge 1 = 0$. This shows that the interference behaviour of the merged entity using conjunction will occur when the next step is 1 in the underlying BN.

It is interesting to know that the interference state graph can be defined for the other Boolean operators, and this is discussed in [4]. For example, if disjunction is used instead of conjunction, then in an interference state graph, interference will occur with the underlying behaviour of a merged entity whenever it wants to transition to 1, but its merged counterpart wants to transition to 0. Note for the interference state graph to be used for the results that follow, it is important for the Boolean operator to be idempotent. It turns out that there are four idempotent Boolean operators.

To formalise the possible *interference* that can occur between composed BNs, they develop the notion of the *interference state graph* [48] by adding additional edges to a BN's state transition graph to represent interference. In a composition using conjunction, if the entity to be merged in the state transition graph of the underlying BN transitions to 1, we will add a new edge to represent the step 0 in order to capture the interference.

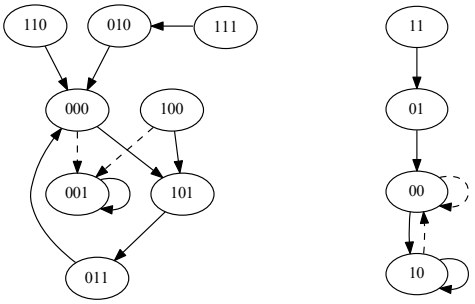
They define an *interference state graph* as follows [49, 4].

Definition 7. *(Interference State Graph) Let $\mathcal{BN}$ be a Boolean network with entities $g, g_1, \ldots, g_n$. Then we define the* interference state graph $SG_g(\mathcal{BN})$ *for $\mathcal{BN}$ on g by*

$$SG_g(\mathcal{BN}) = (\mathcal{S}_{\mathcal{BN}}, \xrightarrow[g]{\mathcal{BN}})$$

The extended edge relation $\xrightarrow[g]{\mathcal{BN}}$ is defined by $\xrightarrow[g]{\mathcal{BN}} = \xrightarrow{\mathcal{BN}} \cup \mathcal{E}$ where

$$\mathcal{E} = \{((s\ s_1\ \ldots\ s_n), (0\ s'_1\ \ldots\ s'_n)) \mid (s\ s_1\ \ldots\ s_n) \xrightarrow{\mathcal{BN}} (1\ s'_1\ \ldots\ s'_n)\}$$



$SG_{g_1^1}(\mathcal{BN}_{Ex1})$ $\qquad$ $SG_{g_1^2}(\mathcal{BN}_{Ex2})$

Figure 2.8 The interference state graphs for $\mathcal{BN}_{Ex1}$ and $\mathcal{BN}_{Ex2}$ induced by the composition $\mathcal{C}(\mathcal{BN}_{Ex1}, \mathcal{BN}_{Ex2}, g_1^1, g_1^2)$.

To illustrate this idea, consider the interference state graphs depicted in Figure 2.8 for the BNs $\mathcal{BN}_{Ex1}$ and $\mathcal{BN}_{Ex2}$ introduced above (see Figures 2.6), where dashed edges are added to the state transition graphs to represent the interference that occurs for g_1^1 and g_1^2 when we compose them using the AND operator.

A crucial result has been established that an interference state graph captures all the possible behaviours that could occur for the underlying BN in the composition, as the following result from [49] shows.

Theorem 1. *Let $\mathcal{C} = \mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$. Then we have:*
i) $Path(\mathcal{P}_{\mathcal{BN}_1}(\mathcal{C})) \subseteq Path(SG_{g_1^1}(\mathcal{BN}_1))$; and
ii)$Path(\mathcal{P}_{\mathcal{BN}_2}(\mathcal{C})) \subseteq Path(SG_{g_1^2}(\mathcal{BN}_2))$.

Recalling the compatibility definition (Definition 4), it can be seen that Alkhudhayr and Steggles consider the full behaviour of the composed model to confirm the compatibility. Therefore, they introduce a new property, denoted as *weak alignment*,

to ensure compatibility when using an idempotent Boolean operator. They prove that the *weak alignment* property is a sufficient and necessary condition for compatibility.

The weak alignment property is based on the interference state graph. They consider the interference occur in the composition to verify the alignment of the projected paths of the underlying BNs. The alignment property is the exact matching of the state transitions when we map two projected traces on an entity in certain states.

The following definition formalise the weak alignment property [49].

Definition 8. *(Weak Alignment) Let $\mathcal{BN}_1$ and $\mathcal{BN}_2$ be Boolean networks, and let $g_1^1 \in \mathcal{BN}_1$ and $g_1^2 \in \mathcal{BN}_2$. Then we say that $\mathcal{BN}_1$ and $\mathcal{BN}_2$ are weakly aligned on g_1^1 and g_1^2 iff*

$$Path(SG_{g_1^1}(\mathcal{BN}_{\mathcal{BN}_1})) \subseteq \mathcal{P}_{g_1^2}(Path(SG_{g_1^2}(\mathcal{BN}_2))), \text{ and}$$

$$Path(SG_{g_1^2}(\mathcal{BN}_{\mathcal{BN}_2})) \subseteq \mathcal{P}_{g_1^1}(Path(SG_{g_1^1}(\mathcal{BN}_1)))$$

The weak alignment property is a crucial result for the compositional framework. It allows fully characterising compatibility for underlying BNs without referring to all behaviours in the composed system. Fully characterising compatibility means it is sound and complete, where sound means when $\mathcal{BN}_1$ and $\mathcal{BN}_2$ are weakly aligned on g_1^1 and g_1^2, then they are compatible, and completeness means when $\mathcal{BN}_1$ and $\mathcal{BN}_2$ on g_1^1 and g_1^2 are compatible, then they are weakly aligned.

Theorem 2. *Let $\mathcal{BN}_1$ and $\mathcal{BN}_2$ be two Boolean networks and let $g_1^1 \in \mathcal{BN}_1$ and $g_1^2 \in \mathcal{BN}_2$. Then $\mathcal{BN}_1$ and $\mathcal{BN}_2$ are fully compatible on g_1^1 and g_1^2 iff $\mathcal{BN}_1$ and $\mathcal{BN}_2$ are fully weakly aligned on g_1^1 and g_1^2.*

Extending the above definitions and results to allow the compositions of multiple BNs over multiple entities has been investigated. However, the results produced in [4] were only partial, and several constraints had to be input due to the complexity of the approach.

2.5 Related Work

In this thesis, we focus on developing compositional techniques to facilitate the construction and analysis of the BN approach. In this section, we review several of the

related works considering compositional and decompositional techniques and attractor identification approaches. We justify research into compositional techniques by showing their importance in biology and model checking; then, we present some related works on composing BNs. We then review several decomposition techniques in biology and BNs. Finally, we review the related works on attractor analysis using compositional and decompositional approaches.

2.5.1 Compositional Techniques

Compositional techniques have been used to model and analyse biological systems [170–172, 50]. Constructing and analysing models of GRNs is difficult, as they grow in size and complexity. These large models are constructed from submodels that contain subsets of reactions within the large model. For instance, dynamical models have been built from parts such as budding yeast cells, which model the cell cycle [170] and the response to osmotic shock [173]. The importance of constructing models has inspired a number of studies to develop formal composition approaches. Randhawa et al. [174] developed a formal approach for model composition and implementing the approach to assist modellers in the composition process. Furthermore, in [172], a compositional approach allowed the analysis of the autocatalytic pathways, which are a part of core metabolism. Moreover, a novel compositional approach that depends on using process algebra framework has been developed to support the logical modelling of interconnected cellular networks [50]. Developing such composition frameworks could be useful in the field of *synthetic biology* [175–177] which aims at designing new biological systems.

Biological networks, which are an abstract representation of biological systems, have a modulator property [178, 179]. For example, proteins are known to work in slightly overlapping, coregulated groups such as pathways and complexes. The modularity allows the identification of the functionality of modules to be analysed, and then composed, to build a comprehensive model. Further, it can overcome the challenge of analysing large and complex regulatory networks. For example, the composition of the logical models of three regulatory modules of the mitotic cell cycle in budding yeast developed by Faure´ et al. [180], produces a single comprehensive model that preserves the characteristics of the original modules. In [181], the logical model was applied to compose three cross regulatory models to build the hierarchical system in *Drosophila melanogaster*.

The above two composed models were manually defined from smaller modules. Hence, a number of tools have been developed to aid composition processes. Randhawa et al. [182] developed a compositional framework and implemented the approach. They defined three operators to combine Systems Biology Markup Language (SBML) models [183], namely Fusion, Composition, Aggregation and Flattening [171]. However, they faced an issue in analysing the resulting model. The framework developed by Chaouiya et al. [184] addresses the model composition and provides systematic procedure. A prototype tool for their developed approach has been implemented, which also calculates all stable states of the composed model. However, the composed model needs to be analysed in a monolithic manner, which raises scalability issues.

Compositional approaches have been used in model checking to limit the state space explosion, and to reduce its complexity (see for example [185–188]). In [186], the compositional techniques have been used to reduce the complexity of model checking in composed systems consisting of concurrent processes. They modelled the environment of a process using another process, called the interface process, which provided the basis for their compositional model checking techniques. Besides, they could guarantee the preservation of the behaviour at the global level by checking the properties of the composition. The work in [188] provides a compositional model checking framework, with a number of state graph reduction techniques to verify complex highly concurrent systems.

A range of related work on compositional techniques developed for BNs can be found in the literature, where the main interest in this regard is analysing BN models by identifying their attractors. For example, attractors are revealed by the composition of the input-state cycles of subnetworks generated by an aggregation algorithm to reduce the computational cost [47]. Moreover, Dubrova et al. [169] showed that the attractors of the random Boolean networks can be identified compositionally from the connected components of the reduced subgraph. More compositional approaches will be discussed in Section 2.5.3.

The compositional approach of the existing compositional framework, on which our thesis is focused, is based on composing synchronous submodels by merging entities using logical connectives to support engineering and analysing the final model, and subsequently, characterising behavioural preservation using interference which appears to be different from the above approaches. However, it lacks support for attractor analysis, and its composition structure is too restricted. Thus, we focused on addressing these issues in this thesis.

2.5.2 Decompositional Techniques

Model decomposition [189–191] is an approach to breaking down a large model into submodels. Analysing and understanding large and complex models could be a challenge. More precisely, each submodel might derive all the properties and characteristics of original models, or at least part of their properties, which makes it easier to understand what happens in the entire model without referring to it. Therefore, model decomposition can be seen as a way to reduce the difficulty of gaining insight into these models, solving the problems and extracting the information. In addition, these submodels might reduce the computational cost for seeking specific knowledge, or simplifying the representation or simulation.

A wide range of studies in biology have considered decomposition approaches to simplify analysis and understand behaviours. They have been used for biochemical network models that have hundreds or thousands of variables, which increase complexity and become barriers to analysing such models (see for example [192–194]). A work in [195] focuses on the dynamic characteristics of transcriptional components to define functional submodels. A manual decomposition, induced by the underlying biological structure for autocatalytic pathways proposed to subsystems with complicating nonlinearities dynamics, is proposed in [172]. This work provides a compositional analysis framework to construct Lyapunov functions for autocatalytic pathways. The above approaches are based on an assumed network's structure. In contrast, an algorithmic decomposition approach presented in [191], which does not require prior knowledge, has been developed to identify weakly interacting submodels in a signalling network, which are called functional modules.

To date, a range of works in the literature have focused on developing decomposition approaches for BNs. A compositional approach for *Boolean automata networks (BANs)* was developed in [196], which are a generalisation of Boolean cellular automata, to aid decomposing them into parts called modules. These modules have external inputs to encode information, and can be linked to other module inputs and automata using wiring operations. To overcome the challenge of identifying attractors in large BN models, a number of studies in the literature propose a decomposition approach for BNs, which will be discussed in the following section (see for example [36, 122, 123, 51]).

In the future, we will consider developing a decompositional approach for BNs based on the theoretical results developed in this thesis (see Section 6.4).

2.5.3 Attractor Identification

Given the practical limitations imposed by the state space explosion problem, there has been a range of work assessing compositional/decompositional techniques for identifying attractors. We briefly review some relevant examples from the literature, below and relate them to our presented the approach.

A compositional approach for computing the attractors of large random BNs was considered in [169]. An approach based on identifying independent subnetworks of a BN, whose composed behaviour can infer attractors, was developed. Prior to that, they reduced the state space by removing the vertices that did not influence the network's dynamics.

An interesting attractor identification approach was developed in [47] based on decomposing a synchronous BN into a set of *Boolean control networks*. The idea is that Boolean control networks are subnetworks that contain input entities duplicated from other subnetworks. An approach for aggregating the *input-state cycles* of these control networks was used to calculate the attractors of the original BN. The approach considers using the strongly connected components (SCCs) of a BN's interaction graph to optimise the decomposition.

The above approach was further developed in [36, 122, 123] with the refinement of the SCC decomposition of a BN into blocks and the iterative recombination of blocks, to calculate attractors. Yuan et al. [123] proposed a new decomposition approach based on the SCCs suitable for large average in-degree networks. Their previous work in [122] does not consider the dependency relation among different subnetworks, resulting in many unnecessary states. However, the work in [123] considered the dependency relation among subnetworks, which can potentially improve the performance of their approach.

A similar approach for decomposing a BN into subnetworks was also developed in [51], where the underlying attractor identification technique was based on a SAT-based approach and a parallel version of their attractor identification algorithm was considered. A key issue with this approach is that it becomes inefficient for BNs with high in-degree. This issue was considered in [197], where the researchers investigated techniques for finding the smallest optimal partition for decompositionally applying a SAT solver and introducing the concept of a *minimum essential block*.

In [196] a compositional/decompositional theoretical approach for *Boolean Automata Networks (BANs)* was developed based on adding external inputs to models and then

allowing these to be wired together. Interesting initial theoretical work on developing attractor analysis techniques for this framework is presented in [198] but the complexity of the approach appears to be an issue for its practical application.

While the approaches above have basic similarities to the techniques developed here there are some significant differences. The approach of attractor identification in this thesis is based on an existing compositional framework, which focuses on the construction and analysis of BN models [4, 49], wheres the work in [36, 122, 123, 51, 47] are based on the decompositional approach for wiring diagrams. The notion of merging subnetworks in the above work does not make use of logical operators to merge entities, as the compositional framework presented in here does, and is instead based on the idea of simple input entities (see for example [47]). In particular, our work focuses on the behavioural interference generated when subnetworks are composed using the notion of an *interference state graph*. The use of SCCs is also significantly different, since we use them to identify potential cyclic behaviour in a subnetwork's behaviour, rather than as a basis for decomposing a BN's structure such as the work in [36, 122, 123, 47] (see Chapter 3 and Chapter 4 for more details regarding our new approach).

2.6 Conclusion

In this chapter, we have reviewed and discussed different background areas and related works. We provided an overview of several qualitative modelling approaches. We then introduced BNs' definitions and we briefly review some of existing tool support for BNs. Following that, we provided an overview of BNs and their applications in biology. We then introduced the compositional framework for BNs developed at Newcastle University [48, 49, 4]. We also reviewed related works on compositional and decompositional techniques and identifying attractors.

Our investigation has revealed that our approach to identifying attractors differs from others reported in the literature. The approach is based on an existing compositional framework, which focuses on the construction and analysis of BN models [49, 4]. Therefore, we are able to use the compositional structure inherent with these engineered models as the basis for our analysis techniques, and indeed, our techniques can be used to help guide the construction of a model. The development of automatic decompositional techniques for our approach is an interesting topic for further work to support the analysis of existing large-scale models. It is also important to note that the compositional attractor identification techniques and the generalised form

of a composition developed in this thesis form an important addition to the existing framework.

Chapter 3

Identifying Attractors for Basic Compositions

3.1 Introduction

In this chapter, we present initial work on extending the compositional framework presented in [48, 49, 4] with techniques and tools for compositionally identifying the attractors in a composed model. In those recent studies [48, 49, 4] (see Section 2.4), a novel compositional approach was proposed based on composing two BNs by merging entities using idempotent Boolean operators such as conjunction. This compositional framework provides a range of interesting results as a foundation for both engineering BNs and decomposing them to aid analysis. However, it focuses on the preservation of the behaviour of subnetworks in a composition. One of the areas for which it does not currently provide support is attractor identification. Given the importance of attractor analysis [28, 7], this is a key area that motivates us to extend the existing framework by developing new techniques for attractor analysis to support the practical application of the compositional framework. In the following, we focus on conjunction, but it should be noted that the results will hold if disjunction is used in a composition. This is discussed further in section 3.6.

The attractor analysis approach we have developed is based on analysing each subnetwork's *interference state graph* [49], a state graph that extends a BN's normal behaviour with the additional dynamics arising from interference in a composition. We use the *strongly connected components (SCCs)* [52] in the interference state graphs to identify cyclic behaviour, which is then selectively merged based on a new property

called *interference alignment*. We formally show that our approach is correct by proving that it is *sound* (any identified attractor is valid) and *complete* (all valid attractors are identified) for attractor identification.

We used the theoretical techniques we developed to formulate an algorithm for compositional attractor identification, and discuss how this was used to develop tool support. The idea behind the algorithm is based on finding an *interference aligned next state pair* to pair up the cyclic paths in the SCCs associated with the interference state graphs of the submodels.

The practical application of the developed techniques and tools is illustrated by applying them to a case study from the literature that is based on a regulatory network for cell differentiation [5, 6], found in the bacteria *Caulobacter crescentus* [199]. We applied the two submodels of the regulatory network to the developed tool support and the attractors were identified correctly.

This chapter is organised as follows. In Section 3.2, we extend the existing compositional framework by developing a new approach for compositionally identifying attractors and we prove that our approach is correct by showing that it is *sound* and *complete*. In Section 3.3, we consider developing tool support based on the developed algorithm to identify attractors compositionally. In Section 3.4, we illustrate our approach using a case study from the literature. In Section 3.5, we extend the results to a composition of a set of three BNs and a sequence of pairwise BNs structure. Finally, in Section 3.6, we make some concluding remarks and discuss further work.

3.2 Compositionally Identifying Attractors

In this section we develop a technique for compositionally identifying the attractors in a composed model by finding cyclic behaviour in each subnetwork's interference state graph that can be merged.

Recall that a *strongly connected component (SCC)* [52] is a maximal set of vertices in a graph such that any two vertices are mutually reachable. We define formally what we mean by an SCC for an interference state graph as follows.

Definition 9. Let $SG_g(\mathcal{BN}) = (S_{\mathcal{BN}}, \xrightarrow[g]{\mathcal{BN}})$ be an interference state graph for $\mathcal{BN}$ and let $\varphi = (\mathcal{S}_\varphi, \xrightarrow{\varphi})$ be a non-empty subgraph of $SG_g(\mathcal{BN})$ (i. e. $\mathcal{S}_\varphi \subseteq S_{\mathcal{BN}}$ is a non-empty set of global states and $\xrightarrow{\varphi} \subseteq \xrightarrow[g]{\mathcal{BN}}$ is a non-empty edge relation). Then φ is

a *Strongly Connected Component (SCC)* for $SG_g(\mathcal{BN})$ iff the following holds: i) for any two states $S, T \in \mathcal{S}_\varphi$ there is a directed path from S to T; and ii) φ is maximal (adding any further nodes or edges from $SG_g(\mathcal{BN})$ to φ breaks the above connectivity property).

We let $SCC(SG_g(\mathcal{BN}))$ denote the set of all SCCs for an interference state graph $SG_g(\mathcal{BN})$.

As an example, consider the SCCs depicted in Figure 3.1 for the interference state graphs $SG_{g_1^1}(\mathcal{BN}_{Ex1})$ and $SG_{g_1^2}(\mathcal{BN}_{Ex2})$.

We use the SCCs in a subnetwork's interference state graph to identify cyclic behaviour that can be merged to generate attractors in the composed model. Let $\varphi \in SCC(SG_g(\mathcal{BN}))$ and let α be an infinite path over φ. Then we say that α is a *cyclic path* iff there exists $k \in \mathbb{N}$ and $S_1, \ldots, S_k \in \mathcal{S}_\varphi$ such that

$$\alpha = \langle S_1, \ldots, S_k, S_1, \ldots, S_k, S_1, \ldots, S_k, \ldots \rangle$$

We let $CPaths(\varphi)$ denote the set of all cyclic paths for an SCC φ.

The cyclic paths that are associated with SCCs represent a basis for potential attractors generated in a composed model. Using a new property called *interference alignment* to check when cyclic paths can be merged, we merge the cyclic paths generated from the SCCs for the subnetworks to identify attractors.

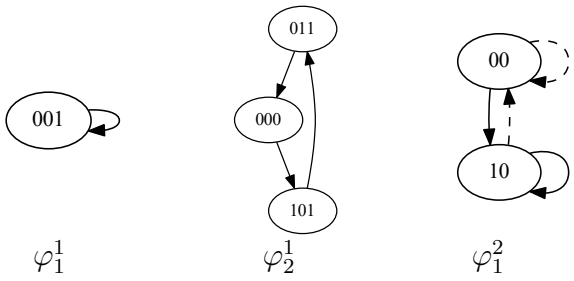


Figure 3.1 The SCCs for the composition $\mathcal{C}(\mathcal{BN}_{Ex1}, \mathcal{BN}_{Ex2}, g_1^1, g_1^2)$, where φ_1^1 and φ_2^1 are from $SG_{g_1^1}(\mathcal{BN}_{Ex1})$, φ_1^2 is from $SG_{g_1^2}(\mathcal{BN}_{Ex2})$.

Before we can define the interference alignment property, we need the following notions. Given two states $S_1, S_2 \in S_{\mathcal{BN}}$, we refer to a state transition step $S_1 \xrightarrow[g]{\mathcal{BN}} S_2$ as a *normal step* iff $S_1 \xrightarrow{\mathcal{BN}} S_2$ (i.e. it does not require interference). A step $S_1 \xrightarrow[g]{\mathcal{BN}} S_2$ is referred to as an *interference step* iff it is not a normal step (i.e. it does require interference). For a Boolean network $\mathcal{BN}$ We define the entity state projection

$\mathcal{P}_{g_i}(S) = s_i$, for any entity g_i in $\mathcal{BN}$ and $S = (s_1, \ldots, s_n) \in S_{\mathcal{BN}}$. The projection operators can be lifted to a path $\alpha \in Path(SG(\mathcal{BN}))$ and a set of paths in the standard way.

Definition 10. Let $\alpha_1 = \langle S_1, S_2, ... \rangle \in Path(SG_{g_1^1}(\mathcal{BN}_1))$ and $\alpha_2 = \langle T_1, T_2, ... \rangle \in Path(SG_{g_1^2}(\mathcal{BN}_2))$. Then we say that α_1 and α_2 *interference align* iff $\mathcal{P}_{g_1^1}(\alpha_1) = \mathcal{P}_{g_1^2}(\alpha_2)$ and for any $i \in \mathbb{N}$, we have $S_i \xrightarrow[g_1^1]{\mathcal{BN}_1} S_{i+1}$ and $T_i \xrightarrow[g_1^2]{\mathcal{BN}_2} T_{i+1}$ are not both interference steps.

Interference alignment captures when paths can be merged to create a path in the composed model by checking to ensure that interference actually occurs at the points required in each path. This is formally shown by the following result.

Lemma 3. *Let $\alpha_1 \in Path(SG_{g_1^1}(\mathcal{BN}_1))$ and $\alpha_2 \in Path(SG_{g_1^2}(\mathcal{BN}_2))$. Then if α_1 and α_2 interference align then*

$$\alpha_1 \wedge \alpha_2 \in Path(SG(\mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2))$$

Proof. Let $\mathcal{C} = \mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$. Let $\alpha_1 = \langle S_1, S_2, ... \rangle \in Path(SG_{g_1^1}(\mathcal{BN}_1))$ and $\alpha_2 = \langle T_1, T_2, ... \rangle \in Path(SG_{g_1^2}(\mathcal{BN}_2))$ such that α_1 and α_2 interference align. To show $\alpha_1 \wedge \alpha_2 \in Path(SG(\mathcal{C}))$ it suffices to show that for any $i \in \mathbb{N}$

$$S_i \wedge T_i \xrightarrow{\mathcal{C}} S_{i+1} \wedge T_{i+1} \tag{3.1}$$

For any $i \in \mathbb{N}$ we have $S_i \xrightarrow[g_1^1]{\mathcal{BN}_1} S_{i+1}$ and $T_i \xrightarrow[g_1^2]{\mathcal{BN}_2} T_{i+1}$, where $S_i = (s_1^i \ \ldots \ s_n^i)$, $S_{i+1} = (s_1^{i+1} \ \ldots \ s_n^{i+1})$, $T_i = (t_1^i \ \ldots \ t_m^i)$ and $T_{i+1} = (t_1^{i+1} \ldots t_m^{i+1})$. Then by our assumption and definition of merging states we have

$$S_i \wedge T_i = (s_1^i \wedge t_1^i \ s_2^i \ \ldots \ s_n^i \ {t_2}^i \ \ldots \ t_m^i)$$

$$S_{i+1} \wedge T_{i+1} = (s_1^{i+1} \wedge t_1^{i+1} \ s_2^{i+1} \ \ldots \ s_n^{i+1} \ {t_2}^{i+1} \ \ldots \ t_m^{i+1})$$

By the definition of interference alignment we know that $s_1^i = t_1^i$ and so by idempotency of $\wedge$ we have

$$s_1^i \wedge t_1^i = s_1^i = t_1^i \tag{3.2}$$

Given the assumption of interference alignment, there are three cases in which we must consider focusing on where the interference can occur (note that the update steps cannot both be interference steps by the definition of interference alignment).

Case 1: Suppose $S_i \xrightarrow[g_1^1]{\mathcal{BN}_1} S_{i+1}$ is an interference step and $T_i \xrightarrow[g_1^2]{\mathcal{BN}_2} T_{i+1}$ is a normal step. Then by the above assumptions and the definition of $SG_{g_1^1}(\mathcal{BN}_1)$ we know $s_1^{i+1} = 0$ and

$$S_i \xrightarrow{\mathcal{BN}_1} (1\ s_2^{i+1} \dots s_n^{i+1})$$

Then by (3.2) and the definition of $\mathcal{C}$ we know that

$$(s_1^i \wedge t_1^i\ s_2^i \dots s_n^i\ {t_2}^i\ \dots\ t_m^i) \xrightarrow{\mathcal{C}} (1 \wedge t_1^{i+1}\ s_2^{i+1} \dots\ s_n^{i+1}\ t_2^{i+1}\ \dots\ t_m^{i+1})$$

To prove (3.1) we have to show $s_1^{i+1} \wedge t_1^{i+1} = 1 \wedge t_1^{i+1}$. This follows by the definition of conjunction since we know $s_1^{i+1} = 0$ and by interference alignment we have $s_1^{i+1} = t_1^{i+1}$.

Case 2: Suppose $S_i \xrightarrow[g_1^1]{\mathcal{BN}_1} S_{i+1}$ is a normal step and $T_i \xrightarrow[g_1^2]{\mathcal{BN}_2} T_{i+1}$ is an interference step. Then this follows along similar lines to Case 1 above.

Case 3: Suppose $S_i \xrightarrow[g_1^1]{\mathcal{BN}_1} (s_1^{i+1} \dots\ s_n^{i+1})$ and $T_i \xrightarrow[g_1^2]{\mathcal{BN}_2} (t_1^{i+1} \dots t_m^{i+1})$ are both normal steps. Then by (3.2) and definition of $\mathcal{C}$ we have

$$(s_1^i \wedge t_1^i\ s_2^i\ \dots\ s_n^i\ {t_2}^i \dots t_m^i) \xrightarrow{\mathcal{C}} (s_1^{i+1} \wedge t_1^{i+1}\ s_2^{i+1}\ \dots\ s_n^{i+1}\ {t_2}^{i+1}\ \dots\ t_m^{i+1})$$

and so (3.1) holds as required. □

We now have a basis for compositionally identifying attractors. Suppose we have two SCCs, $\varphi_1 \in SCC(SG_{g_1^1}(\mathcal{BN}_1))$ and $\varphi_2 \in SCC(SG_{g_1^2}(\mathcal{BN}_2))$, and cyclic paths $\alpha_1 \in CPaths(\varphi_1)$ and $\alpha_2 \in CPaths(\varphi_2)$. Then if these cyclic paths interference align then they can be merged $\alpha_1 \wedge \alpha_2$ producing a path that must represent an attractor in the composed model $\mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1, g_2)$.

To illustrate this idea consider the SCCs φ_1^1 and φ_2^2 for $SG_{g_1^1}(\mathcal{BN}_{Ex1})$ and $SG_{g_1^2}(\mathcal{BN}_{Ex2})$ (see Figure 3.1) and the cyclic paths $\langle 011, 000, 101, 011, 000, 101, 011, \dots\rangle \in CPaths(\varphi_2^1)$ and $\langle 00, 00, 10, 00, 00, 10, 00 \dots\rangle \in CPaths(\varphi_1^2)$. These two cyclic paths can be seen to interference align, and so by Lemma 3, we know that the path that results from merging them

$$\langle 0110, 0000, 1010, 0110, 0000, 1010, 0110 \dots\rangle$$

is in the composed model. It follows that $[0110, 0000, 1010, 0110]$ is an attractor for the composed model $\mathcal{C}(\mathcal{BN}_{Ex1}, \mathcal{BN}_{Ex2}, g_1^1, g_1^2)$.

We formally prove that our approach for compositionally identifying attractors is correct by showing that it is *sound* (any attractor found is valid) and *complete* (all valid attractors are found).

Theorem 4. *(Soundness) Let* $\varphi_1 \in SCC(SG_{g_1^1}(\mathcal{BN}_1))$ *and* $\varphi_2 \in SCC(SG_{g_1^2}(\mathcal{BN}_2))$ *be SCCs. Let* $\alpha_1 \in CPaths(\varphi_1)$ *and* $\alpha_2 \in CPaths(\varphi_2)$ *be cyclic paths such that* α_1 *and* α_2 *interference align. Then* $\alpha_1 \wedge \alpha_2$ *represents an attractor in the composed model* $\mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1, g_2)$.

Proof. Let $\varphi_1 \in SCC(SG_{g_1^1}(\mathcal{BN}_1))$ and $\varphi_2 \in SCC(SG_{g_1^2}(\mathcal{BN}_2))$ be SCCs. Let $\alpha_1 \in CPaths(\varphi_1)$ and $\alpha_2 \in CPaths(\varphi_2)$ be cyclic paths such that α_1 and α_2 interference align. By above assumptions and Lemma 3 it follows

$$\alpha_1 \wedge \alpha_2 \in Path(SG(\mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)))$$

By the definition of cyclic paths we know there must exist minimal $k_1, k_2 \in \mathbb{N}$ and states $S_1, \ldots, S_{k_1} \in S_{\mathcal{BN}_1}$ and $T_1, \ldots, T_{k_2} \in S_{\mathcal{BN}_2}$ such that

$$\alpha_1 = \langle S_1, \ldots, S_{k_1}, S_1, \ldots, S_{k_1}, \ldots \rangle, \quad \alpha_2 = \langle T_1, \ldots, T_{k_2}, T_1, \ldots, T_{k_2}, \ldots \rangle$$

Let $LCM(k_1, k_2)$ represent the lowest common multiple of k_1 and k_2. Then it follows that the first $LCM(k_1, k_2) + 1$ states of $\alpha_1 \wedge \alpha_2$ must represent an attractor in $\mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$. □

We now consider showing that the proposed approach is complete, and begin with some necessary preliminary results about projecting and merging paths in a composed model.

Lemma 5. *Let* $\beta \in Path(SG(\mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)))$. *Then*

$$\mathcal{P}_{\mathcal{BN}_1}(\beta) \wedge \mathcal{P}_{\mathcal{BN}_2}(\beta) = \beta$$

Proof. Let $\mathcal{C} = \mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$. Let $\beta = \langle F_1, F_2, ... \rangle \in Path(SG(\mathcal{C}))$.

It is sufficient to show that, for any $i \in \mathbb{N}$

$$F_i = \mathcal{P}_{\mathcal{BN}_1}(F_i) \wedge \mathcal{P}_{\mathcal{BN}_2}(F_i)$$

Let $i \in \mathbb{N}$ and let $F_i = (d\ s_2\ ...\ s_n\ t_2 ...\ t_m)$, where d is the state of the merged entity $g_c = g_1^1 \wedge g_1^2$. Then $\mathcal{P}_{\mathcal{BN}_1}(F_i) = (d\ s_2\ ...s_n)$ and $\mathcal{P}_{\mathcal{BN}_2}(F_i) = (d\ t_2\ ...\ t_m)$.

So by idempotency of $\wedge$, we know that $d \wedge d = d$ as required. □

We now show that any path in the composed model projected over the entities of an underlying BN will produce a path in the interference state graph of that underlying BN.

Lemma 6. *Let* $\beta \in Path(SG(\mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)))$. *Then we must have:*
i) $\mathcal{P}_{\mathcal{BN}_1}(\beta) \in Path(SG_{g_1^1}(\mathcal{BN}_1))$
ii) $\mathcal{P}_{\mathcal{BN}_2}(\beta) \in Path(SG_{g_1^2}(\mathcal{BN}_2))$

Proof. i) Let $\mathcal{C} = \mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$. Let $\beta = \langle F_1, F_2, ... \rangle \in Path(SG(\mathcal{C}))$. Then, we need to show that $\mathcal{P}_{\mathcal{BN}_1}(\beta) \in Path(SG_{g_1^1}(\mathcal{BN}_1))$. It suffices to show that for any $i \in \mathbb{N}$ there exists a step

$$\mathcal{P}_{\mathcal{BN}_1}(F_i) \xrightarrow[g_1^1]{\mathcal{BN}_1} \mathcal{P}_{\mathcal{BN}_1}(F_{i+1}) \tag{1}$$

in the interference state graph $SG_{g_1^1}(\mathcal{BN}_1)$. Let $i \in \mathbb{N}$ and let

$$F_i = (d^i \ s_2^i \ ... \ s_n^i \ t_2^i \ ... \ t_m^i) \in \mathcal{S}_{\mathcal{C}}$$

where d^i is the state of the merged entity $g_c = g_1^1 \wedge g_1^2$. By the definition of projection we know that

$$\mathcal{P}_{\mathcal{BN}_1}(F_i) = (d^i \ s_2^i \ ... \ s_n^i) \text{ and } \mathcal{P}_{\mathcal{BN}_2}(F_i) = (d^i \ t_2^i \ ... \ t_m^i)$$

Suppose that

$$(d^i \ s_2^i \ ... \ s_n^i) \xrightarrow{\mathcal{BN}_1} (s_1^{i+1} \ ... \ s_n^{i+1}) \tag{2}$$

$$(d^i \ t_2^i \ ... \ t_m^i) \xrightarrow{\mathcal{BN}_2} (t_1^{i+1} \ ... \ t_m^{i+1}) \tag{3}$$

Then, by the definition of composition we have

$$F_{i+1} = (s_1^{i+1} \wedge t_1^{i+1} \ s_2^{i+1} \ \ldots \ s_n^{i+1} \ {t_2}^{i+1} \ \ldots \ t_m^{i+1})$$

To show (1) holds, we therefore need to show

$$(d^i \ s_2^i \ ... \ s_n^i) \xrightarrow[g_1^1]{\mathcal{BN}_1} (s_1^{i+1} \wedge t_1^{i+1} \ s_2^{i+1} \ ... \ s_n^{i+1}) \tag{4}$$

We have two cases to consider with respect to $s_1^{i+1} \wedge t_1^{i+1}$.

Case 1: Suppose there is no interference between s_1^{i+1} and t_1^{i+1}. Then we must have $s_1^{i+1} \wedge t_1^{i+1} = s_1^{i+1} = t_1^{i+1}$. It therefore follows from (2) and definition of $\wedge$ that (4) holds.

Case 2: Suppose there is interference between s_1^{i+1} and t_1^{i+1}. Then, there are two sub-cases to consider based on which BN generated the interference.

Case 2.1: Suppose $s_1^{i+1} = 1$ and $t_1^{i+1} = 0$, then by definition of $\wedge$ we must have $s_1^{i+1} \wedge t_1^{i+1} = 0 = \neg s_1^{i+1}$ (i.e. $\mathcal{BN}_2$ interfered with $\mathcal{BN}_1$). Then by definition of projection we have

$$\mathcal{P}_{\mathcal{BN}_1}(F_{i+1}) = (\neg s_1^{i+1}\ s_2^{i+1}\ ...\ s_n^{i+1})$$

By definition of the interference state graph, we must have the interference edge

$$(d^i\ s_2^i\ ...\ s_n^i) \xrightarrow[g_1^1]{\mathcal{BN}_1} (\neg s_1^{i+1}\ s_2^{i+1}\ ...\ s_n^{i+1})$$

in $SG_{g_1^1}(\mathcal{BN}_1)$ as required.

Case 2.2: Suppose $s_1^{i+1} = 0$ and $t_1^{i+1} = 1$, then by definition of $\wedge$ we must have $s_1^{i+1} \wedge t_1^{i+1} = s_1^{i+1}$ (i.e. $\mathcal{BN}_1$ interfered with $\mathcal{BN}_2$). Then by definition of projection we have

$$\mathcal{P}_{\mathcal{BN}_1}(F_{i+1}) = (s_1^{i+1}\ ...\ s_n^{i+1})$$

Then we must have the normal edge

$$(d^i\ s_2^i\ ...\ s_n^i) \xrightarrow{\mathcal{BN}_1} (s_1^{i+1}\ ...\ s_n^{i+1})$$

in $SG_{g_1^1}(\mathcal{BN}_1)$ as required.

ii) Follows along similar lines to i) above. □

We can now prove the proposed compositional attractor identification approach is *complete*.

Theorem 7. *(Completeness) Let $\alpha = [F_1, F_2, ...F_n, F_1]$ be an attractor in $\mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$. Then there must exist $\varphi_1 \in SCC(SG_{g_1^1}(\mathcal{BN}_1))$ and $\varphi_2 \in SCC(SG_{g_1^2}(\mathcal{BN}_2))$, and cyclic paths $\alpha_1 \in CPaths(\varphi_1)$ and $\alpha_2 \in CPaths(\varphi_2)$ such that α_1 and α_2 interference align, and $\alpha_1 \wedge \alpha_2$ results in the attractor α.*

Proof. Let $\alpha = [F_1, F_2, \ldots F_n, F_1]$ be an arbitrary attractor in the composed model $\mathcal{C}(\mathcal{BN}_1, \mathcal{BN}_2, g_1^1, g_1^2)$. Then α can be viewed as representing the infinite path $\langle F_1, \ldots F_n, F_1, \ldots F_n, \ldots\rangle$. Let $\alpha_1 = \mathcal{P}_{\mathcal{BN}_1}(\alpha)$ and $\alpha_2 = \mathcal{P}_{\mathcal{BN}_2}(\alpha)$. Let $\alpha_1 = \langle S_1, ..., S_n, S_1, ..., S_n, ...\rangle$ and $\alpha_2 = \langle T_1, ..., T_n, T_1, ..., T_n, ...\rangle$.

First, by Lemma 5, it follows that $\alpha_1 \wedge \alpha_2$ must result in α.

Next, we need to consider showing that α_1 and α_2 interference align. By definition they must align, so it suffices to show that each corresponding step $S_i \to (s_1^{i+1} \ \ldots \ s_k^{i+1})$ and $T_i \to (t_1^{i+1} \ \ldots \ t_m^{i+1})$ cannot both be interference steps. This clearly must be true by the definition of composition because otherwise we would have $S_i \xrightarrow{\mathcal{BN}_1} (\neg s_1^{i+1} \ s_2^{i+1} \ \ldots \ s_k^{i+1})$ and $T_i \xrightarrow{\mathcal{BN}_2} (\neg t_1^{i+1} \ t_2^{i+1} \ \ldots \ t_m^{i+1})$, resulting in a contradiction, since $\neg s_1^{i+1} \wedge \neg t_1^{i+1} \neq s_1^{i+1} \wedge t_1^{i+1}$.

Finally, we need to show that α_1 and α_2 are cyclic paths resulting from an SCC in $SCC(SG_{g_1^1}(\mathcal{BN}_1))$ and $SCC(SG_{g_1^2}(\mathcal{BN}_2))$, respectively. It follows by Lemma 12 that $\alpha_1 \in Path(SG_{g_1^1}(\mathcal{BN}_1))$ and $\alpha_2 \in Path(SG_{g_1^2}(\mathcal{BN}_2))$. Furthermore, since α_1 and α_2 are clearly cyclic paths, it can be seen that they must result from an SCC structure in their corresponding interference state graphs as required. □

3.3 Developing Tool Support

In this section, we use the theoretical approach outlined in the previous section as the basis to develop tool support for compositionally identifying attractors in a composed model consisting of two BNs.

The theoretical approach introduced above is based on identifying two cyclic paths over SCCs in the submodels' interference state graphs, which are interference aligned and therefore can be merged. We are interested here in developing an algorithm for finding such cyclic paths. Our algorithmic approach starts from an aligned state pair (i.e. they have the same value on the entities to be merged) over an SCC in each submodel. Then, we transition from the state pair to the next one until a repeated state pair occurs, indicating that cyclic paths have been formed. When making a transition to a new state pair, we ensure that the two states align and are not both interference steps; we refer to this as an *interference aligned next state pair*.

We formally define an *interference aligned next state pair* as follows.

Definition 11. Let $S_1, S_2 \in S_{\mathcal{BN}_1}$ and $T_1, T_2 \in S_{\mathcal{BN}_2}$ such that $S_1 \xrightarrow[g_1^1]{\mathcal{BN}_1} S_2$ and $T_1 \xrightarrow[g_1^2]{\mathcal{BN}_2} T_2$. Then we say (S_2, T_2) is an *interference aligned next state pair* for (S_1, T_1), iff $\mathcal{P}_{g_1^1}(S_2) = \mathcal{P}_{g_1^2}(T_2)$ and at least one of the state transitions $S_1 \xrightarrow[g_1^1]{\mathcal{BN}_1} S_2$ and $T_1 \xrightarrow[g_1^2]{\mathcal{BN}_2} T_2$ is a normal step.

Note that for any state pair (S_1, T_1), there is at most one interference aligned next state pair.

Now, we formulate an algorithm for identifying and merging cyclic paths from the given SCCs for the subnetworks. The pseudocode given below outlines **Algorithm 1** *findAttTwo* which given two SCCs $\varphi_1 \in SCC(SG_{g_1^1}(\mathcal{BN}_1))$ and $\varphi_2 \in SCC(SG_{g_1^2}(\mathcal{BN}_2))$ finds all the attractors associated with these cyclic structures. The algorithm makes use of the following auxiliary functions:

- $alignSet(\varphi_1, \varphi_2)$ returns the set of all aligned state pairs for the given SCCs, and is defined by

$$alignSet(\varphi_1, \varphi_2) =$$

$$\{(S, T) \mid S \in \mathcal{S}_{\varphi_1}, T \in \mathcal{S}_{\varphi_2},\ \mathcal{P}_{g_1^1}(S) = \mathcal{P}_{g_1^2}(T)\};$$

- $noStep((S, T), \varphi_1, \varphi_2)$ is a Boolean function that checks if a state pair (S, T) does not have an interference aligned next state pair with respect to φ_1 and φ_2;
- $doStep((S, T), \varphi_1, \varphi_2)$ returns, if it exists, the interference aligned next state pair for (S, T) with respect to φ_1 and φ_2;
- $extCP(listSP)$ returns the attractor that $listSP$ must end with.

The idea is to apply $findAttTwo(\varphi_1, \varphi_2)$ to each possible pair of SCCs $\varphi_1 \in SCC(SG_{g_1^1}(\mathcal{BN}_1))$ and $\varphi_2 \in SCC(SG_{g_1^2}(\mathcal{BN}_2))$. There are standard algorithms based on a depth-first search for finding SCCs, such as *Tarjan's algorithm* [200], which runs in linear time $O(|V| + |E|)$ for a graph (V, E). The overall performance of the proposed algorithmic approach depends on the number of SCCs and their states. Assuming that the number of combined states for the SCCs in any subnetwork is bounded by k gives an upperbound of k^2 state pairs to consider.

To illustrate how the algorithm works, consider applying it to the composition $\mathcal{C}(\mathcal{BN}_{Ex1}, \mathcal{BN}_{Ex2}, g_1^1, g_1^2)$ (Figure 2.7). In this example, we have two SCCs φ_1^1 and φ_2^1 for $\mathcal{BN}_{Ex1}$, and one SCC φ_1^2 for $\mathcal{BN}_{Ex2}$ (see Figure 3.1). We therefore have $2 \times 1 = 2$

Algorithm 1: *findAttTwo*(φ_1, φ_2)

Input : $\varphi_1 \in SCC(I_1); \varphi_2 \in SCC(I_2)$
Output : $attSet : Set\ of\ Attractors$
Variables : $S_1, S_2 \in \mathcal{S}_{\varphi_1}; T_1, T_2 \in \mathcal{S}_{\varphi_2}$; $listSP : List\ of\ State\ Pairs\ \mathcal{S}_{\varphi_1} \times \mathcal{S}_{\varphi_2}$; $seen : Set\ of\ State\ Pairs\ \mathcal{S}_{\varphi_1} \times \mathcal{S}_{\varphi_2}$

```
1  Begin
2    attSet := {}; seen := {}
3    foreach (S1, T1) ∈ alignSet(φ1, φ2) do
4      listSP := [ ]
5      Loop
6        if (S1, T1) ∈ seen then
7          Exit Loop
8        else
9          seen := seen ∪ {(S1, T1)}
10         if noStep((S1, T1), φ1, φ2) then
11           Exit Loop
12         else
13           listSP := listSP + +[(S1, T1)]
14           (S2, T2) := doStep((S1, T1), φ1, φ2)
15           if (S2, T2) ∈ listSP then
16             attSet := attSet ∪ {extCP(listSP + +[(S2, T2)])}
17             Exit Loop
18           else
19             (S1, T1) := (S2, T2)
20           end
21         end
22       end
23     EndLoop
24   end
25   return attSet
26 End
```

possible pairs of SCCs $(\varphi_1^1, \varphi_1^2)$ and $(\varphi_2^1, \varphi_1^2)$ to apply *findAttTwo* to.

findAttTwo$(\varphi_2^1, \varphi_1^2)$: We have $|(\mathcal{S}_{\varphi_2^1} \times \mathcal{S}_{\varphi_1^2})| = 6$ possible state pairs but there are only three state pairs that align

$$alignSet(\varphi_2^1, \varphi_1^2) = \{(011, 00), (000, 00), (101, 10)\}$$

The following is a trace for *findAttTwo*$(\varphi_2^1, \varphi_1^2)$:
1) Suppose *findAttTwo* processes $(011, 00)$ first.

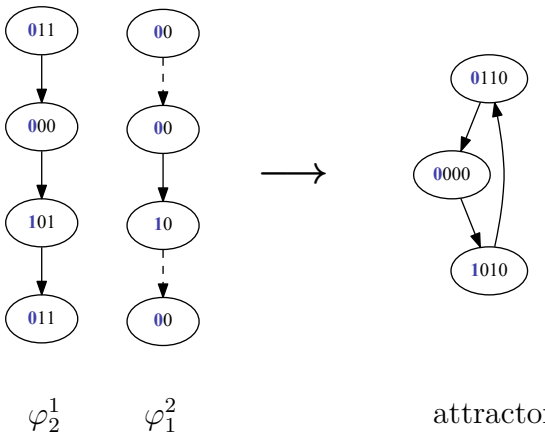


Figure 3.2 Interference aligning a path in φ_2^1 with a path in φ_1^2 starting with the pair $(011, 00)$ followed by $(000, 00)$, then we end with the repeated pair $(011, 00)$, which means we found the attractor. Then by merging the cyclic paths on g_1^1 in $\mathcal{BN}_{Ex1}$ and g_1^2 in $\mathcal{BN}_{Ex2}$ using AND, we identify the composed model attractor $[0110, 0000, 1010, 0110]$.

2) Clearly we have $(011, 00) \notin seen$ and so we set $seen = \{(011, 00)\}$.
3) We have $noStep((011, 00), \varphi_1^1, \varphi_2^2)$ is false and so we set $listSP = [(011, 00)]$.
4) We then set $(S_2, T_2) = (000, 00)$ since $doStep((011, 00), \varphi_1^1, \varphi_2^2) = (000, 00)$ is the interference aligned next state pair for $(011, 00)$.
5) Since $(000, 00) \notin listSP$ we set $(S_1, T_1) = (000, 00)$.
6) Since $(000, 00) \notin seen$ we set $seen = \{(011, 00), (000, 00)\}$.
7) We have $noStep((000, 00), \varphi_1^1, \varphi_2^2)$ is false and so we set $listSP = [(011, 00), (000, 00)]$.
8) We then set $(S_2, T_2) = (101, 10)$ since $doStep((000, 00), \varphi_1^1, \varphi_2^2) = (101, 10)$ is the interference aligned next state pair for $(000, 00)$.
9) Since $(101, 10) \notin listSP$ we set $(S_1, T_1) = (101, 10)$.
10) Since $(101, 10) \notin seen$ we set $seen = \{(011, 00), (000, 00), (101, 10)\}$.
11) We have $noStep((101, 10), \varphi_1^1, \varphi_2^2)$ is false and so we set $list = [(011, 00), (000, 00)$ $, (101, 10)]$.
12) We then set $(S_2, T_2) = (011, 00)$ since $doStep((101, 10), \varphi_1^1, \varphi_2^2) = (011, 00)$ is the interference aligned next state pair for $(101, 10)$.
13) Since $(011, 00) \in listSP$, $extCP(listSP)$ will then return the attractor $[0110, 0000, 1010,$ $0110]$. The above steps are illustrated in Figure 3.2.
14) We now return to the outer loop to process the remaining two aligned state pairs $(000, 00)$ and $(101, 10)$, but they have already been considered in the process above.

findAttTwo$(\varphi_1^1, \varphi_1^2)$: We have $|(\mathcal{S}_{\varphi_1^1} \times \mathcal{S}_{\varphi_1^2})| = 2$ possible state pairs, but there is only one state pair $(001, 00)$ that aligns. So *findAttTwo* will process $(001, 00)$ resulting in $listSP = [(001, 00), (001, 00)]$ and $extCP(listSP)$ returns the point attractor $[0010, 0010]$. It can be seen that the algorithm has correctly identified the attractors for $\mathcal{C}(\mathcal{BN}_{Ex1}, \mathcal{BN}_{Ex2}, g_1^1, g_1^2)$.

In order to make our approach practical, the algorithm *findAttTwo* has been used as the basis for developing a prototype support tool[1] for compositionally identifying attractors. The tool is implemented in Python and makes use of the *NetworkX package* [201] which has tools to represent and manipulate network structures. The support tool reads in state graphs for the subnetworks using the *DOT* file format [53], and takes the position of the merged entities of each submodel as inputs. The tool then generates the required interference state graphs by adding the necessary interference edges based on the composed entities. It then uses an implementation of *Tarjan's algorithm* [200], with a modification of it [202] in the NetworkX package to find the required SCCs. The support tool then applies an implementation of the *findAttTwo* algorithm to each of the possible pairs of SCCs, and returns all the attractors found.

3.4 Case Study

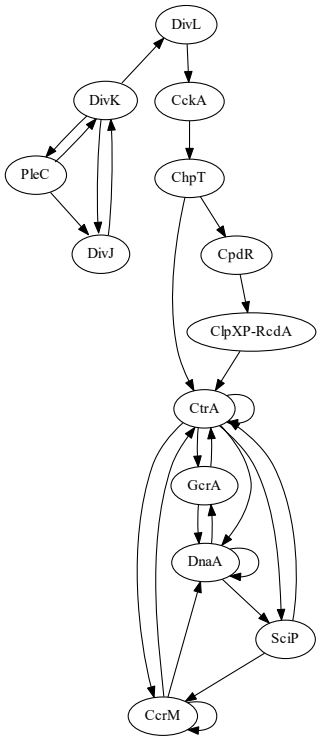
In this section, we illustrate the application of our techniques and tools with a case study based on analysing a BN modelling the regulatory network for cell differentiation in the bacteria *Caulobacter crescentus* [5, 6].

3.4.1 Qualitative Model for Cell Differentiation

The bacteria *C. crescentus* [199] is a model organism for studying cellular differentiation and asymmetric division in bacteria, as its division cycle generates two phenotypes: the stalked and the swarmer cell types. A set of BN models for analysing the regulatory network behind cell differentiation in *C. crescentus* were proposed in [5, 6], and we use these models as the basis for the Boolean network $\mathcal{BN}_{Cc}$ presented in Figure 3.3.

Based on the approach suggested in [5, 6] we decompose this BN into two subnetworks $\mathcal{BN}_{Cc1}$ and $\mathcal{BN}_{Cc2}$ (see Figure 3.4) by splitting the entity $CtrA$ into two entities

[1]For more information about the tool and to obtain a copy, please email the authors.



$$
\begin{aligned}
CtrA &= ((CtrA \vee GcrA) \wedge \\
&\quad \neg CcrM \wedge \neg SciP) \wedge \\
&\quad (ChpT \wedge \\
&\quad \neg ClpXP - RcdA) \\
GcrA &= DnaA \wedge \neg CtrA \\
DnaA &= CtrA \wedge CcrM \wedge \\
&\quad \neg GcrA \wedge \neg DnaA \\
CcrM &= CtrA \wedge \neg CcrM \wedge \\
&\quad \neg SciP \\
SciP &= CtrA \wedge \neg DnaA \\
DivK &= DivJ \wedge \neg PleC \\
PleC &= \neg DivK \\
DivJ &= DivK \wedge \neg PleC \\
DivL &= \neg DivK \\
CckA &= DivL \\
ChpT &= CckA \\
CpdR &= ChpT \\
ClpXP - RcdA &= \neg CpdR
\end{aligned}
$$

Figure 3.3 The BN model $\mathcal{BN}_{Cc}$ based on the model of the regulatory network for cell differentiation in *C. crescentus* developed by [5, 6].

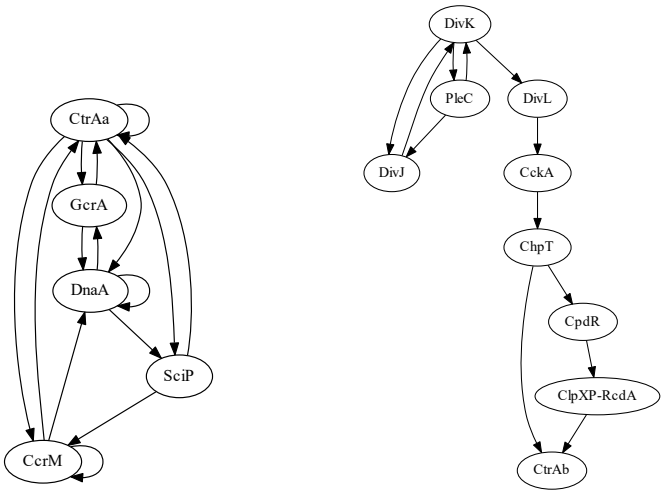
$CtrA_a$ and $CtrA_b$ with corresponding next-state functions

$$CtrA_a = ((CtrA \vee GcrA) \wedge \neg CcrM \wedge \neg SciP),$$

$$CtrA_b = (ChpT \wedge \neg ClpXP - RcdA)$$

It can be shown that the composition $\mathcal{C}(\mathcal{BN}_{Cc1}, \mathcal{BN}_{Cc2}, CtrA_a, CtrA_b)$ of the two subnetworks on the entities $CtrA_a$ and $CtrA_b$ using conjunction does correctly result in the original BN $\mathcal{BN}_{Cc}$.

This decomposition can be seen to dramatically reduce the state space that needs to be considered; the original model had $2^{13} = 8192$ global states, while the two submodels have $2^5 = 32$ and $2^9 = 512$.



$\mathcal{BN}_{Cc1}$ $\mathcal{BN}_{Cc2}$

Figure 3.4 Subnetworks $\mathcal{BN}_{Cc1}$ and $\mathcal{BN}_{Cc2}$ of the qualitative model $\mathcal{BN}_{Cc}$ (based on [5, 6]), where a node $CtrA$ is shared between them and named $CtrA_a$ in $\mathcal{BN}_{Cc1}$ and $CtrA_b$ in $\mathcal{BN}_{Cc2}$.

3.4.2 Application of Our Approach

In this section we explain how we apply our tool support to $\mathcal{BN}_{Cc1}$ and $\mathcal{BN}_{Cc2}$ to identify the attractors of the composed model $\mathcal{BN}_{Cc}$. We note that the purpose of this work is not to produce any new biological insight but to provide a clear example of the practical application of the developed techniques and tools.

The process starts by generating dot files for the state graphs corresponding to $\mathcal{BN}_{Cc1}$ and $\mathcal{BN}_{Cc2}$ (this is achieved using standard tools such as *GinSim* [3]). These dot files are then used to input the state graphs to our support tool which then generates the corresponding interference state graphs. The support tool then identifies the SCCs in the interference state graphs: there are two SCCs φ_1^1 and φ_2^1 in $SG_{CtrA_a}(\mathcal{BN}_{Cc1})$; and two SCCs φ_1^2 and φ_2^2 in $SG_{CtrA_b}(\mathcal{BN}_{Cc2})$ (see Figure 3.5). The total number of possible state pairs is 15, but the tool only considers the eight state pairs that align (see below).

We are then able to apply the implementation of *findAttTwo* to the four possible SCC pairs. The result is that the support tool correctly identifies the three attractors for $\mathcal{BN}_{Cc}$ (this was verified using existing tools such as *BoolNet* [55]).

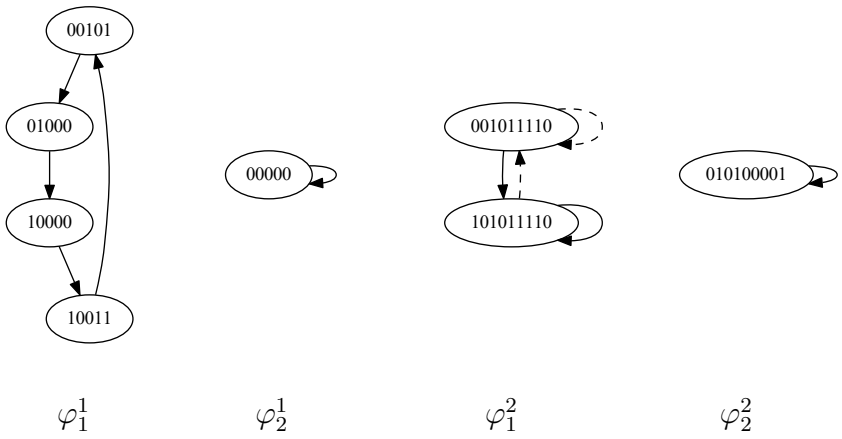


Figure 3.5 The identified SCCs φ_1^1 and φ_2^1 for $SG_{CtrA_a}(\mathcal{BN}_{Cc1})$) (state order is $CtrA_a$, $GcrA$, $DnaA$, $CcrM$, $SciP$), and φ_1^2 and φ_2^2 for $SG_{CtrA_b}(\mathcal{BN}_{Cc2})$) (state order is $CtrA_b$, $DivK$, $PleC$, $DivJ$, $DivL$, $CckA$, $ChpT$, $CpdR$, $ClpXP - RcdA$).

The following summarises the results produced by the tool:

1) *findAttTwo*$(\varphi_1^1, \varphi_1^2)$: There are four aligned state pairs to consider. Starting with the state pair $(00101, 001011110)$ the result is the single list of state pairs
$\{[(00101, 001011110), (01000, 001011110), (10000, 101011110), (10011, 10101110), (00101, 001011110)]\}$
which are merged to produce the attractor
$[0010101011110, 0100001011110, 1000001011110, 1001101011110, 0010101011110]$.
The other aligned state pairs $\{(01000, 001011110), (10000, 101011110), (10011, 10101110)\}$ would produce the same attractor but are not processed as they have already been added to the *seen* list.

2) *findAttTwo*$(\varphi_1^1, \varphi_2^2)$: There are two aligned state pairs to consider. The algorithm returns $\{\}$ and no attractors are found.

3) *findAttTwo*$(\varphi_2^1, \varphi_1^2)$: There is one aligned state pair $(00000, 001011110)$ that results in the single list of state pairs
$\{[(00000, 001011110), (00000, 001011110)]\}$
which are merged to produce the attractor
$[0000001011110, 0000001011110]$.

4) *findAttTwo*$(\varphi_2^1, \varphi_2^2)$: There is one aligned state pair $(00000, 010100001)$ that re-

sults in the single list of state pairs
$\{[(00000, 010100001), (00000, 010100001)]\}$
which are merged to produce the attractor
$[0000010100001, 0000010100001]$.

Note that the genes in the composed model are encoded in the following order: $CtrA$, $GcrA$, $DnaA$, $CcrM$, $SciP$, $DivK$, $PleC$, $DivJ$, $DivL$, $CckA$, $ChpT$, $CpdR$, $ClpXP - RcdA$.

3.5 Extending Attractor Identification to Arbitrary Compositions

In practice, we would like to generalise our approach to identify attractors in an arbitrary composition consisting of multiple BNs by merging multiple entities. To gain insight to arbitrary compositions, we started to consider composing three BNs based on a sequence of pairwise BNs structure presented in the existing framework [4]. It turned out that there are two main cases which we have investigated:

- Case 1 is composing three BNs on the same entities from each submodel (see Figure 3.6). Let $\mathcal{C}^{\wedge}(\mathcal{BN}_1, \mathcal{BN}_2, \mathcal{BN}_3, (g_1^1, g_1^2), (g_1^2, g_1^3))$ represent the composition of $\mathcal{BN}_1$, $\mathcal{BN}_2$ and $\mathcal{BN}_3$ by merging g_1^1, g_1^2 to form g_1^c and then merging g_1^c with g_1^3 which resulted in only one merged entity g_2^c in the final model.
- Case 2 is composing three BNs on different entities from each submodel (see Figure 3.7). Let $\mathcal{C}^{\wedge}(\mathcal{BN}_1, \mathcal{BN}_2, \mathcal{BN}_3, (g_1^1, g_1^2), (g_2^2, g_1^3))$ represent the composition of a $\mathcal{BN}_1$, $\mathcal{BN}_2$ and $\mathcal{BN}_3$ by merging g_1^1 and g_1^2 to form a first merged entity g_1^c and merging g_2^2 and g_1^3 to form a second merged entity g_2^c.

That leads to two extended definitions for *interference alignment*. First, we provide a formal definition of *interference alignment* for a composition of three BNs composed of the same entities as follows.

Definition 12. Let $\alpha_1 = \langle S_1, S_2, ...\rangle \in Path(SG_{g_1^1}(\mathcal{BN}_1))$, $\alpha_2 = \langle T_1, T_2, ...\rangle \in Path(SG_{g_1^2}(\mathcal{BN}_2))$ and $\alpha_3 = \langle V_1, V_2, ...\rangle \in Path(SG_{g_1^3}(\mathcal{BN}_3))$. Then we say that α_1, α_2 and α_3 *interference align* iff $\mathcal{P}_{g_1^1}(\alpha_1) = \mathcal{P}_{g_1^2}(\alpha_2) = \mathcal{P}_{g_1^3}(\alpha_3)$ and for any $i \in \mathbb{N}$, we have at least one step of $S_i \xrightarrow[g_1^1]{\mathcal{BN}_1} S_{i+1}$, $T_i \xrightarrow[g_1^2]{\mathcal{BN}_2} T_{i+1}$ or $V_i \xrightarrow[g_1^3]{\mathcal{BN}_3} V_{i+1}$ is not an interference step.

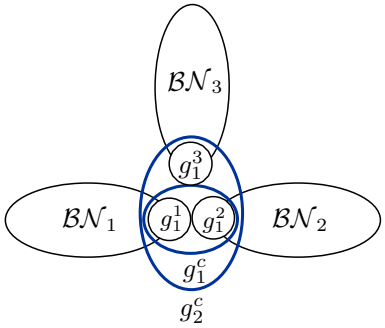


Figure 3.6 Pictorial representation of composing $\mathcal{BN}_1$, $\mathcal{BN}_2$ and $\mathcal{BN}_3$ by merging g_1^1, g_1^2 to form g_1^c and then merging g_1^c with g_1^3 which resulted in a merged entity g_2^c in the final model.

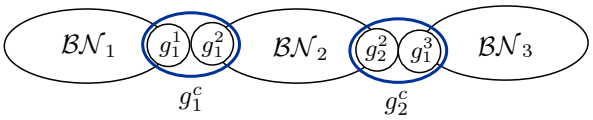


Figure 3.7 Pictorial representation of composing $\mathcal{BN}_1$, $\mathcal{BN}_2$ and $\mathcal{BN}_3$ by merging g_1^1, g_1^2 to form a first merged entity g_1^c and merging g_2^2 and g_1^3 to form a second merged entity g_2^c.

Then, we provide a formal definition of *interference alignment* for a composition of three BNs composed of different entities as follows.

Definition 13. Let $\alpha_1 = \langle S_1, S_2, ... \rangle \in Path(SG_{g_1^1}(\mathcal{BN}_1))$, $\alpha_2 = \langle T_1, T_2, ... \rangle \in Path(SG_{g_1^2,g_2^2}(\mathcal{BN}_2))$ and $\alpha_3 = \langle V_1, V_2, ... \rangle \in Path(SG_{g_1^3}(\mathcal{BN}_3))$. Then we say that α_1, α_2 and α_3 *interference align* iff $\mathcal{P}_{g_1^1}(\alpha_1) = \mathcal{P}_{g_1^2}(\alpha_2)$, $\mathcal{P}_{g_2^2}(\alpha_2) = \mathcal{P}_{g_1^3}(\alpha_3)$ and for any $i \in \mathbb{N}$, we have:
i) $S_i \xrightarrow[g_1^1]{\mathcal{BN}_1} S_{i+1}$ and $T_i \xrightarrow[g_1^2,g_2^2]{\mathcal{BN}_2} T_{i+1}$ are not both interference steps on g_1^1 and g_1^2.
ii) $T_i \xrightarrow[g_1^2,g_2^2]{\mathcal{BN}_2} T_{i+1}$ and $V_i \xrightarrow[g_1^3]{\mathcal{BN}_3} V_{i+1}$ are not both interference steps on g_2^2 and g_1^3.

The difference between the two definitions above is that the submodel in the middle of a composition on different entities can experience interference in two entities, while each submodel in the other form of composition experiences interference in one entity.

For both composition forms, we have proved that the *interference alignment* property captures when three paths in interference state graphs can be merged to create a path in a constructed model. Following that, we proved that the extended approach is correct by showing its *soundness* and *completeness*. We extended the preliminary results concerning projecting and merging paths in a composed model to show that

the extended approach is complete. Based on the theoretical approach, we formulated and implemented an algorithm to identify attractors in three BNs.

We then started to consider multiple BNs based on a restricted composition structure of a sequence of pairwise BNs building on the idea is presented in [4] (see Figure 3.8). We partially proved the results corresponding to the above lemmas and theorems, but we started to experience difficulties. First, the representation of definitions and notations became too complex. In addition, it was difficult to keep track of the names of composed entities that appeared in the final model, and the submodels' entities were merged together. We also wanted to expand the types of compositional structure allowed, which was difficult to do with the current definitions. Therefore, it became clear that we needed to develop a new general definition of a composition to support arbitrary compositions and simplify the representation of definitions and results to extend our attractor identification approach. This is considered in detail in the next chapter.

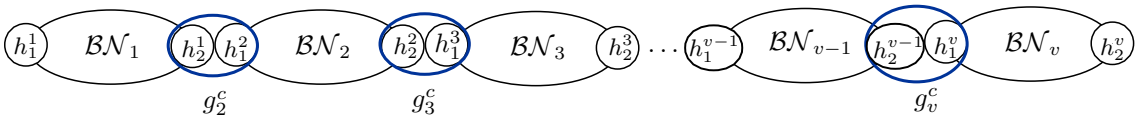


Figure 3.8 Pictorial representation of sequentially composing multiple Boolean networks $\mathcal{BN}_1, ..., \mathcal{BN}_v$ to form a composed system $\mathcal{C}_v$, adapted from [4]. We consider tow cases of merging entities: (1) merging distinct entities from each $\mathcal{BN}_j$ that for each $j \in \{1, ..., v\}$ we have $h_1^j \neq h_2^j$. We end up with multiple new merged entities, such as $g_2^c, ..., g_v^c$; and (2) merging same entities from each $\mathcal{BN}_j$ that for each $j \in \{1, ..., v\}$ we have $h_1^j = h_2^j$. This composition results in a single new merged entity called g_v^c.

3.6 Conclusions

In this chapter we took an existing compositional framework for BNs [48, 49] and extended it by developing new compositional techniques and tools for attractor identification in a composed model involving two BNs. This work was important as providing support for compositional attractor analysis is crucial in addressing the state space explosion problem, and in ensuring the practical applicability of the compositional framework for analysing BNs.

The initial work presented here focused on the composition of two subnetworks and introduced important new concepts and ideas for attractor analysis. Our theoretical approach is based on merging cyclic paths in each submodel's interference state graph.

The key idea is to use the SCCs in the interference state graphs to identify cyclic paths that can be merged to form attractors. We defined a crucial new property called *interference alignment*, and we proved that this property captures when paths in interference state graphs can be merged to create a path in a composed model. As a result, merging interference aligned cyclic paths in the identified SCCs forms attractors in the composed model. We proved that our approach of identifying attractors is correct by proving its the *soundness* and *completeness*.

It is important to note that while this chapter focuses on using a conjunction to merge the behaviour of entities in a composition, all the results presented can straightforwardly be adapted to the use of disjunction. It turns out that the key property for the results is the idempotency. In addition, the algorithm will not be changed when we use the disjunction for merging entities because the algorithm works on state graphs.

Developing tool support is important in automating our developed approach, and allowing its practical application. An interesting challenge of this work is how to find an algorithmic way to implement our theoretical approach, as cyclic paths represent infinite objects. We developed a key new concept of *interference aligned next state pair* to form cyclic paths compositionally. We formulated an algorithm for a function *findAttTwo*, which given the SCCs from the interference state graphs for the two subnetworks can find all attractors associated with the composed model based on the idea of *interference aligned next state pairs*. We used *findAttTwo* algorithm to implement the support tool using Python. We applied the tools developed for attractor analysis to a case study based on analysing a BN modelling the regulatory network for cell differentiation in the bacteria *Caulobacter crescentus* [5, 6]. While the case study is small, it gave important initial insight into the applicability of the approach, and motivated us to consider generalising the existing composition definition to support multiple submodels of large biological systems.

The work in this chapter is significant because it provides a new fundamental idea for compositionally identifying attractors; however, more work is required to develop a technique into a practical approach. Thus, we would like to extend the attractor analysis approach conducted in this chapter to an arbitrary composition (i.e. a composition involving multiple BNs, which are composed over multiple entities). However, the initial work on extending the results to three and a sequence of pairwise BNs structures (Section 3.5) showed that the current definition of a composition could not be used as the basis of such a development. This is because of the complexity of

the underlying definitions and notations, and the difficulty of finding a way of tracking composed entities. It has become clear in this chapter that a new definition for a composition is required, specifically focused on the idea of an arbitrary composition. In the subsequent chapter, we set out to develop such a new general formulation of a composition, and to extend our initial results from the attractor identification approach to this.

A range of work on compositionally identifying attractors exists in the literature (see the related work given in Section 2.5.3). However, the framework we have begun to develop in this chapter is fundamentally different from the existing work in the literature. To begin with, the compositional approach we used is designed to compositionally construct BN models and not just for analysis. Moreover, we use the SCCs associated with the behaviour of a subnetwork's interference state graph, while in the literature, the SCCs appear to be used to decompose the wiring diagram. The new approach we are developing here is novel because it allows the compositional construction models and at the same time the compositional analysis.

The interesting ideas introduced in this chapter are now further extended and developed into practical techniques and tools in Chapter 4 that follows.

Chapter 4

Identifying Attractors for Generalised Compositions

4.1 Introduction

In the previous chapter, we developed a new approach to identify attractors in a model resulting from a composition of two BNs. We proved that the approach is correct by showing that it is *sound* and *complete*. We developed a prototype support tool, and illustrated its application using a biologically relevant case study from the literature. The results were promising, but to be practically useful, we need to expand the type of composition allowed so that a large BN could be constructed from small components simultaneously. At the end of Chapter 3, we started to investigate extending the approach to a composition involving multiple BNs, in particular, three BNs and a sequence of pairwise BNs structure based on the existing framework. This investigation highlighted the limitations of the current formulation of a composition when considering arbitrary compositions. In particular, the notations and definitions became complex and this prevented the formulation of results for arbitrary compositions. It is therefore clear that a new definition for arbitrary composition was required.

In this chapter, we develop a new general formulation of an arbitrary composition involving a set of BNs based on the underlying ideas of the original compositional framework. The new formulation of a composition is based on a graph structure to represent arbitrary compositions. It has led to a more concise formal definition of a composition and remove the need for multiple cases. The new formulation requires us to update the definition of an *interference state graph* and reprove an important theorem in

order to show that an interference state graph captures all possible behaviour that can result for a BN. It is important to note that this new formulation is not straightforward. We have to introduce definitions and notations for reasoning about a composition. In particular, we must find a way to define the entities used in a composition for each BN. Furthermore, we must define the group of entities used in a composition and the BN to which each entity belongs. The most important aspect is defining the set of entities that merged together and experienced interference.

We take the developed approach for identifying attractors in Chapter 3 and extend the key ideas to the new general formalisation of a composition. We follow this approach of identifying the strongly connected components associated with the interference state graphs for the underlying BNs, and then merged their cyclic behaviour; in doing so, there are a number of areas that needed to be considered. Subsequently, we analyse an interference state graph based on having a set of entities that can experience interference, rather than just one entity. Therefore, we extend the definitions of a *normal step* and an *interference step* to an interference state graph for multiple entities. In addition, we have to indicate when potential paths from underlying BNs could be merged on multiple entities to produce attractors in a composed model. As a consequence, we extend the key property *interference alignment*. We then formally prove that it is correct by showing that it is *sound* and *complete*.

We note that while this thesis focuses on using a conjunction to merge the behaviour of entities in a composition, all the results presented can straightforwardly be adapted to the use of disjunction. This is further discussed in Section 4.4.

This chapter is organised as follows. In Section 4.2, we present a new general definition of a composition. In Section 4.3, we extend our new theoretical approach for compositionally identifying attractors to support an arbitrary composition, which we prove is correct by showing that it is *sound* and *complete*. Finally, we make concluding remarks and discuss the future work in Section 4.4.

4.2 New Formulation of Composition

In order to simultaneously construct a composition from multiple BNs, a new formulation for a composition is required. In this section, we develop a new general formulation of a composition based on using a graph structure that extends the existing approach

of composing two BNs [48, 49, 4]. This work represents a significant step forward in the original compositional framework.

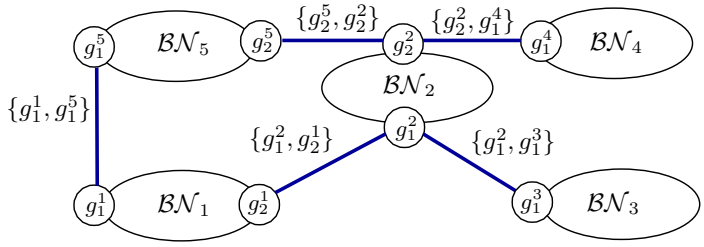


Figure 4.1 Pictorial representation of a composition involving five Boolean networks $\mathcal{BN}_1, ..., \mathcal{BN}_5$ where thick blue edges represent the entities used in the composition.

The idea is that each BN involved in the composition is a node in the graph. The composition of two BNs is then represented using an edge connecting the two BNs, which specifies the entities used in the composition.

To illustrate the idea behind this graph structure to specify a general composition, consider composing five Boolean networks $\mathcal{BN}_1, ..., \mathcal{BN}_5$ as shown in Figure 4.1. The composition of $\mathcal{BN}_1$ and $\mathcal{BN}_2$ is specified by the edge $\{g_1^2, g_2^1\}$ and the composition of $\mathcal{BN}_2$ and $\mathcal{BN}_3$ by the edge $\{g_1^2, g_1^3\}$ (note that a set is used as there is no order in the composition).

This leads to the following new general definition of a composition, which significantly extends the definition given in [49].

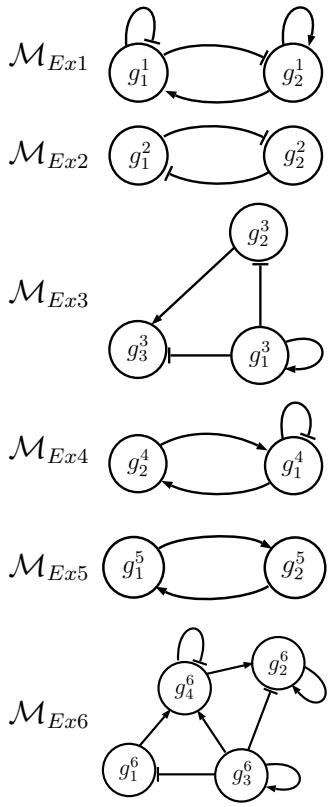
Definition 14. *(Composition) A composition Σ is defined to be a pair $\Sigma = (M, E)$, where $M = \{\mathcal{BN}_1, \ldots, \mathcal{BN}_n\}$ is a set of BNs for some $n \in \mathbb{N}$, $n > 1$, and*

$$E \subseteq \{\{g_1, g_2\} \mid \mathcal{BN}_i, \mathcal{BN}_j \in M,\ \mathcal{BN}_i \neq \mathcal{BN}_j \text{ and } g_1 \in \mathcal{BN}_i,\ g_2 \in \mathcal{BN}_j\}$$

is a set defining the entities merged which satisfies the following condition: for each $\mathcal{BN}_i \in M$ there must exist an entity $g_1 \in \mathcal{BN}_i$ such that $\{g_1, g_2\} \in E$, for some $\mathcal{BN}_j \in M$, $\mathcal{BN}_i \neq \mathcal{BN}_j$ and $g_2 \in \mathcal{BN}_j$.

Recall six example Boolean networks $\mathcal{M}_{Ex1}$, $\mathcal{M}_{Ex2}$, $\mathcal{M}_{Ex3}$, $\mathcal{M}_{Ex4}$, $\mathcal{M}_{Ex5}$ and $\mathcal{M}_{Ex6}$ (see Figure 2.6). Consider composing $\mathcal{M}_{Ex1}$, $\mathcal{M}_{Ex2}$, $\mathcal{M}_{Ex3}$, $\mathcal{M}_{Ex4}$, $\mathcal{M}_{Ex5}$ and $\mathcal{M}_{Ex6}$ using conjunction as follows:

- $\mathcal{M}_{Ex1}$, $\mathcal{M}_{Ex3}$, $\mathcal{M}_{Ex4}$ and $\mathcal{M}_{Ex5}$ are composed by merging entities g_1^1, g_2^3 , g_2^4 and g_2^5.



$$[g_1^1] = \overline{g_1^1} \vee g_2^1$$
$$[g_2^1] = \overline{g_1^1} \vee g_2^1$$
$$[g_1^2] = \overline{g_2^2}$$
$$[g_2^2] = \overline{g_1^2}$$
$$[g_1^3] = g_1^3$$
$$[g_2^3] = \overline{g_1^3}$$
$$[g_3^3] = \overline{g_1^3} \wedge g_2^3$$
$$[g_1^4] = \overline{g_1^4} \vee g_2^4$$
$$[g_2^4] = g_1^4$$
$$[g_1^5] = g_2^5$$
$$[g_2^5] = g_1^5$$
$$[g_1^6] = \overline{g_3^6}$$
$$[g_2^6] = g_2^6 \wedge \overline{g_3^6} \wedge g_4^6$$
$$[g_3^6] = g_3^6$$
$$[g_4^6] = g_1^6 \wedge g_3^6 \wedge \overline{g_4^6}$$

Figure 4.2 Six further example Boolean networks $\mathcal{M}_{Ex1}$, $\mathcal{M}_{Ex2}$, $\mathcal{M}_{Ex3}$, $\mathcal{M}_{Ex4}$, $\mathcal{M}_{Ex5}$ and $\mathcal{M}_{Ex6}$.

- $\mathcal{M}_{Ex2}$, $\mathcal{M}_{Ex5}$ and $\mathcal{M}_{Ex6}$ are composed by merging entities g_1^2, g_1^5 and g_1^6.
- $\mathcal{M}_{Ex3}$ and $\mathcal{M}_{Ex6}$ are composed by merging entities g_3^3 and g_2^6.

The resulting composition Σ_{Ex} is depicted in Figure 4.3 and can be formally specified by $\Sigma_{Ex} = (M_{Ex}, E_{Ex})$, where

$$M_{Ex} = \{\mathcal{M}_{Ex1}, \ldots, \mathcal{M}_{Ex6}\} \text{ and}$$

$$E_{Ex} = \{\{g_1^1, g_2^5\}, \{g_1^1, g_2^3\}, \{g_2^3, g_2^4\}, \{g_1^5, g_1^2\}, \{g_1^2, g_1^6\}, \{g_3^3, g_2^6\}\}$$

It is worth noting that an entity can be used in more than one composition as is the case for entity g_1^2 which is used to compose $\mathcal{M}_{Ex2}$ with $\mathcal{M}_{Ex5}$ and $\mathcal{M}_{Ex6}$.

The representation of a *global state* which we used in Chapter 3 is based on a tuple of a Boolean states $(s_1 \ldots s_n)$, where $s_i \in \mathbb{B}$ represent the state of $g_i \in \mathcal{BN}$ and a global state in the composition has the form $S = (s\ s_2^1\ \ldots\ s_n^1\ s_2^2\ \ldots\ s_m^2) \in \mathcal{S}_\mathcal{C}$, where s is the state of the new merged entity g^c (see Section 2.4). However, this

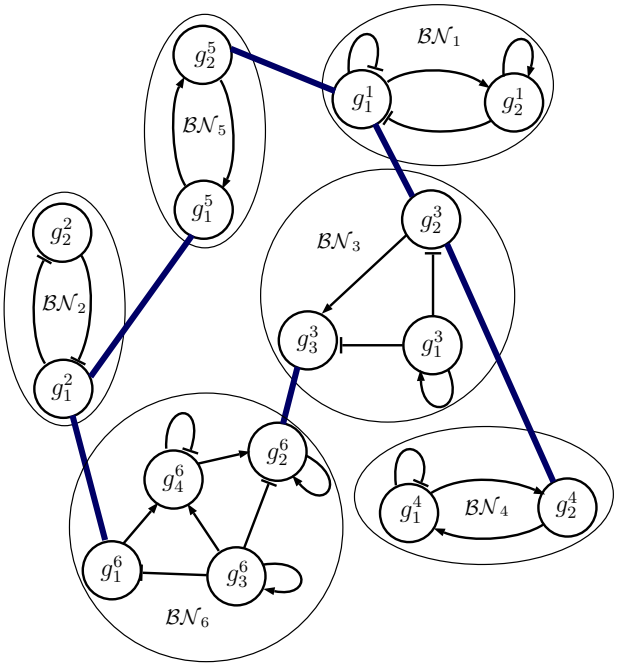


Figure 4.3 An example composition Σ_{Ex} in which the six Boolean networks $\mathcal{M}_{Ex1}, \ldots,$ $\mathcal{M}_{Ex6}$ are composed (where the thick blue edges represent entity composition) resulting in the composed entities $g_1^c = \{g_1^1, g_2^3, g_2^4, g_2^5\}$, $g_2^c = \{g_1^2, g_1^5, g_1^6\}$, $g_3^c = \{g_3^3, g_2^6\}$.

representation becomes problematic to use when developing theoretical results for the new formulation of a composition because the exact position of an entity cannot be identified. It therefore became clear that a new representation of a global state was needed.

In order to facilitate the theoretical results developed in this chapter we use the following alternative approach of representing global states as functions instead of tuples. For any Boolean network $\mathcal{BN} = (G, N, F)$, we define a global state to be a function $S : G \to \mathbb{B}$ and let $S(g)$ represent the state of entity $g \in G$ in global state S. As normal we let $S_{\mathcal{BN}} = [G \to \mathbb{B}]$ represent the set of all global states. It should be clear that representing global states as functions, or by using tuples, is equivalent and we move between the two approaches as required to support our formal development.

Given a non-empty subset $X \subseteq G$ we let $S[X] : X \to \mathbb{B}$ represent the global state $S \in S_{\mathcal{BN}}$ projected over X, where $S[X](g) = S(g)$, for any $g \in X$. Given a path $\alpha = \langle S_1, S_2, \ldots \rangle \in Path(SG(\mathcal{BN}))$ we let $\alpha[X] = \langle S_1[X], S_2[X], \ldots \rangle$.

In the sequel, assume we have $n \in \mathbb{N}$, $n > 1$ BNs to compose and we let $\mathcal{BN}_i = (G_i, N_i, F_i)$ be an arbitrary BN for $i = 1, \ldots, n$. We assume that the sets of entities

$G_1, \ldots, G_n$ are disjointed. We let $\Sigma = (M, E)$ be an arbitrary composition with $M = \{\mathcal{BN}_1, \ldots, \mathcal{BN}_n\}$.

We introduce the following important definitions and notation for reasoning about a given composition $\Sigma = (M, E)$.

- We let $gc(\Sigma, \mathcal{BN}_i)$ be the set of all entities from $\mathcal{BN}_i \in M$ involved in the composition Σ defined by

$$gc(\Sigma, \mathcal{BN}_i) = \{g \mid g \in \mathcal{BN}_i \text{ and } \{g, g'\} \in E\}$$

- We let $gc(\Sigma) = gc(\Sigma, \mathcal{BN}_i) \cup \ldots \cup gc(\Sigma, \mathcal{BN}_n)$ be the set of all entities used in the composition.

- We define $\lambda(g)$ to be the index of the BN entity g belongs to, i.e. if $g \in G_i$, for some $i \in \{1, \ldots, n\}$, then $\lambda(g) = i$.

- For any $g \in gc(\Sigma)$ we define $\Delta(g)$ to be the set of entities that entity g is composed with (i.e. the entities that can interfere with the behaviour of g). This set is important, since an entity can be used in more than one composition. We will see later that it is useful to include g itself in this set. We define $\Delta(g)$ formally by

$$\Delta(g) = (\bigcup_{i \in \mathbb{N}} H_i(g)) \ \cup \{g\}$$

  where $H_i(g)$ is defined recursively as follows:

  1) **Base Case**: $H_0(g) = \{g' \mid \{g, g'\} \in E\}$
  2) **Recursive case**: let $i \in \mathbb{N}$ and define

$$H_{i+1}(g) = \{g'' \mid g' \in H_i(g),\ \{g', g''\} \in E\}$$

  For any entity $g \in gc(\Sigma)$ we use $\Delta(g)$ as the name of the composed entity in the Boolean network $BN(\Sigma)$ that results from the composition.

  This is a crucial idea to be able to name entities to be merged in a composition. In Chapter 3, the name of a newly merged entity is g^c. However, here, we have multiple newly merged entities, and need a way to uniquely name them.

- Given an entity $g \in (G_1 \cup \ldots \cup G_n)$ we define

$$\Sigma(g) = \begin{cases} g, & \text{if } g \notin gc(\Sigma); \\ \Delta(g), & \text{otherwise} \end{cases}$$

Given a set of entities $X \subseteq (G_1 \cup \ldots \cup G_n)$ we define $\Sigma(X) = \{\Sigma(g) \mid g \in X\}$.

We let $BN(\Sigma)$ be the BN that results from a composition Σ. We can define $BN(\Sigma)$ formally, as follows.

Definition 15. *(Composed Model) Let $\Sigma = (M, E)$ be a composition. Then, we define the Boolean network $BN(\Sigma) = (G(\Sigma), N(\Sigma), F(\Sigma))$ that results from Σ as follows.*
1. Entities: *$G(\Sigma) = \Sigma(G_1 \cup \ldots \cup G_n)$.*
2. Neighbourhood: *for any entity $h \in G(\Sigma)$, the neighbourhood $N(\Sigma)(h)$ is defined by*

$$N(\Sigma)(h) = \begin{cases} \bigcup_{g \in h} \Sigma(N_{\lambda(g)}(g)), & \text{if } h = \Delta(g'), \text{ for some } g' \in gc(\Sigma); \\ \Sigma(N_{\lambda(h)}(h)), & \text{otherwise} \end{cases}$$

3. Functions: *For any $g \in G(\Sigma)$ we define the next–state function $F(\Sigma)(g)$ on any $S \in S_{BN(\Sigma)}$ by*

$$F(\Sigma)(g)(S[N(\Sigma)(g)]) =$$

$$\begin{cases} \bigwedge_{h \in \Delta(g')} F_{\lambda(h)}(h)(S[\Sigma(N_{\lambda(h)}(h))]), & \text{if } g = \Delta(g'), \text{ for some } g' \in gc(\Sigma); \\ F_{\lambda(g)}(g)(S[\Sigma(N_{\lambda(g)}(g))]), & \text{otherwise} \end{cases}$$

This definition is illustrated by the example shown in Figure 4.3 where the Boolean network $BN(\Sigma_{Ex})$ results from the composition Σ_{Ex}.

Recall that, in order to formalise the possible *interference* that can occur between composed BNs, the *interference state graph* for a BN was introduced (see Definition 16 in Chapter 2). The idea is to extend a BN's state graph with additional edges representing the behaviour that could result from interference caused by a composition. In particular, whenever an entity used in the composition transitions to 1, then we add another edge to the state graph to represent that the entity could instead transition to 0 due to interference (this is assuming conjunction is used in the composition). Since any given BN can have a number of entities involved in the composition, we need to define the set of all possible next states that can occur based on the possibility of interference, as follows.

Let $\mathcal{BN} = (G, N, F)$ be a BN and $S \in S_{\mathcal{BN}}$. For any entity $g \in G$ and $b \in \mathbb{B}$ we define $S[g \to b]$ to represent an update to state S so that g now has state b. More formally, for any $h \in G$ define

$$S[g \to b](h) = \begin{cases} b, & \text{if } g = h; \\ S(h), & \text{otherwise} \end{cases}$$

Let $\upsilon : \mathbb{B} \to \mathcal{P}(\mathbb{B})$ be defined by $\upsilon(0) = \{0\}$ and $\upsilon(1) = \{0, 1\}$.

υ is the set of the possible next state values and there are only two possibilities based on the use of conjunction.

Let $X = \{g_1, \ldots, g_k\} \subseteq G$ be a non–empty subset of entities. Then we define $\Upsilon_X : S_{\mathcal{BN}} \to \mathcal{P}(S_{\mathcal{BN}})$ for any $S \in S_{\mathcal{BN}}$ by

$$\Upsilon_X(S) = \{S[g_1 \to b_1] \cdots [g_k \to b_k] \mid b_1 \in \upsilon(S(g_1)), \ldots, b_k \in \upsilon(S(g_k))\}$$

$\Upsilon_X(S)$ is the set of all possible states that can result from interference (including S itself).

To illustrate the idea, consider the global state $011 \in SG_{\{g_2^3, g_3^3\}}(\mathcal{M}_{Ex3})$ (see Figure 4.4) that experience interference for entities $\{g_2^3, g_3^3\}$ using conjunction. We have $\Upsilon_{\{g_2^3, g_3^3\}}(011) = \{011, 010, 001\}$ representing all possible states that can result from interference.

We can now define the interference state graph based on having a set X of entities that can experience interference (we build our definition based on [4]).

Definition 16. *(Interference State Graph) Let $\mathcal{BN} = (G, N, F)$ be a Boolean network and let $X \subseteq G$. Then we define the interference state graph $SG_X(\mathcal{BN})$ by*

$$SG_X(\mathcal{BN}) = (S_{\mathcal{BN}}, \xrightarrow[X]{\mathcal{BN}})$$

The extended edge relation $\xrightarrow[X]{\mathcal{BN}}$ is defined by $\xrightarrow[X]{\mathcal{BN}} = \xrightarrow{\mathcal{BN}} \cup \mathcal{E}$, where

$$\mathcal{E} = \{S_1 \xrightarrow[X]{\mathcal{BN}} T \mid S_1, S_2 \in S_{\mathcal{BN}},\ S_1 \xrightarrow{\mathcal{BN}} S_2,\ T \in \Upsilon_X(S_2)\}$$

As illustrative examples, consider the interference state graphs depicted in Figure 4.4 for the Boolean networks $\mathcal{M}_{Ex1}$, $\mathcal{M}_{Ex2}$, $\mathcal{M}_{Ex3}$, $\mathcal{M}_{Ex4}$, $\mathcal{M}_{Ex5}$ and $\mathcal{M}_{Ex6}$ introduced previously (see Figure 4.2).

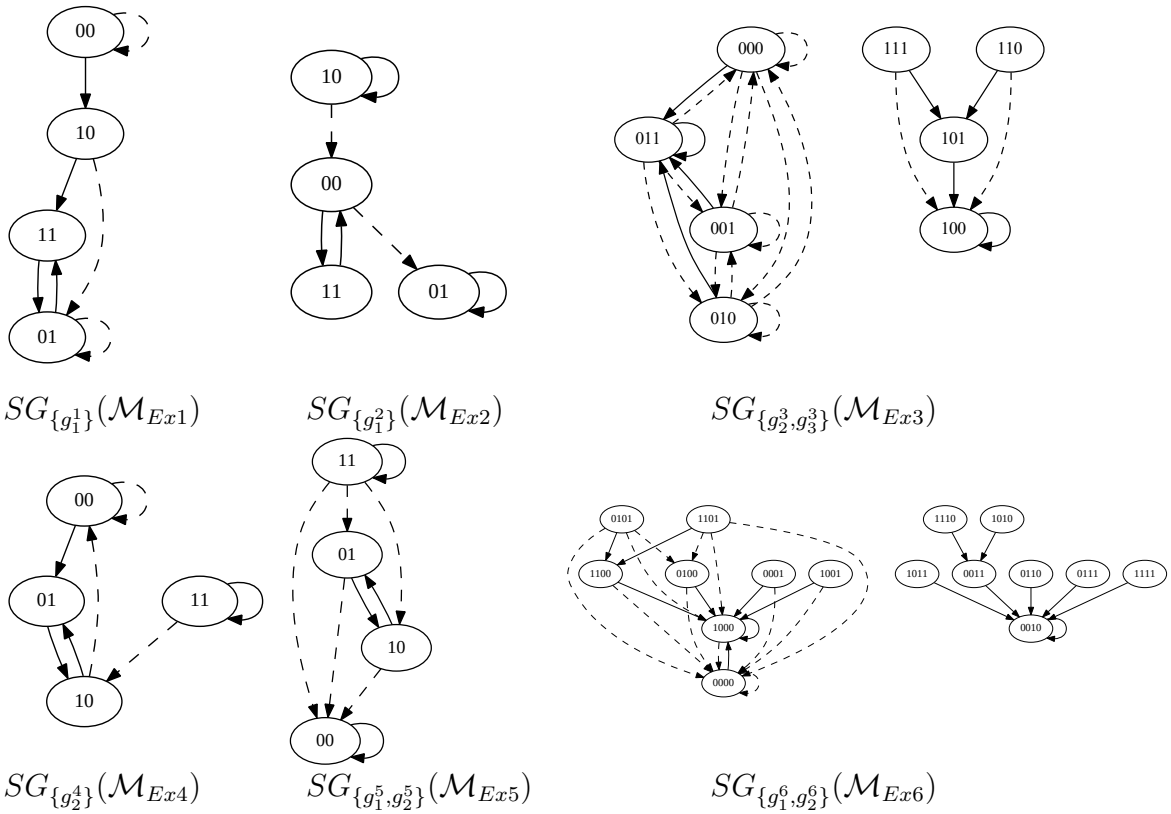


Figure 4.4 The interference state graphs for $\mathcal{M}_{Ex1}$, $\mathcal{M}_{Ex2}$, $\mathcal{M}_{Ex3}$, $\mathcal{M}_{Ex4}$, $\mathcal{M}_{Ex5}$ and $\mathcal{M}_{Ex6}$ induced by the composition Σ_{Ex}.

In the sequel, we let I_i represent the interference state graph $SG_{gc(\Sigma,\mathcal{BN}_i)}(\mathcal{BN}_i)$ when the composition Σ and Boolean network $\mathcal{BN}_i \in M$ are clear from the context.

The following is an important result which shows that an interference state graph captures all the possible behaviour that can result for a BN if it is used in a composition Σ. (Note that this is an updated version of the theorem presented in [49, 4].)

Theorem 8. *Let $\beta \in Path(SG(BN(\Sigma)))$. Then for $i = 1, \ldots, n$ we have*

$$\beta[\Sigma(G_i)] \in Path(I_i)$$

Proof. Let $\beta = \langle S_0, S_1, \ldots \rangle \in Path(SG(BN(\Sigma)))$. Then it suffices to show that for any $i \in \{1, \ldots, n\}$ and any $k \in \mathbb{N}$ we have

$$S_k[\Sigma(G_i)] \xrightarrow[gc(\Sigma, \mathcal{BN}_i)]{\mathcal{BN}_i} S_{k+1}[\Sigma(G_i)] \tag{4.1}$$

Let $S_k[\Sigma(G_i)] \xrightarrow{\mathcal{BN}_i} T$, for some $T \in S_{\mathcal{BN}_i}$. Clearly, for any $g \in G_i$, $g \notin gc(\Sigma, \mathcal{BN}_i)$ we have by the definition of $BN(\Sigma)$ (Definition 15) that

$$T(g) = S_{k+1}[\Sigma(G_i)](g)$$

For any $g \in gc(\Sigma, \mathcal{BN}_i)$ we know by the definition of $BN(\Sigma)$ (Definition 15) that $S_{k+1}(\Sigma(g)) \in \upsilon(T(g))$ and so it follows that

$$S_{k+1}[\Sigma(G_i)] \in \Upsilon_{gc(\Sigma, \mathcal{BN}_i)}(T)$$

Then, by the definition of the interference state graph (Definition 16), it follows that (4.1) must hold. □

4.3 Identifying Attractors in a Composition

In this section, we develop theoretical results for compositional identifying attractors for a general composition (introduced in the previous section). The results and techniques are based on adapting and extending the results introduced in the previous chapter (see section 3.2). However, here, we have to consider multiple cyclic paths in the interference state graphs for the underlying BNs to be merged. In addition, we can have multiple composed entities that experience interference. Thus, the *interference alignment* property needs to be updated. We formally prove that the extended approach is *sound* and *complete* for attractor identification.

Recall the definition of an SCC φ for an interference state graph (see Definition 9). We let φ denote an SCC for an interference state graph $SG_X(\mathcal{BN})$, where X is the set of entities involved in a composition. As an illustrative example, consider the SCCs depicted in Figure 4.5 for the interference state graphs $SG_{\{g_1^1\}}(\mathcal{M}_{Ex1})$, $SG_{\{g_1^2\}}(\mathcal{M}_{Ex2})$, $SG_{\{g_2^3, g_3^3\}}(\mathcal{M}_{Ex3})$, $SG_{\{g_2^4\}}(\mathcal{M}_{Ex4})$, $SG_{\{g_1^5, g_2^5\}}(\mathcal{M}_{Ex5})$ and $SG_{\{g_1^6, g_2^6\}}(\mathcal{M}_{Ex6})$.

Let $X \subseteq G$, $\varphi \in SCC(SG_X(\mathcal{BN}))$ and we use $\alpha \in Path(\varphi)$ again to denote an infinite path over φ. Then we say that α is a *cyclic path* iff there exists $k \in \mathbb{N}$, $k > 0$

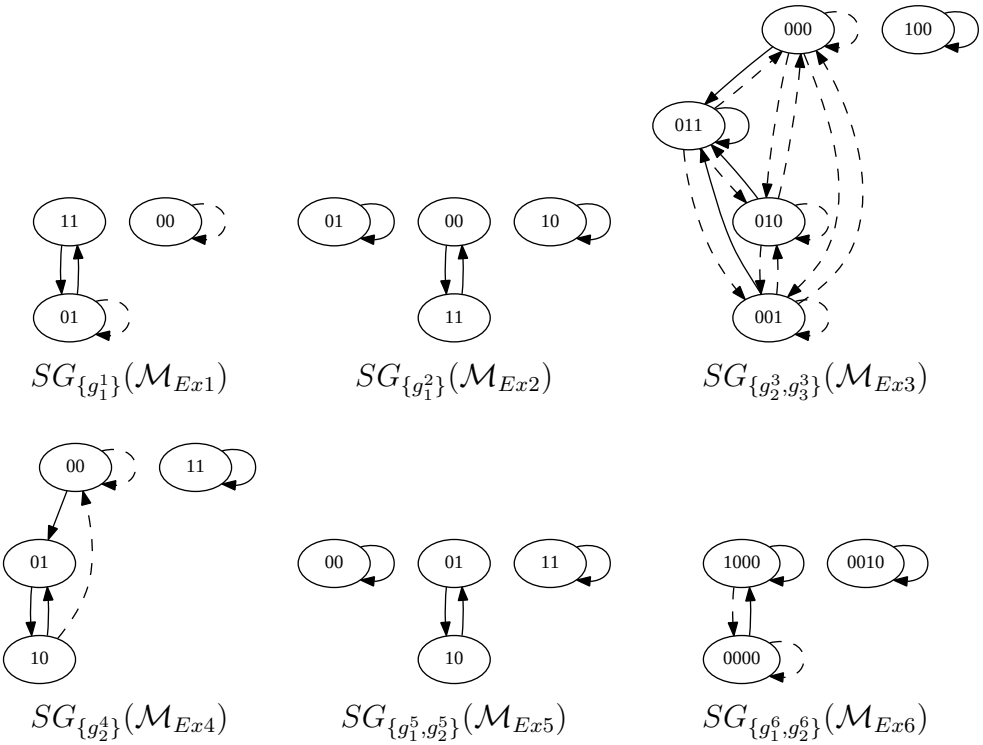


Figure 4.5 The SCCs that arise for the example composition Σ_{Ex}.

and $S_1, \ldots, S_k \in \mathcal{S}_\varphi$ such that

$$\alpha = \langle S_1, \ldots, S_k, S_1, \ldots, S_k, \ldots \rangle$$

We use $CPaths(\varphi)$ again to denote the set of all cyclic paths for an SCC φ.

Given the sets of SCCs for the BNs that are to be composed, we can consider merging the cyclic paths the SCCs generate to identify attractors in the composed model. In order to facilitate this process, we extend the definition of the *interference alignment* property that indicates when potential paths from submodels can be merged to produce paths in the composed model.

We begin by extending the definitions of a *normal step* and an *interference step* (see Section 3.2) to an interference state graph $SG_X(\mathcal{BN})$.

Definition 17. *Let $\mathcal{BN} = (G, N, F)$ be a Boolean network, let $g \in G$ and let $X \subseteq G$. Suppose $S_i \xrightarrow{\mathcal{BN}} S_{i+1}$ and $S_i \xrightarrow[X]{\mathcal{BN}} S'_{i+1}$, for some $S_i, S_{i+1}, S'_{i+1} \in \mathcal{S}_{\mathcal{BN}}$. We say $S_i \xrightarrow[X]{\mathcal{BN}} S'_{i+1}$ is a normal step for g iff $S_{i+1}(g) = S'_{i+1}(g)$. We say $S_i \xrightarrow[X]{\mathcal{BN}} S'_{i+1}$ is an interference step for g iff it is not a normal step for g.*

As an example, consider the $SG_{\{g_2^3,g_3^3\}}(\mathcal{M}_{Ex3})$ depicted in Figure 4.4. Then the step $011 \xrightarrow[\{g_2^3,g_3^3\}]{\mathcal{M}_{Ex3}} 010$ is an interference step for g_3^3 and a normal step for g_2^3.

Next, we define how states and paths can be merged in the context of a composition Σ.

Definition 18. *(Merging States and Paths) Let $S_i \in S_{\mathcal{BN}_i}$, for $i = 1, \ldots, n$. We define $\wedge_\Sigma(S_1, \ldots, S_n) \in S_{BN(\Sigma)}$ to be the composed state where for any $g \in G(\Sigma)$ we have*

$$\wedge_\Sigma(S_1, \ldots, S_n)(g) = \begin{cases} \bigwedge_{h \in \Delta(g')} S_{\lambda(h)}(h), & \text{if } g = \Delta(g'), \text{ for some } g' \in gc(\Sigma); \\ S_{\lambda(g)}(g), & \text{otherwise} \end{cases}$$

Let $\alpha_i = \langle S_0^i, S_1^i, ... \rangle \in Path(I_i)$, for $i = 1, \ldots, n$. We define $\wedge_\Sigma(\alpha_1, \ldots, \alpha_n)$ to be the composed path defined by

$$\wedge_\Sigma(\alpha_1, \ldots, \alpha_n) = \langle \wedge_\Sigma(S_0^1, \ldots, S_0^n), \wedge_\Sigma(S_1^1, \ldots, S_1^n), \ldots \rangle$$

We now extend the important definition of *interference alignment* on paths involved in a composition.

Definition 19. *(Interference Alignment) Let $\alpha_i = \langle S_0^i, S_1^i, ... \rangle \in Path(I_i)$, for $i = 1, \ldots, n$. We say $\alpha_1, \ldots, \alpha_n$* interference align *(for Σ) iff for each $g \in gc(\Sigma)$ and each $k \in \mathbb{N}$ the following hold:*

1) $S_k^{\lambda(g)}(g) = S_k^{\lambda(h)}(h)$, for all $h \in \Delta(g)$;
2) There exists $h \in \Delta(g)$, such that $S_k^{\lambda(h)} \xrightarrow[gc(\Sigma,\mathcal{BN}_{\lambda(h)})]{\mathcal{BN}_{\lambda(h)}} S_{k+1}^{\lambda(h)}$ is a normal step for h.

The idea behind interference alignment is that it captures when paths from the BNs involved in a composition can be merged to create a path in the resulting composed model. It does this by checking to ensure that interference actually occurs at the points required in each path. This is formally shown by the following result.

Lemma 9. *Let $\alpha_1 \in Path(I_1), ..., \alpha_n \in Path(I_n)$. Then if $\alpha_1, \ldots, \alpha_n$ interference align then*

$$\wedge_\Sigma(\alpha_1, \ldots, \alpha_n) \in Path(SG(BN(\Sigma)))$$

Proof. Let $\alpha_i = \langle S_0^i, S_1^i, ... \rangle \in Path(I_i)$, for $i = 1, \ldots, n$, such that $\alpha_1, \ldots, \alpha_n$ interference align. To show $\wedge_\Sigma(\alpha_1, \ldots, \alpha_n) \in Path(SG(BN(\Sigma)))$ it suffices by the definition

of merging paths (Definition 18) to show for any $k \in \mathbb{N}$ that

$$\wedge_\Sigma (S_k^1, \ldots, S_k^n) \xrightarrow{BN(\Sigma)} \wedge_\Sigma (S_{k+1}^1, \ldots, S_{k+1}^n) \tag{4.1}$$

By Definition 15, to prove (4.1) we need to show that for any $g \in G(\Sigma)$ we have

$$\wedge_\Sigma (S_{k+1}^1, \ldots, S_{k+1}^n)(g) = F(\Sigma)(g)(\wedge_\Sigma (S_k^1, \ldots, S_k^n)[N(\Sigma)(g)]) \tag{4.2}$$

Note that by interference alignment, the definition of merging states (Definition 18) and idempotency of conjunction, it follows that for any $p \in \{1, \ldots, n\}$ and any non-empty subset $X \subseteq G_p$ we know

$$\wedge_\Sigma (S_m^1, \ldots, S_m^n)[\Sigma(X)] = S_m^p[X] \tag{4.3}$$

for any $m \in \mathbb{N}$. To show (4.2), there are two cases to consider for each $g \in G(\Sigma)$ depending on whether or not g is a composed entity.

Case 1: Suppose $g \in (G_1 \cup \ldots \cup G_n) \setminus gc(\Sigma)$ (i.e. g is not used in the composition). Then we show each side of (4.2) reduces to $S_{k+1}^{\lambda(g)}(g)$. By (4.3) we know

$$\wedge_\Sigma (S_{k+1}^1, \ldots, S_{k+1}^n)(g) = S_{k+1}^{\lambda(g)}(g)$$

By definition of $BN(\Sigma)$ (Definition 15), given the assumption that g is not used in the composition and (4.3) it follows

$$F(\Sigma)(g)(\wedge_\Sigma (S_k^1, \ldots, S_k^n)[N(\Sigma)(g)]) = F_{\lambda(g)}(g)(S_k^{\lambda(g)}[N_{\lambda(g)}(g)])$$

Then, by assumptions on $\alpha_{\lambda(g)}$ and the assumption that g is not used in the composition, we have

$$F_{\lambda(g)}(g)(S_k^{\lambda(g)}[N_{\lambda(g)}(g)]) = S_{k+1}^{\lambda(g)}(g)$$

Case 2: Suppose $g = \Delta(g')$, for some $g' \in gc(\Sigma)$ (i.e. g is a composed entity).

By definition of merging states (Definition 18) and by assumption that $g = \Delta(g')$ we have

$$\wedge_\Sigma (S_{k+1}^1, \ldots, S_{k+1}^n)(g) = \bigwedge_{h \in g} S_{k+1}^{\lambda(h)}(h)$$

By definition of $BN(\Sigma)$ (Definition 15) and (4.3) it follows

$$F(\Sigma)(g)(\wedge_\Sigma(S_k^1,\ldots,S_k^n)[N(\Sigma)(g)]) = \bigwedge_{h\in g} F_{\lambda(h)}(h)(S_k^{\lambda(h)}[N_{\lambda(h)}(h)])$$

It now remains to show that

$$\bigwedge_{h\in g} S_{k+1}^{\lambda(h)}(h) = \bigwedge_{h\in g} F_{\lambda(h)}(h)(S_k^{\lambda(h)}[N_{\lambda(h)}(h)])$$

There are two cases to consider:

Case 2.1: Suppose $\bigwedge_{h\in g} F_{\lambda(h)}(h)(S_k^{\lambda(h)}[N_{\lambda(h)}(h)]) = 1$.
Then by conjunction it follows that for any $h \in g$ we have

$$F_{\lambda(h)}(h)(S_k^{\lambda(h)}[N_{\lambda(h)}(h)]) = 1$$

It then follows by interference alignment that there must exist a normal step resulting in

$$S_{k+1}^{\lambda(h')}(h') = 1$$

for some $h' \in g$. Then by interference alignment it follows that for all $h \in g$ we have

$$S_{k+1}^{\lambda(h)}(h) = 1$$

and so by conjunction it follows that

$$\bigwedge_{h\in g} S_{k+1}^{\lambda(h)}(h) = 1$$

Case 2.2: Suppose $\bigwedge_{h\in g} F_{\lambda(h)}(h)(S_k^{\lambda(h)}[N_{\lambda(h)}(h)]) = 0$. Then by conjunction there must exist $h \in g$ such that

$$F_{\lambda(h)}(h)(S_k^{\lambda(h)}[N_{\lambda(h)}(h)]) = 0$$

Then, since $\alpha_{\lambda(h)} \in Path(I_{\lambda(h)})$ and by definition of the interference state graph we must have

$$S_{k+1}^{\lambda(h)}(h) = 0$$

and so by definition of conjunction we have

$$\bigwedge_{h \in g} S_{k+1}^{\lambda(h)}(h) = 0$$

□

Combining the formal definitions and results above provides a basis for compositionally identifying attractors in a composed BN model. Suppose we have SCCs $\varphi_i \in SCC(I_i)$ and cyclic paths $\alpha_i \in CPaths(\varphi_i)$, for $i = 1, \ldots, n$. Subsequently, if these cyclic paths interference align then they can be merged to create a cyclic path $\wedge_\Sigma(\alpha_1, \ldots, \alpha_n)$ that must exist in the composed model $BN(\Sigma)$ and so represents an attractor cycle.

To illustrate this idea consider the following cyclic paths that result from the SCCs associated with the composition Σ_{Ex} (see Figure 4.5):

$$\langle 11, 01, 11, \ldots \rangle \in CPaths(SG_{\{g_1^1\}}(\mathcal{M}_{Ex1})),$$

$$\langle 00, 11, 00, \ldots \rangle \in CPaths(SG_{\{g_1^2\}}(\mathcal{M}_{Ex2})),$$

$$\langle 010, 000, 010, \ldots \rangle \in CPaths(SG_{\{g_2^3, g_3^3\}}(\mathcal{M}_{Ex3})),$$

$$\langle 01, 10, 01, \ldots \rangle \in CPaths(SG_{\{g_2^4\}}(\mathcal{M}_{Ex4})),$$

$$\langle 01, 10, 01, \ldots \rangle \in CPaths(SG_{\{g_1^5, g_2^5\}}(\mathcal{M}_{Ex5})),$$

$$\langle 0000, 1000, 0000, \ldots \rangle \in CPaths(SG_{\{g_1^6, g_2^6\}}(\mathcal{M}_{Ex6})).$$

These cyclic paths interference align (for Σ_{Ex}) and when merged produce the cyclic path $\langle 100100000, 010110100, 100100000, \ldots \rangle$. It follows by Lemma 9 that this path is in $Path(SG(BN(\Sigma_{Ex})))$ and therefore that

$$[100100000, 010110100, 100100000]$$

is an attractor for $BN(\Sigma_{Ex})$.

We now consider showing formally that the approach for compositionally identifying attractors in a composed model based on a new formulation is correct. To do this we need to show that it is: *sound* – all attractors found are present in the composed model; and *complete* – every attractor in the composed model is found using the approach.

This idea for compositionally identifying attractors is formally shown to be *sound* by the following theorem.

Theorem 10. *(Soundness) Let $\varphi_i \in SCC(I_i)$ and let $\alpha_i \in CPaths(\varphi_i)$, for $i = 1, \ldots, n$. Then if the cyclic paths $\alpha_1, \ldots, \alpha_n$ interference align then the path $\wedge_\Sigma(\alpha_1, \ldots, \alpha_n)$ represents an attractor in the composed model $BN(\Sigma)$.*

Proof. Let $\varphi_i \in SCC(I_i)$ and let $\alpha_i \in CPaths(\varphi_i)$, for $i = 1, \ldots, n$. Suppose that the cyclic paths $\alpha_1, \ldots, \alpha_n$ interference align. Then, by the above assumptions and Lemma 9, it follows

$$\wedge_\Sigma(\alpha_1, \ldots, \alpha_n) \in Path(SG(BN(\Sigma)))$$

By the definition of cyclic paths we know that for each $i = 1, \ldots, n$ there must exist a minimal $k_i \in \mathbb{N}$ and states $S^i_1, \ldots, S^i_{k_i} \in S_{\mathcal{BN}_i}$ such that

$$\alpha_i = \langle S^i_1, \ldots, S^i_{k_i}, S^i_1, \ldots, S^i_{k_i}, \ldots \rangle,$$

Let $LCM(k_1, \ldots, k_n)$ represent the lowest common multiple of $k_1, \ldots, k_n$. Then it follows that the first $LCM(k_1, \ldots, k_n) + 1$ states of $\wedge_\Sigma(\alpha_1, \ldots, \alpha_n)$ must represent an attractor in $BN(\Sigma)$. □

We now consider showing that the proposed approach is complete, and begin with some necessary preliminary results about projecting and merging paths in a composed model. The first of these results shows that the composition of projected paths in the composed model must result in the original path.

Lemma 11. *Let $\beta \in Path(SG(BN(\Sigma)))$. Then*

$$\wedge_\Sigma(\beta[\Sigma(G_1)], ..., \beta[\Sigma(G_n)]) = \beta$$

Proof. Let $\beta = \langle T_0, T_1, ... \rangle \in Path(SG(BN(\Sigma)))$. Then, it suffices to show that

$$T_i(g) = \wedge_\Sigma(T_i[\Sigma(G_1)], ..., T_i[\Sigma(G_n)])(g)$$

for any $i \in \mathbb{N}$ and any $g \in G(\Sigma)$. We do this using the following two cases.

Case 1: Suppose $g \in (G_1 \cup \ldots \cup G_n) \setminus gc(\Sigma)$ (i.e. g is not used in the composition). Then, by definition of state projection, we know

$$T_i(g) = T_i[\Sigma(G_{\lambda(g)})](g)$$

and so by definition of merging states (Definition 18) we have

$$T_i[\Sigma(G_{\lambda(g)})](g) = \wedge_\Sigma(T_i[\Sigma(G_1)], ..., T_i[\Sigma(G_n)])(g)$$

Case 2: Suppose $g = \Delta(g')$, for some $g' \in gc(\Sigma)$ (i.e. g is a composed entity). By the definition of merging states (Definition 18), we have

$$\wedge_\Sigma(T_i[\Sigma(G_1)], ..., T_i[\Sigma(G_n)])(g) = \bigwedge_{h \in \Delta(g')} T_i[\Sigma(G_{\lambda(h)})](h) \tag{I}$$

Also by the definition of state projection we have

$$T_i[\Sigma(G_{\lambda(g')})](g') = T_i[\Sigma(G_{\lambda(h)})](h)$$

for all $h \in \Delta(g')$, and so it follows from (I) and the idempotency of conjunction that

$$\wedge_\Sigma(T_i[\Sigma(G_1)], ..., T_i[\Sigma(G_n)])(g) = T_i[\Sigma(G_{\lambda(g')})](g')$$

Then, by definition of projection, we have

$$T_i[\Sigma(G_{\lambda(g')})](g') = T_i(g)$$

□

We now show that any path in the composed model projected over the entities of an underlying BN will be a valid path in the interference state graph for that BN.

Lemma 12. *Let $\beta \in Path(SG(BN(\Sigma)))$. Then for any $i \in \mathbb{N}$ we have*

$$\beta[\Sigma(G_i)] \in Path(I_i)$$

Proof. Let $\beta = \langle T_0, T_1, ... \rangle \in Path(SG(BN(\Sigma)))$. Then, it suffices to show that for any $i \in \{1, ..., n\}$ and any $k \in \mathbb{N}$ we have

$$T_k[\Sigma(G_i)] \xrightarrow[gc(\Sigma, \mathcal{BN}_i)]{\mathcal{BN}_i} T_{k+1}[\Sigma(G_i)] \tag{I}$$

Suppose $T_k[\Sigma(G_i)] \xrightarrow{\mathcal{BN}_i} S$, for some $S \in \mathcal{S}_{\mathcal{BN}i}$. Then, by definition of the interference state graph I_i (Definition 16) to show (I), we need to prove

$$T_{k+1}[\Sigma(G_i)] \in \Upsilon_{gc(\Sigma, \mathcal{BN}_i)}(S)$$

We do this by showing that for each $g \in gc(\Sigma, \mathcal{BN}_i)$ we have

$$T_{k+1}[\Sigma(G_i)](g) \in \upsilon(S(g))$$

using the following two cases.

Case 1: Suppose $S(g) = 1$. Then $\upsilon(S(g)) = \{0, 1\}$ and so it follows that

$$T_{k+1}[\Sigma(G_i)](g) \in \upsilon(S(g))$$

Case 2: Suppose $S(g) = 0$. Then by the definition of $BN(\Sigma)$ and conjunction we must have

$$T_{k+1}[\Sigma(G_i)](g) = 0$$

Since $\upsilon(S(g)) = \{0\}$ it then follows that

$$T_{k+1}[\Sigma(G_i)](g) \in \upsilon(S(g))$$

□

We can now prove that the proposed attractor identification approach is *complete* (i.e. every attractor in the composed model is identified by the approach).

Theorem 13. *(Completeness) Let ψ be an attractor for the composed Boolean network $BN(\Sigma)$. Then there must exist $\varphi_i \in SCC(I_i)$, for $i = 1, ..., n$, and cyclic paths $\varphi_i \in CPaths(\varphi_i)$, for $i = 1, ..., n$, such that $\varphi_1, ..., \varphi_n$ interference align and $\wedge_\Sigma(\alpha_1, \ldots, \alpha_n)$ results in the attractor ψ.*

Proof. Let $\psi = [T_1, T_2, \ldots T_k, T_1]$ be an arbitrary attractor in the composed model $BN(\Sigma)$. Then, ψ can be viewed as representing the infinite cyclic path

$$\overline{\psi} = \langle T_1, \ldots T_k, T_1, \ldots T_k, \ldots \rangle$$

It follows by Lemma 12 that for $i = 1, ..., n$ we have

$$\overline{\psi}[\Sigma(G_i)] \in Path(I_i)$$

Furthermore, it is clear that each path $\overline{\psi}[\Sigma(G_i)]$ must be a cyclic path and thus results from an SCC φ_i in the interference state graph I_i. By Lemma 11 it follows that

$$\wedge_\Sigma(\overline{\psi}[\Sigma(G_1)], ..., \overline{\psi}[\Sigma(G_n)]) = \overline{\psi}$$

Given the above, it remains to show that $\varphi_1, ..., \varphi_n$ interference align. To do this, first note that for any $g \in gc(\Sigma)$ we clearly have by the definition of projection that

$$\overline{\psi}[G_{\lambda(g)}](g) = \overline{\psi}[G_{\lambda(h)}](h), \tag{I}$$

for all $h \in \Delta(g)$. So it remains to show that for any $g \in gc(\Sigma)$ there exists $h \in \Delta(g)$ such that

$$T_j[\Sigma(G_{\lambda(h)})] \xrightarrow[gc(\Sigma, \mathcal{BN}_{\lambda(h)})]{\mathcal{BN}_{\lambda(h)}} T_{j+1}[\Sigma(G_{\lambda(h)})]$$

is a normal step. This must hold by the definition of $F(\Sigma)$ in the composed model $BN(\Sigma)$ (Definition 15), since it composes normal steps together using conjunction, and this means at least one normal step cannot be interfered with, given the resulting Boolean value. □

4.4 Conclusion

This chapter set out to address the limitations in the initial work on attractor analysis presented in Chapter 3. In particular, we considered a challenging problem of extending compositional attractor identification techniques and results to an arbitrary composition. A crucial first step was to reformulate a new definition of composition that allows multiple BNs to be composed simultaneously over multiple entities. We then extended the results of Chapter 3 significantly with this new framework. The new techniques and results we have developed significantly advance this compositional framework providing practical new techniques and tools for attractor analysis.

A significant contribution of this chapter is the formulation of a new general definition for the compositional construction of BNs based on using a graph based structure to allow arbitrary compositions. This addressed one of the major challenges we have faced during the process of extending our attractor analysis results to multiple BNs because the original notations and definitions became too complex to handle. Importantly, this simplifies the presentation of key definitions and results for the compositional framework. Providing a way of naming the new merged entities, using

the functions to return the state of specific entities and defining functions to return the index of the BN to which an entity belongs are significant steps in the extended work results. Notably, the new formulation is much more applicable to compositionally constructing models which supports engineering biological systems and aiding the development of decomposition techniques.

Using the new generalised definition of a composition, we were then able to significantly extend the compositional approach presented in Chapter 3. This involved dealing with analysing multiple SCCs in the subnetwork's interference state graphs to identify potential cyclic behaviour. Multiple entities can experience interference under composition, resulting new edges in state graphs. Thus, we provided extended definitions for normal and interference edges. Furthermore, we extended the property of *interference alignment*, which indicates when cyclic paths generated by SCCs can be composed to form attractors. We formally showed that the extended approach was correct by proving that it is both *sound* and *complete*.

Again, as noted in Chapter 3 the results clearly still follow for the OR operator, where the key change is that 0 now interferes with 1. In an interference state graph, interference will occur with the underlying behaviour of a merged entity whenever it wants to transition to 1, but its merged counterpart wants to transition to 0. The definition of the interference state graph can be straightforwardly changed by updating the definition of $\upsilon : \mathbb{B} \to \mathcal{P}(\mathbb{B})$ to be $\upsilon(0) = \{0, 1\}$ and $\upsilon(1) = \{1\}$ (see Section 4.2). In fact, idempotency is the main property for AND and OR Boolean operators (note that there are four such binary operators). The property of idempotence is needed in *Lemma 3*, *Lemma 5*, *Lemma 9* and *Lemma 11*. It is worth noting that the attractor identification techniques and tools are being applied over the state graphs, therefore the $findAtt$ algorithm can be applied over the interference state graph generated when the merge operator used in the composition is OR.

Developing tool support to make attractor analysis techniques practical to realistic models is an important focus in this research. Thus, in the subsequent chapter, we develop a prototype support tool and investigate its practical performance. We formulate an algorithm based on the theoretical results developed in this chapter to involve multiple BNs. We then implement the developed algorithm and conduct experimental studies to evaluate the performance of the tool, and to compare the performance result to existing tools in the literature.

The original compositional framework was developed to support the engineering of BNs from basic parts in order to help address current challenges in synthetic biology

[175–177]. In the future, it would be interesting to use this framework as a basis for developing techniques for automatically decomposing a large BN into parts in order to aid its analysis and the identification of key subnetworks. Moreover, the behaviour preservation results of the original framework [48, 49] need to be extended to the new general setting in this chapter to support the analysis.

The developed new framework provides a basis for considering many interesting problems in BNs. For example, in future work, it would be interesting to consider synthesising BNs with given behaviour using this compositional framework.

Chapter 5

Practical Application: Tool Support and Experimental Studies

5.1 Introduction

In the previous chapter, we developed a new general formulation of composing multiple BNs to support arbitrary compositions. The new formulation clearly simplified the theoretical results and definitions of the original framework. This work significantly advances the original compositional framework, using it as a basis for decomposing and engineering biological systems aimed at addressing the current practical limitations imposed by the state space explosion problem. Then, we extended the compositional attractor analysis results developed in Chapter 3 to support multiple BNs composed by merging multiple entities. In order to make the attractor identification approach practical, we need to develop an algorithm based on the theoretical approach, and implement it to automate our results and conduct experimental studies.

In this chapter, based on our theoretical results, we extend the developed algorithm from Chapter 3 to support arbitrary compositions that involve multiple BNs. The algorithmic approach we take uses a new notion of *aligned state tuples*, that is, a collection of states for the underlying BNs in the composition which are consistent with each other and which are realisable in practice. We must consider the issue of having multiple next states for each global state in the SCCs while pairing up states to form cyclic paths. Thus, we provide the definition of an *interference aligned next state tuple*, which is a key property in our algorithmic approach. We formally show that there is, at most, one interference aligned next state tuple for any state tuple processed

in the algorithm. We carefully consider how to efficiently generate aligned state tuples to ensure that the algorithm developed is practical. We present an interesting idea of computing interference aligned next state tuple efficiently by focusing on composed entities in each BN. The algorithm developed is then used as the basis for a prototype support tool, and we discuss the implementation of this tool.

We evaluate the new techniques and tools we have developed by conducting three experimental studies, using a series of compositional models that range from using 10 entities to 233 entities. We compare the performance of the first experiment to the performance of three existing attractor identification tools: *BoolNet* [55], *BNS* [35] and *BoolSim* [56]. Finally, we utilise our tool to identify attractors of an existing BN model [7] for the signalling network underpinning *T cell large granular lymphocyte (T-LGL) leukemia* [54].

This chapter is organised as follows. In Section 5.2, we formulate an extended algorithm for compositionally identifying attractors based on the approach presented in Section 4.3, and present our approach for generating the set of aligned tuples and computing the interference aligned next state tuple. We then briefly discuss the implementation of the tool support and improving its efficiency. In Section 5.3, we evaluate the tool by performing a range of performance tests. We start with our test model and then we present the results and performance evaluation. We then compare our approach to three existing tools for attractor identification using the last 16 tests performed in the first experiment. In Section 5.4, we illustrate the practical application of our techniques for analysis with a case study.

5.2 Developing Tool Support

Objective 3 is very important in ensuring that we develop tool support for our developed results. Hence, in this section, we take the theoretical approach outlined in the previous chapter and use it as the basis to extend the algorithm introduced in Section 3.3. The extended algorithm identifies attractors compositionally in a generalised form of a composed model. We then use this to develop a prototype support tool and in particular, consider efficiently addressing the key challenges that arise.

5.2.1 Algorithm for Compositionally Analysing Attractors

The theoretical approach we presented was based on identifying cyclic paths over the SCCs in the subnetworks' interference state graphs. In each step, we ensure that the states align over the entities to be merged, and that there is at least one normal step for one of these merged entities. In the algorithmic approach we consider constructing cyclic paths one step at a time. We focus on tuples of states over the SCCs which align (i.e. they have the same state values on the entities that are merged), and consider how to make a step from one aligned tuple of states to the next in order to ensure that well-defined interference aligned cyclic paths are constructed.

More formally, let $\Sigma = (M, E)$ be an arbitrary composition with subnetworks $M = \{\mathcal{BN}_1, \ldots, \mathcal{BN}_n\}$. Then, we say that a state tuple $(S_1, \ldots, S_n)$, for $S_1 \in S_{\mathcal{BN}_1}, \ldots, S_n \in S_{\mathcal{BN}_n}$ is an *aligned state tuple (for Σ)* iff for each $g \in gc(\Sigma)$ we have $S_{\lambda(g)}(g) = S_{\lambda(h)}(h)$, for all $h \in \Delta(g)$.

When making a transitional step to a new aligned state tuple we need to ensure that at least one normal step occurs for each composed entity in order ensure that the step is realisable in practice. This leads to the following definition of an *interference aligned next state tuple*, an adaptation of the definition of interference alignment on paths (see Definition 19).

Definition 20. *Suppose $S_i \xrightarrow[gc(\Sigma,\mathcal{BN}_i)]{\mathcal{BN}_i} S_i'$ for some $S_i, S_i' \in S_{\mathcal{BN}_i}$, for $i = 1, \ldots, n$. Then we say $(S_1', \ldots, S_n')$ is an* interference aligned next state tuple *for $(S_1, \ldots, S_n)$ iff for every $g \in gc(\Sigma)$ we have:*

1) $S_{\lambda(g)}(g) = S_{\lambda(h)}(h)$, for all $h \in \Delta(g)$; and

2) There exists $h \in \Delta(g)$ such that $S_{\lambda(h)} \xrightarrow[gc(\Sigma,\mathcal{BN}_{\lambda(h)})]{\mathcal{BN}_{\lambda(h)}} S'_{\lambda(h)}$ is a normal step for h.

Interestingly, it can be shown that for any state tuple $(S_1, \ldots, S_n)$, there is at most only one interference aligned next state tuple.

Lemma 14. *Let $(S_1, ..., S_n)$ be a state tuple for Σ, where $S_i \in S_{\mathcal{BN}_i}$, for $i = 1, ..., n$. Then $(S_1, ..., S_n)$ has at most one interference aligned next state tuple.*

Proof. Let $(S_1, ..., S_n)$ be a state tuple, where $S_i \in \mathcal{S}_{\mathcal{BN}i}$, for $i = 1, ..., n$. Assume $(S_1', ..., S_n')$, where $S_i' \in \mathcal{S}_{\mathcal{BN}i}$, is the interference aligned next state tuple for $(S_1, ..., S_n)$.

Then, we need to show that the interference aligned next state tuple $(S'_1, ..., S'_n)$ is unique.

To show that this interference aligned next state tuple is unique, it suffices to show that for every $g \in gc(\Sigma)$, the state $S'_{\lambda(g)}(g)$ is unique. Let $g \in gc(\Sigma)$ and (WLOG) assume that $S_{\lambda(g)} \xrightarrow[gc(\Sigma, \mathcal{BN}_{\lambda(g)})]{\mathcal{BN}_{\lambda(g)}} S'_{\lambda(g)}$ is a normal step for g as required by definition of the interference aligned next state tuple. Then, we have two cases to consider for the value of $S'_{\lambda(g)}(g)$ using the key properties given in Definition 20:

Case 1: Suppose $S'_{\lambda(g)}(g) = 1$. Then, it follows that for any $h \in \Delta(g)$ we must have by the definition of the interference aligned next state tuple

$$S'_{\lambda(h)}(h) = 1$$

Then it is clear that all entities in $\Delta(g)$ have normal steps in the truth values being 1. Therefore there is no other choice in this case.

Case 2: Suppose $S'_{\lambda(g)}(g) = 0$. It follows that for any $h \in \Delta(g)$ we must have by definition of the interference aligned next state tuple

$$S'_{\lambda(h)}(h) = 0$$

Then, there are two possibilities:

1. It was a normal step, in which case there is no other choice for it.
2. It was an interference step, and in that case there is only one possible step for that entity.

Therefore, there is only one value for $\Delta(g)$, then the interference aligned next state tuple has to be unique. □

The SCCs in the interference state graphs can be found using standard algorithms based on depth-first search, such as *Kosaraju's algorithm* [203] and *Tarjan's algorithm* [200], which run in linear time $O(|V| + |E|)$ on a graph (V, E). In order to efficiently deal with these SCCs we create a data structure Φ_Σ which combines all the information needed from the SCCs. For each $\mathcal{BN}_i$ we store in $\Phi_\Sigma[i]$ all the possible states that appear in an SCC associated with $\mathcal{BN}_i$ along with the set of next states to which they

can transition. As an example, consider the Φ_{Ex} presented in Table 5.1 for Σ_{Ex} (see Figure 4.5 for the corresponding SCCs).

Table 5.1 The data structure Φ_{Ex} which contains the information for the SCCs associated with Σ_{Ex} (see Figure 4.5).

BN	SCC	Next
$\mathcal{M}_{Ex1}$	00	00
	01	01, 11
	11	01
$\mathcal{M}_{Ex2}$	00	11
	01	01
	10	10
	11	00
$\mathcal{M}_{Ex3}$	000	000, 001, 010, 011
	001	000, 001, 010, 011
	010	000, 001, 010, 011
	011	000, 001, 010, 011
	100	100

BN	SCC	Next
$\mathcal{M}_{Ex4}$	00	00, 01
	01	10
	10	00, 01
	11	11
$\mathcal{M}_{Ex5}$	00	00
	01	10
	10	01
	11	11
$\mathcal{M}_{Ex6}$	0000	0000, 1000
	0010	0010
	1000	0000, 1000

We can now formulate an algorithm for a function *findAtt*, which takes the SCC data structure Φ_Σ associated with a composition Σ and then returns all the attractors in the resulting merged model. The algorithm works by iterating through the set of all aligned state tuples over Φ_Σ. For each one, it attempts to generate a sequence of interference aligned next state tuples until a repeated aligned state tuple is reached, indicating that a set of interference aligned cyclic paths have been found. If at any point we have an aligned state tuple that has previously been generated, then we skip the current state tuple sequence, since it has already been considered. The pseudo code for this algorithm is given in Algorithm 2, and it makes use of the following functions:

- $alignSet(\Phi_\Sigma)$ returns the set of all aligned state tuples generated by the SCC information in Φ_Σ.
- $doStep((S_1, ..., S_n), \Phi_\Sigma)$ returns the interference aligned next state tuple for a given state tuple $(S_1, ..., S_n)$ based on the SCC information in Φ_Σ. If no interference aligned next state tuple exists, then it returns a *Null* value.
- $extCP(listST)$ merges the list of state tuples $listST$ into a path (which is straightforward as the algorithm ensures that they align), and extracts the attractor in which the path must end.

Algorithm 2: *findAtt*(Φ_Σ)

Input : Φ_Σ : *SCC Data Structure*
Output : *attSet* : *Set of Attractors*
Variables : ST, ST' : *StateTuple*; *listST* : *List of StateTuples*; *seen* : *Set of StateTuples*

```
1  Begin
2      attSet := {}
3      seen := {}
4      foreach ST ∈ alignSet(Φ_Σ) do
5          listST := [ ]
6          Loop
7              if ST ∈ seen then
8                  Exit Loop
9              else
10                 seen := seen ∪ {ST}
11                 ST' := doStep((ST), Φ_Σ)
12                 if ST' = Null then
13                     Exit Loop
14                 else
15                     listST := listST + +[ST]
16                     if ST' ∈ listST then
17                         attSet := attSet ∪ {extCP(listST + +[ST'])}
18                         Exit Loop
19                     else
20                         ST := ST'
21                     end
22                 end
23             end
24         EndLoop
25     end
26     return attSet
27 End
```

To illustrate how the algorithm works consider applying it to the example composition Σ_{Ex} given in Figure 4.3. The SCCs for Σ_{Ex} result in 2880 possible state tuples, but only 60 of these align. Suppose the *findAtt* algorithm has selected the aligned state tuple $ST = (01, 11, 000, 00, 10, 1000)$. Then the following interference aligned next state tuples are generated:

$$\begin{aligned} listST &= [(01, 11, 000, 00, 10, 1000), (11, 00, 010, 01, 01, 0000), \\ &\quad (01, 11, 000, 10, 10, 1000), (11, 00, 010, 01, 01, 0000)] \end{aligned}$$

Applying $extCP$ to $listST$ returns the attractor $[100100000, 010110100, 100100000]$. It can be verified that the algorithm for *findAtt* correctly identifies the nine attractors contained by the composed model $BN(\Sigma_{Ex})$ resulting from Σ_{Ex}.

The overall performance of the proposed algorithmic approach is impacted by the number of subnetworks used, and the number of corresponding SCCs and their size. The algorithm should perform well in practice if a good compositional structure is used, and we explore this experimentally in Section 5.3 using a series of scalable tests. There are two main functions used by the algorithm that need to be carefully considered: 1) generating the set $alignSet(\Phi_\Sigma)$ of all aligned state tuples; and 2) computing the interference aligned next state tuple $doStep((S_1, ..., S_n), \Phi_\Sigma)$ for a given state tuple $(S_1, ..., S_n)$. We now consider how to efficiently implement these key functions.

5.2.2 Generating the Set of Aligned State Tuples

The algorithm for *findAtt* is based on iterating through the set of all aligned state tuples over Φ_Σ. If the number of SCC states for each $\mathcal{BN}_i$ is k_i, for $i = 1, \ldots, n$, then there are $k_1 \times \cdots \times k_n$ possible state tuples to check for alignment. However, the number of actual aligned state tuples is normally much smaller, and it is therefore essential to consider how to efficiently generate the set $alignSet(\Phi_\Sigma)$.

We begin by observing that only the entities merged in the composition need to be considered when checking tuple alignment. For each BN, we can therefore focus on the entities it contains which are used in the composition, and we refer to this as its *key*. For example, in the composition Σ_{Ex} the Boolean network $\mathcal{M}_{Ex4}$ has two key states, 0 and 1, associated with the entity g_2^4 which forms part of the composed entity g_1^c. It can be seen that key state 0 represents two SCC states, 00 and 10, and that key state 1 represents states 01 and 11.

Let $\mathcal{BN}_i$ be a BN in a composition Σ. We let $key(\mathcal{BN}_i) = gc(\Sigma, \mathcal{BN}_i)$ and for any state $S \in S_{\mathcal{BN}_i}$, we let $key(S) = S[key(\mathcal{BN}_i)]$. We let $key(\Phi_\Sigma[i]) = \{key(S) \mid S \in \Phi_\Sigma[i]\}$ be the set of key states it contains. The definition of the SCC data structure Φ_Σ can straightforwardly be extended so that each key state is linked with the corresponding full states in the SCCs with which it is associated.

Since a key state may form part of many states within the SCCs for a BN, we can reduce the number of combinations to check by focusing on key states when computing aligned tuples. It is straightforward then to use the identified key state tuples to generate all the aligned state tuples. The aligned key state tuples are constructed by considering each BN involved in the composition in turn, and by incrementally assigning values to the composed entities. We iterate through a BN's key states recursively, applying the procedure to the next BN entities.

It can be seen that the order in which the BNs are considered can impact the efficiency of this approach. We therefore attempt to maximise the efficiency by using the following heuristic to order the BNs: place the BNs in descending order of how many entities they influence which are involved in the composition. More formally, for each Boolean network $\mathcal{BN}_i$, we calculate its *impact factor* using the following formula:

$$\Sigma_{gc \in \{\Delta(g) \mid g \in gc(\Sigma, \mathcal{BN}_i)\}} |gc|$$

We select the BN with the highest impact factor to be the first one to be considered (if there is more than one BN with the same impact factor, then we simply randomly choose one). We then update the remaining impact factors by subtracting from them the entities they included, which are now covered by the composed entities in the selected BN. We then repeatedly apply the above process to the remaining BNs to generate an order index Or, where $\mathcal{BN}_{Or[i]}$ is the ith BN in the order.

To illustrate this idea, consider applying the above procedure to Σ_{Ex}. The composed entities have the following values: $g_1^c = 4$, $g_2^c = 3$ and $g_3^c = 2$. Consequently, we have the following initial impact factors: $\mathcal{M}_{Ex1} = 4$, $\mathcal{M}_{Ex2} = 3$, $\mathcal{M}_{Ex3} = 4 + 2 = 6$, $\mathcal{M}_{Ex4} = 4$, $\mathcal{M}_{Ex5} = 4+3 = 7$, and $\mathcal{M}_{Ex6} = 3+2 = 5$. We therefore select $\mathcal{M}_{Ex5}$ to be the first BN and then recalculate the impact factors for the remaining BNs by removing the values associated with g_1^c and g_2^c as follows: $\mathcal{M}_{Ex1} = 0$, $\mathcal{M}_{Ex2} = 0$, $\mathcal{M}_{Ex3} = 2$, $\mathcal{M}_{Ex4} = 0$, and $\mathcal{M}_{Ex6} = 2$. Both $\mathcal{M}_{Ex3}$ and $\mathcal{M}_{Ex6}$ now have the same largest impact factor and so we simply choose one to be the next BN, say $\mathcal{M}_{Ex3}$. Now all the remaining BNs have an impact factor of 0 and so we can place them in any relative order. The result is the following processing order: $\mathcal{M}_{Ex5}$, $\mathcal{M}_{Ex3}$, $\mathcal{M}_{Ex1}$, $\mathcal{M}_{Ex2}$, $\mathcal{M}_{Ex4}$, and $\mathcal{M}_{Ex6}$.

The following recursive algorithm (Algorithm 3) uses the order index Or and the ideas above to recursively identify the collection of the aligned states' values of the composed entities T. The algorithm makes use of the following two functions:

- The function $cons(k, i, Or, A)$ checks whether the key state k for $\mathcal{BN}_{Or[i]}$ is consistent with the corresponding values already assigned to the composed entities.
- The function $set(A, key(\mathcal{BN}_{Or[i]}), k)$ updates A by setting the values given by k for any composed entities covered by $\mathcal{BN}_{Or[i]}$ whose values have not yet been set.

Algorithm 3: $genAST(i, \Phi_\Sigma, Or, A, t, T)$

Input : $i : Nat, \Phi_\Sigma : SCC\ Data\ Structure, Or : OrderIndex,$
$A : Key\ States, t : Nat$
Output : $T : Array\ of\ Aligned\ Key\ States$
Variables : $k : KeyState$

```
1  Begin
2      foreach k in key(Φ_Σ[Or[i]]) do
3          if cons(k, i, Or, A) then
4              set(A, key(BN_{Or[i]}), k)
5              if i < (n − 1) then
6                  genAST(i + 1, Φ_Σ, Or, A, t, T)
7              else
8                  T[t] = A
9                  t = t + 1
10             end
11         end
12     end
13 End
```

The set of aligned state tuples can be generated straightforwardly using a recursive based algorithm based on the collection T.

To illustrate the above idea, consider applying the above processes to Σ_{Ex}. The resulting T with the aligned states for the composed entities is shown in Table 5.2. Then, we use T to generate aligned key states, and each key state is associated with a number of full states. For example, the key state 00 in the submodel $\mathcal{M}_{Ex3}$ is associated with the two full states 000 and 100. The total number of the aligned state tuple for the first row is 32 tuples, because the key states of $\mathcal{M}_{Ex3}$, $\mathcal{M}_{Ex1}$, $\mathcal{M}_{Ex2}$, $\mathcal{M}_{Ex4}$ and

$\mathcal{M}_{Ex6}$ are associated with two full states, while the key state of $\mathcal{M}_{Ex5}$ is associated with only one full state (see Figure 4.5).

Table 5.2 Summary of the generation of aligned state tuples for Σ_{Ex}. The first column shows the resulting T , the second column shows the aligned key states based on T, and the final one illustrates the total number of aligned state tuples for each row.

T			Aligned key states						No of aligned state tuples
g_1^c	g_2^c	g_3^c	$\mathcal{M}_{Ex5}$	$\mathcal{M}_{Ex3}$	$\mathcal{M}_{Ex1}$	$\mathcal{M}_{Ex2}$	$\mathcal{M}_{Ex4}$	$\mathcal{M}_{Ex6}$	
0	0	0	00	00	0	0	0	00	32
1	0	0	01	10	1	0	1	00	8
0	1	0	10	00	0	1	0	10	16
1	1	0	11	10	1	1	1	10	4

5.2.3 Computing Interference Aligned Next State Tuples

The *findAtt* algorithm attempts to generate a sequence of interference aligned next state tuples for each aligned state tuple until a repeated sequence occurs, which indicates that the set of interference aligned cyclic paths has been identified. Each state included in the tuple must have one or multiple next states in its associated SCC. Thus, if the number of next states for each state in the tuple $(S_1, ..., S_n)$ is k_i, for $i = 1, ..., n$, then there are $k_1, \times ... \times k_n$ possible next state tuples that need to considered in the algorithm. **Lemma 14** has shown, that, for any state tuple there is at most one interference aligned next state tuple. Therefore, the function $doStep((S_1, ..., S_n), \Phi_\Sigma)$ needs to be efficiently implemented.

We can focus on the composed entities to generate the next state tuple because only their states can vary in a step. It turns out that for any BN, for each of its entities used in the composition there are only four possibilities for its next state and we can label those possibilities. By examining the labels, we can then determine what the next state for composed entities should be, and from this we generate the next state tuple. We now formalise this idea below.

For each state S_k for a Boolean network $\mathcal{BN}_i$ given in Φ_Σ and $g \in gc(\Sigma, \mathcal{BN}_i)$, there are four possibilities for the next step for g (see Figure 5.1):

1. The state of g is updated to 0 (i.e. $S_{k+1}(g) = 0$) using a normal step.

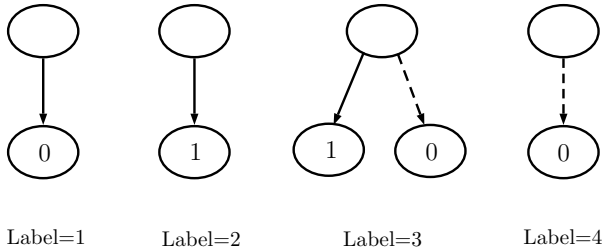


Figure 5.1 Labelling the four possibilities for the next state in the SCCs.

2. The state of g is updated to 1 (i.e. $S_{k+1}(g) = 1$) using a normal step and interference is not possible.

3. The state of g is updated to 1 (i.e. $S_{k+1}(g) = 1$) using a normal step or it can be updated to 0 (i.e. $S_{k+1}(g) = 0$) using an interference step.

4. The state of g is updated to 0 (i.e. $S_{k+1}(g) = 0$) using an interference step and no normal step can be applied.

Our interesting idea to compute the interference aligned next state tuple starts with pre-processing the states in the SCCs by labelling them with one of the four possible next state categories listed above. We label the states before running the $findAtt$ algorithm. As an example, Table 5.3 presents the assigned labels for each state in the SCCs data structure Φ_{Ex} containing the information for the SCCs associated with Σ_{Ex} (see Figure 4.5).

The crucial idea is that counting the number of labels across all BNs for a particular composed entity can be used to deduce the states of the composed entities in an interference aligned next state tuple. We let $\Delta(\Sigma) = \{\Delta(g) \mid g \in gc(\Sigma)\}$ be the set of all composed entities that occur in the composed model. Let $gc \in \Delta(\Sigma)$, then $count(gc)$ is an array with four elements, and each position refers to the count representing each category of the next state's labels. The idea is that $count(gc)[i]$ counts the number of occurrences of label i, for $i = 1, 2, 3, 4$, for each $h \in gc$. We iterate through BN labels in the SCC data structure and increment the count for each composed entity based on the categories identified.

We can use these counts $count(gc)$ to determine the value of the composed entity gc for the interference aligned next state tuple. According to the results of counting the labels, we have four scenarios:

1. When $count(gc)[1] > 0$ and $count(gc)[2] = 0$, then the next state value of g will be 0. It can be noticed that $count(gc)[1] + count(gc)[4] = k$, where k is the total

Table 5.3 Pre-processing step of labelling each state in the SCCs data structure Φ_{Ex} which contains the information for the SCCs associated with Σ_{Ex} (see Figure 4.5).

BN	SCC	g_1^c	g_2^c	g_3^c	Next
$\mathcal{M}_{Ex1}$	00	4			00
	01	3			01, 11
	11	1			01
$\mathcal{M}_{Ex2}$	00		2		11
	01		1		01
	10		2		10
	11		1		00
$\mathcal{M}_{Ex3}$	000	3		3	000, 001, 010, 011
	001	3		3	000, 001, 010, 011
	010	3		3	000, 001, 010, 011
	011	3		3	000, 001, 010, 011
	100	1		1	100
$\mathcal{M}_{Ex4}$	00	3			00, 01
	01	1			10
	10	3			00, 01
	11	2			11
$\mathcal{M}_{Ex5}$	00	1	1		00
	01	1	2		10
	10	2	1		01
	11	2	2		11
$\mathcal{M}_{Ex6}$	0000		3	1	0000, 1000
	0010		1	1	0010
	1000		3	1	0000, 1000

number of $\Delta(gc)$. If there is one normal edge that steps to 0, then the value of gc has to be 0 in the aligned next state tuple, if it exists. All of the other edges can be interference aligned with it because they have either normal steps to 0 or interference edges.

2. When $count(gc)[1] = 0$ and $count(gc)[4] = 0$, then the next state value of gc will be 1. This is equivalent to the scenario when $count(gc)[2] + count(gc)[3] = k$, where k is the total number of $|\Delta(gc)|$.

3. When $count(gc)[1] > 0$ and $count(gc)[2] > 0$, then there is an inconsistent situation and no possibility for generating an interference aligned next state tuple. Although we require at least one step to be updated to 0 for gc, the other edges must be updated to one.

4. When $count(gc)[1] = 0$ and $count(gc)[4] > 0$, then there is an inconsistent situation, and no possibility for generating an interference aligned next state tuple. This is because the interference required to create a 0, is missing since $count(gc)[1] = 0$.

The above conditions indicate when an interference aligned next state tuple can potentially occur, and what values the associated composed entities will have in it. The other entities that are not used in the composition will have the same values for the next states as in their submodels. Thus, we can focus on the composed entities because only composed entities' states can be changed due to interference. Having these values of composed entities allows us to calculate the interference aligned next state tuple.

It is important to note that, in some cases, the interference aligned next state tuple will not be possible due to the interaction between different composed entities. As an illustrative example, consider applying our algorithm to three BNs composed on $g_1^c = \{g_1^1, g_1^2\}$ and $g_2^c = \{g_2^2, g_1^3\}$, and their SCCs as represented in Figure 5.2. Suppose we start from the three aligned states $(00, 00, 00)$ and $count(g_1^c) = [1, 0, 1, 0]$ and $count(g_2^c) = [0, 1, 1, 0]$, then the composed entities have the following values: $g_1^c = 0$ and $g_2^c = 1$. Because g_1^c in the first state is updated to 0 as a normal step and updated to 1 in the second state, but has an interference step, we can set the value of g_1^c to 0. By contrast, g_2^c is updated to 1 in the second state, but has an interference step to 0 and is updated to 1 in the third state, so we can set the value of g_2^c to 1. The problem is that we need to use two different edges in the $\mathcal{BN}_2$, and thus cannot generate the next state tuple.

As a result, the above process has been used to determine the values of the composed entities, but the results of the interference aligned next state tuple will not be feasible because of the interaction in different composed entities. To take account of this, we need to perform a final check upon generating the next states tuple to check the interaction between entities. To do so, we pre-process key states for the next states in the SCC data structure Φ_{Ex} before ahead. During the algorithm, we generate key states for the next state for each BN using composed model values, and determine whether the key states already exist in the SCC data structure.

To illustrate the idea, consider applying it to the example composition Σ_{Ex} given in Figure 4.5. Suppose the algorithm selected the aligned state tuple $ST = (01, 11, 000, 00, 10, 1000)$. There are 64 possible next state tuples for this tuple, but

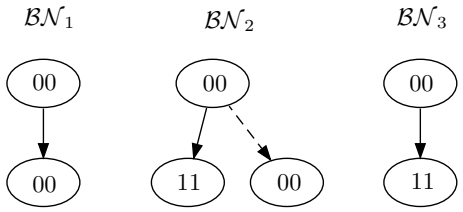


Figure 5.2 Example of the next steps for the three aligned states 00, 00 and 00 in the SCCs of three Boolean networks $\mathcal{BN}_1$, which has one composed entity g_1^1, $\mathcal{BN}_2$, which has two composed entities $\{g_1^2\}$ and $\mathcal{BN}_3$, which composed on g_1^3. It assumes that its composed model entities are $g_1^c = \{g_1^1, g_1^2\}$ and $g_2^c = \{g_2^2, g_1^3\}$.

only one tuple is an interference aligned next state tuple. The following steps are used in our algorithm to compute the interference aligned next state tuple:

1. **Count labels for each composed entity based on Table 5.3:**

$$count(g_1^c) = [0, 1, 3, 0]$$

$$count(g_2^c) = [2, 0, 1, 0]$$

$$count(g_3^c) = [1, 0, 1, 0]$$

2. **Set composed entities' values based on the conditions:**
   – $count(g_1^c)[1] = 0$ and $count(g_1^c)[4] = 0$, then the next state value of g_1^c will be 1.
   – $count(g_2^c)[1] > 0$ and $count(g_2^c)[2] = 0$, then the next state value of g_2^c will be 0.
   – $count(g_3^c)[1] > 0$ and $count(g_3^c)[2] = 0$, then the next state value of g_3^c will be 0.

3. **Final Check:**
   The set of key states for the next states is $\{1, 0, 10, 1, 10, 00\}$, which given by the composed entities' values, are exist in the SCC state structure $\Phi_{E_{Ex}}$. Thus, the interference aligned next state tuple is $(11, 00, 010, 01, 01, 0000)$.

5.2.4 Complexity Analysis

In this section, we discuss the complexity of the algorithm presented in Section 5.2.1. We highlight key areas that impact the performance of the algorithm and consider worst-case situations. A detailed experimental evaluation of the tool is conducted in Section 5.3 to provide further insight into its performance.

The overall performance of the algorithm is impacted by the size of SCCs associated with a composition Σ and the number of entities that are composed. We discuss this in more detail by considering each main part of the algorithm below.

- *Constructing the SCC data structure* Φ_Σ: The SCCs in the interference state graphs can be identified by using standard algorithms based on a depth-first search, such as *Kosaraju's algorithm* [203] and *Tarjan's algorithm* [200], which run in linear time $O(|V| + |E|)$ on a graph (V, E). The worst-case scenario for the algorithm is when all sub-networks states are involved in the SCCs. A state graph has 2^k states and edges, where k is the number of entities in a BN. The worst-case for the number of edges in the interference state graph is that each composed entity state is updated to one. Then, we need to add 2^j edges, where j is the number of composed entities in a BN. For example, suppose a BN has three entities and two entities involved in a composition, and the state of the composed entities is updated to one in all global states in the state graph; then, each state needs to be updated to three different states, and the total number of newly added edges will be $8 \times 3 = 24$. The total number of edges in the interference state graph for that BN would be 24 (i.e. interference edges) + 8 (i.e. normal edges) = 32, which is equivalent to 2^{3+2}.

  Suppose $k \in \mathbb{N}$ is an upper bound for the number of entities in the Boolean network sub-models and j is the upper bound of the number of composed entities in the Boolean network sub-models. Then, the upper bound of the total number of edges in an interference state graph is 2^{k+j}, while the upper bound of the number of edges in the worst-case scenario considered for m BNs will be $2^{k+j} \times m$.

- $alignSet(\Phi_\Sigma)$: This part depends on the set of key states associated with the global states in the SCC data structure Φ_Σ. The worst-case occurs when the extracted key states are equivalent to the number of global states of the SCCs. Note that involving all entities in a BN in a composition is unlikely to happen in practice. Based on this worst-case scenario, the complexity of ordering BNs increases when the number of composed entities resulting in the final model increases. Note that this process still maximises the efficiency of the algorithm.

  Suppose we have a composition $\Sigma = (M, E)$, where $M = \{\mathcal{BN}_1, \ldots, \mathcal{BN}_n\}$ is a set of BNs for some $n \in \mathbb{N}$, $n > 1$. The possible number of the state tuples is $l_1 \times l_2 \times ... \times l_n$, where l_i is the number of states for each $\mathcal{BN}_i$.

  Recall that the definition of the set of all composed entities that occur in the composed model is $\Delta(\Sigma) = \{\Delta(g) \mid g \in gc(\Sigma)\}$, while the definition of the set

of all entities used in the composition is $gc(\Sigma) = gc(\Sigma, \mathcal{BN}_i) \cup \ldots \cup gc(\Sigma, \mathcal{BN}_n)$. Let X be the total number of entities in the sub-models defined by

$$X = \Sigma_{i=1,\ldots,n} \, |G_i|$$

and let Q be the number of entities in the sub-models involved in the composition minus the number of compositions defined by

$$Q = |gc(\Sigma)| - |\Delta(\Sigma)|$$

It is clear that 2^X is the upper bound of the number of possible aligned state tuples if we do not have composed entities. In a composition, we need to take out the number of constrained entities Q from X, which are the number of composed entities that have been constrained with a chosen value in the $getAST$ algorithm. In this algorithm, for each composed entity in $\Delta(\Sigma)$, we know that one of its merged entities will not be constrained. This means that $|\Delta(\Sigma)|$ entities involved in the composition are not constrained and this result in the given definition for Q. Thus, we subtract the $|\Delta(\Sigma)|$ from the $|gc(\Sigma)|$.

The upper bound of the number of the aligned state tuples used in the *findAtt* algorithm is

$$2^{X-Q}$$

The *findAtt* algorithm iterates through the collection of aligned state tuples to construct the interference aligned cyclic paths in sub-models. The algorithm visits each aligned state tuple only once, as the *seen* list records all visited tuples during the algorithm's execution. As a result, the total number of those tuples greatly affects the algorithm's performance in large models. Note that though the above formula considers all of the global states in each BN, in practice, the SCCs are subgraphs resulting from the interference state graph that do not contain all of the global states in the state graphs.

- $doStep((S_1, ..., S_n), \Phi_\Sigma)$: The counting labels process is bounded by the number of composed entities $gc(\Sigma)$ since we focus on them to generate the next state tuple. The worst case of counting the labels for each composed entity is that the composed entity involves merged entities from each BN. However, the counting label is performed linearly by incrementing the count for each composed entity based on the categories identified.

- $extCP(listST)$: The worst case occurs when all the entities are involved in the composition, which increase the complexity of merging the states of the composed entities to generate the composed model attractors.

5.2.5 A Prototype Support Tool

In order to make our approach practical, the algorithm $findAtt$ has been used as the basis for developing a prototype support tool for compositionally identifying attractors in a generalised form of a composed model. The tool is implemented in Python and makes use of the *NetworkX package* [201], which has tools to represent and manipulate network structures. The support tool reads in state graphs for the subnetworks using the *DOT* file format [53], the merged entities of each submodel and the set of all composed entities as inputs. Then, the tool performs several functions to generate attractors. Figure 5.3 demonstrates the logical implementation steps for identifying attractors[1].

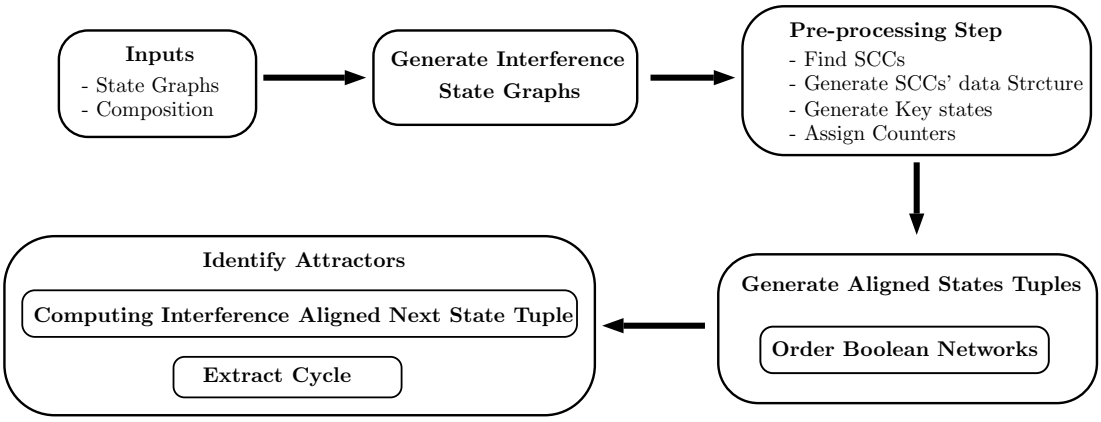


Figure 5.3 Logical implementation steps of the tool.

The following section describes the implementation logic of the tool by listing the main steps to identify attractors compositionally:

1. **Inputs**

   The inputs to our tool support include:

   - Submodels' state graphs: the state graphs are in a DOT format, which can be generated by existing tools such as *GinSim* [71, 125, 3] and *PNST* [49].

[1]For more information about the tool and to obtain a copy, please email the authors.

- BN's composed entities: we need to add information on the composed entities for each BN, such as their IDs and their positions within the state.
- Composition: we need to specify a set of composed entities that are merged together.

2. **Generate interference state graphs**
   Based on the state graphs and BN's composed entities inputs, we add interference edges to state graphs. Edges have properties such as normal or interference steps, and the composed entities that experience interference.

3. **Pre-processing step**
   These functions are required for the next steps:

   - Find SCCs: we find the SCCs for each interference state graph using an exiting function in *NetworkX* package.
   - Create SCCs' data structure: for each BN, we store the next state edges for each state in its SCCs.
   - Generate key states: we generate key states that are linked to full states in the SCCs they are associated with. Key states are used to generate aligned state tuples.
   - Set the labels for each state in SCCs: using SCC's data structure, for each state, we label the state with one of the four categories to be used later in the computing interference aligned next state tuple.

4. **Order Boolean networks**
   This step orders BNs based on the impact factor formula, and it has the following functions:

   - Set up initial impact factor: to calculate the impact factor for each BN based on the formula introduced in Section 5.2.2. Then, we store the BN that has the highest impact factor in *Or*.
   - Update impact factor: we repeatedly update the impact factors to store the BN that has the highest impact factor. We stop when the highest impact factor is zero, then we add the remaining BNs, which have an impact factor of 0 in any relative order.

5. **Generate aligned state tuples**
   This step uses the ordered BNs stored in *Or* to generate aligned key state tuples,

which are stored in T. It is based on Algorithm 3 (see Section 5.2.2), and it has the following functions:

- Create A and assign -1 to each composed entity value.
- Check consistency: to check whether the key state for each BN is consistent with the corresponding values in A.
- Set A: to update A by setting the values given by the key state for any composed entity in the BN.

We use T to generate aligned state tuples recursively using the following functions:

- Generate key states from A: to generate key states for each BN using A's value in T.
- Get full states: to retrieve the corresponding full states that we already generated in Step 2 based on the generated keys in the previous function.

6. **Find attractors**
   This step returns a set of attractors, which is based on Algorithm 2 (see Section 5.2.1). For each aligned state tuple, it attempts to generate a sequence of interference aligned next state tuples until the repeated tuple is reached, indicating that a set of interference aligned cyclic paths has been found. This step uses the following functions:

   - **Get interference aligned next state tuple**
     - Increment counters values: to increment the counter for each composed entity based on the labels that have already been set for each state in Step 2.
     - Generate composed entities' values from counters: to generate a value for each composed entity based on the counters.
     - Generate key states: to generate a key state for each BN from the generated values.
     - Get full states: to retrieve the corresponding full states that we already generated in Step 2 based on the generated keys in the previous function.
   - **Extract cycle**: to extract cyclic paths from interference aligned next state tuples. Then, we merge state tuples on the composed entities to form an attractor and add it to a set including all the attractors.

5.2.6 Improving the Efficiency of the Tool

During implementation, we faced several challenges with regard to improving the efficiency of the code. Initially, the tool was inefficient, and the results produced were poor. To address this, we had to carefully consider each component of the tool, and refine the implementation of the functions to improve their performance. This process definitely needs to be continued, and further improvements can be considered in the future.

One of these challenge was identified in the *genAST* recursion function. In response, we need to undo the change in A made by the set function in line 4. As, when the recursion ends and returns after the call at line 6, we want the previous values of A to be processed for a new key. Both the actual and formal parameters of A refer to exact locations; any changes made in A are reflected in the actual A parameter of the *genAST*. Thus, we can either make a copy of A and send it after the recursion ends, or we need some way of undoing any changes that we have made. Otherwise, if we set values in A, then those values will remain there when the algorithm returns from recursion. We have chosen to make a copy of the last change of A and save it in B, so that it will be sent when the recursion returns in line 6. To do that, we have used the built-in *copy.deepcopy*() function in Python. However, during testing, we discovered that the function was externally slow. Thus, we defined our copy method, and the performance improved as a result. Nevertheless, the copying solution used is inefficient, and we intend to identify alternative solutions in the future.

Yet another challenge faced was that computing interference aligned next state tuples was a slow process. Initially, the function had to call another function to set the counters for the subsequent state edges, in order to identify the values of the key states of the interference aligned with the next state tuple. However, this turned out to be problematic, and had an impact on the efficiency of the tool. We conducted a careful investigation, and this problem was fixed by setting the labels for the states in the sub-models' SCCs and returning their values later.

Adding interference edges to state graphs is done independently for each state graph. The function could decelerate the performance of the tool with a large number of submodels. One option to improve the function's effectiveness, which can be investigated in the future, is using multi-threading to add state edges in parallel. Another option is to build a library for preprocessed models that already have their interference state graphs, so that they can be simply read in the tool.

5.3 Testing and Evaluating the Tool

In this section, we evaluate our tools and techniques for identifying attractors in multiple BNs by performing a series of performance tests. We conduct three experiments using compositional models that employed between 12 and 233 entities to investigate when the tool performs effectively and when it does not and how the tool performs with a big size of attractors. The performance results of the first experiment from Test 5 to Test 20 are then compared to results obtained using *BoolNet* [55], *BNS* [35] and *BoolSim* [56].

In order to allow a range of model sizes to be considered, we constructed 35 models from nine artificial BNs by merging entities. Figure 5.4 provides a summary of the submodels $\mathcal{BN}^1_{Test}$, $\mathcal{BN}^2_{Test}$, $\mathcal{BN}^3_{Test}$, $\mathcal{BN}^4_{Test}$, $\mathcal{BN}^5_{Test}$, $\mathcal{BN}^6_{Test}$, $\mathcal{BN}^7_{Test}$, $\mathcal{BN}^8_{Test}$ and $\mathcal{BN}^9_{Test}$ used in the test. These models were chosen to allow models to be produced, and in particular, some were designed to have complex behaviour such as $\mathcal{BN}^7_{Test}$, and some were chosen to create the big size of attractors ranges between five and 14 states such as $\mathcal{BN}^8_{Test}$ and $\mathcal{BN}^9_{Test}$. The submodels' SCCs, and an overview of the constructed models, are omitted for brevity, and are shown in full in Appendix A.

All the experiments are performed on a laptop that contains an Intel(R) Core(TM) i7-8650U 1.90 GHz processor and 16.0 GB of memory.

5.3.1 Performance Testing

We conducted three different experimental studies to test our developed tool on 35 composed models. We constructed 20 models in the first experiment, using submodels $\mathcal{BN}^1_{Test}$, $\mathcal{BN}^2_{Test}$, $\mathcal{BN}^3_{Test}$, $\mathcal{BN}^4_{Test}$, $\mathcal{BN}^5_{Test}$ and $\mathcal{BN}^6_{Test}$ as depicted in Figure 5.4. In the second experiment, we constructed another 15 models, and we attempted to increase the complexity of the constructed models by using $\mathcal{BN}^7_{Test}$ which has 24 SCC states and a high number of cycles. Furthermore, we attempted to make the tool performs badly by using a poor compositional structure; this was achieved by increasing the number of complex SCCs with several states and cycles and selecting specific merged entities to be used in the composition. In the third one, we constructed six tests to discover how the tool performs when the resulted attractors are big, between five and 14 states.

The results are summarised in Table 5.4 and Table 5.5. Columns 1, 2, 3, 4 and 5 indicate the number of the test, the number of entities in the composed model, the

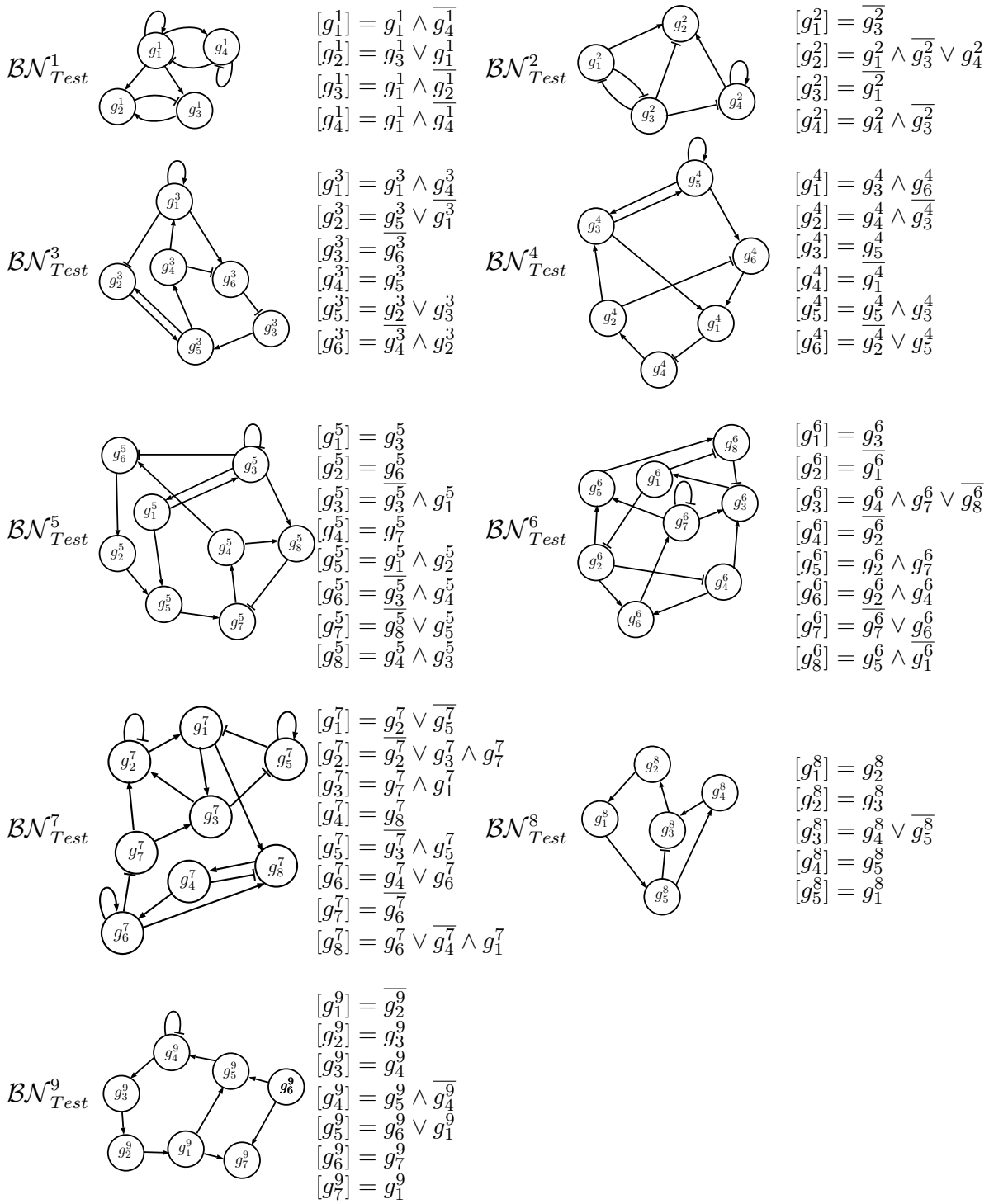


Figure 5.4 A series of test models $\mathcal{BN}^1_{Test}$, $\mathcal{BN}^2_{Test}$, $\mathcal{BN}^3_{Test}$, $\mathcal{BN}^4_{Test}$, $\mathcal{BN}^5_{Test}$, $\mathcal{BN}^6_{Test}$, $\mathcal{BN}^7_{Test}$, $\mathcal{BN}^8_{Test}$ and $\mathcal{BN}^9_{Test}$ which are then composed to create test cases.

number and length of attractors computed by the tool, the number of generated aligned

state tuples, and the number of SCCs' states, respectively. In column 6, we show the runtime of the tool in seconds, which includes all logical steps of the implementation provided in Figure 5.3, except generating interference state graphs step, which needs further work in the future to increase its speed.

Table 5.4 Test Results for the first experiment, where series of test cases created using the test models of $\mathcal{BN}^1_{Test}$, $\mathcal{BN}^2_{Test}$, $\mathcal{BN}^3_{Test}$, $\mathcal{BN}^4_{Test}$, $\mathcal{BN}^5_{Test}$ and $\mathcal{BN}^6_{Test}$.

No	No of entities	No of attractors x length of attractors	No of aligned tuples	Size of SCCs	time, sec
1	12	3 x 1, 1 x 2	5	10	0.003
2	24	2 x 2, 2 x 1	25	19	0.009
3	36	4 x 2	107	28	0.015
4	44	6 x 2	936	54	0.028
5	54	12 x 2	1460	74	0.037
6	62	6 x 2	1400	95	0.044
7	70	14 x 2	1224	97	0.080
8	80	12 x 2	3100	110	0.094
9	90	16 x 2	8100	138	0.182
10	102	24 x 2	15600	151	0.361
11	112	24 x 2	18840	176	0.438
12	122	28 x 2	21744	233	0.575
13	130	32 x 2	14576	267	0.740
14	142	48 x 2	30288	285	1.330
15	152	96 x 2	51480	388	1.594
16	167	32 x 2	70560	460	2.488
17	182	32 x 2	93600	544	3.702
18	197	32 x 2	108000	691	5.592
19	215	24 x 2	129600	848	6.682
20	233	12 x 2	172800	984	9.230

As shown in Table 5.4 and Table 5.5, the number of entities in the first experiment ranges from 12–233 entities and in the second experiment ranges from 14–152. The number of SCC states in the first tables ranges from 10–984 states and in the second one ranges from 30–308. The total number of attractors ranges between 4–96 in the first experiment and between 12–384 in the second one. The number of aligned state tuples in the first table ranged from 5–172800, and the number of states tuples in the second experiment ranged from 32–537600. It can be noted that the difference grows rapidly between aligned state tuples in both tables; for example, in Test 15 in both experiments was 486120. As expected, it can be seen that the runtime for both experiments increased as the number of entities increased.

Table 5.5 Test Results for the second experiment, where series of test cases created using the test models of $\mathcal{BN}^1_{Test}$, $\mathcal{BN}^2_{Test}$, $\mathcal{BN}^3_{Test}$, $\mathcal{BN}^4_{Test}$, $\mathcal{BN}^5_{Test}$, $\mathcal{BN}^6_{Test}$ and $\mathcal{BN}^7_{Test}$.

No	No of nodes	No of attractors x length of attractors	No of aligned tuples	Size of SCCs	Time, sec
1	14	12 x 2	32	30	0.009
2	21	2 x 1, 20 x 2	157	39	0.015
3	38	24 x 2	800	48	0.028
4	46	48 x 2	1800	62	0.052
5	56	69 x 2	5248	86	0.112
6	64	28 x 2, 2 x 4	36480	105	0.472
7	76	48 x 2	38164	116	0.557
8	86	112 x 2	66520	126	1.099
9	90	192 x 2	68628	147	2.003
10	104	192 x 2	148800	163	3.255
11	116	192 x 2	299480	194	6.585
12	126	288 x 2	289408	250	8.125
13	132	192 x 2	362880	258	12.144
14	146	192 x 2	405696	280	14.395
15	152	384 x 2	537600	308	19.819

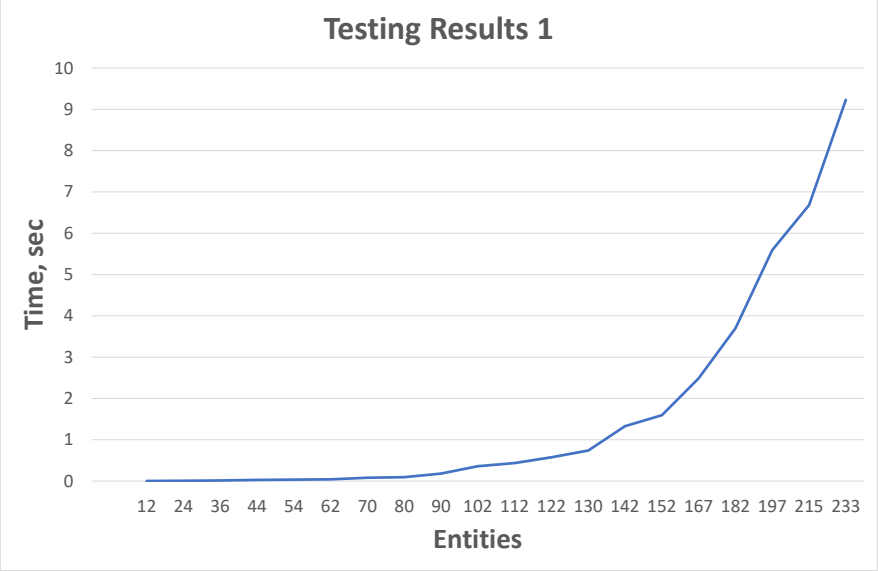


Figure 5.5 Graph to represent the relationship between the number of entities and runtime of the *findAtt* algorithm based on the results summarised in Table 5.4.

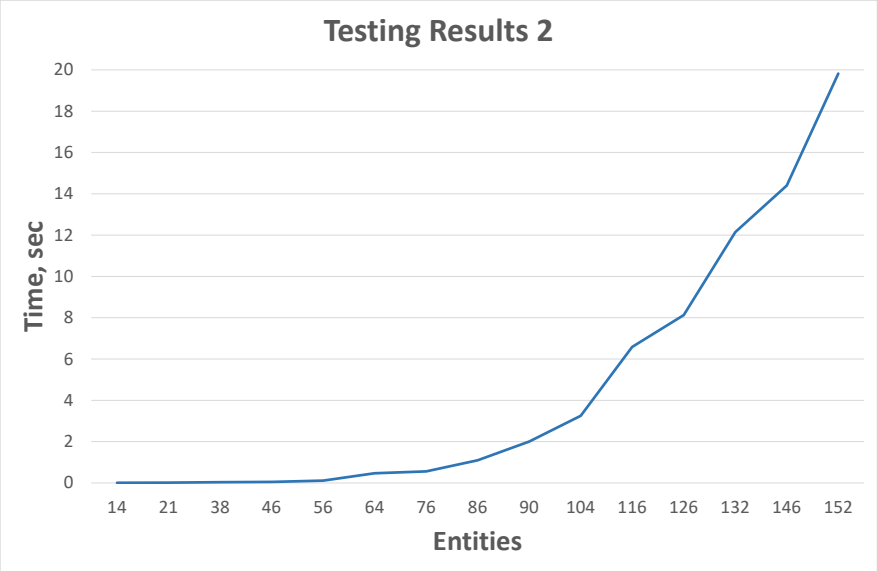


Figure 5.6 Graph to represent the relationship between the number of entities and runtime of the *findAtt* algorithm based on the results summarised in Table 5.5.

When comparing Tables 5.4 and 5.5, our tool performed faster in the first experiment than in the second one; evidently, the complex submodel that we added to the second experiment affected the tool's performance. There were significantly higher numbers of aligned state tuples and attractors in the second experiment, and even the numbers of entities and SCC states were relatively similar in both experiments. This indicates that the performance of our tool was strongly influenced by the structure and size of submodels' SCCs; this is likely due to our method iterating through the set of aligned state tuples, and the effect of SCCs structure when generating a sequence of interference aligned next states tuples to form interference aligned cyclic paths. The relationship between the number of entities and the execution time summarised in Table 5.4 and Table 5.5 is visualised in the graphs presented in Figure 5.5 and Figure 5.6, respectively. It shows that the tool appears to work efficiently, but there is a definite increase in time for the model with a large number of entities.

The set of tests presented in Table 5.6 focuses on identifying the bigger size of attractors than in previous tests, we use the sub-models $\mathcal{BN}^{8}_{Test}$ and $\mathcal{BN}^{9}_{Test}$ alongside the sub-models shown in Figure 5.4 to generate the size of attractors ranges between 5 and 14. A few sub-models used in the previous tests presented in Tables 5.4 and

Table 5.5 have been replaced by the $\mathcal{BN}^8_{Test}$ and $\mathcal{BN}^9_{Test}$ sub-models to generate long attractors. A comparison of the runtime results of those tests to the original ones suggests that the size of attractors does not affect performance. Although the number of entities in Tests 5 and 6 is identical; the tool takes a longer time to identify attractors in Test 6 than in Test 5 due to the complex sub-models used in the test and the large difference in the number of aligned state tuples.

Table 5.6 Test Results for the third experiment, where series of test cases created using the test models of $\mathcal{BN}^1_{Test}$, $\mathcal{BN}^2_{Test}$, $\mathcal{BN}^3_{Test}$, $\mathcal{BN}^4_{Test}$, $\mathcal{BN}^5_{Test}$, $\mathcal{BN}^6_{Test}$, $\mathcal{BN}^7_{Test}$, $\mathcal{BN}^8_{Test}$ and $\mathcal{BN}^9_{Test}$.

No	No of nodes	No of attractors x length of attractors	No of aligned tuples	Size of SCCs	Time, sec
1	25	10 x 2, 2 x 1, 7 x 14	676	54	0.019
2	35	2 x 2, 2 x 10	390	43	0.018
3	47	2 x 2, 1 x 14	10224	62	0.124
4	71	6 x 1, 4 x 5	2400	98	0.067
5	131	16 x 2, 8 x 14	20828	287	1.169
6	131	32 x 2, 32 x 10	369600	286	12.433

The practical experiments above show that the size and structure of the SCCs affect the performance. In addition, the composition structure (i.e. the submodels used in the compositions and their selected entities to be merged) used in the experiments can affect the tool's performance. Although the number of attractors in the larger models in Test 19 and 20 presented in Table 5.4 is less than the number of attractors that resulted in Test 12–18, the former tests are slower than the latter tests due to the number of aligned state tuples and their SCCs size. This situation indicates that the number of aligned state tuples influences the performance in large models. It can be noted that the number of aligned state tuples and identified attractors did not usually increase as the number of entities in the model increased. For example, the number of aligned tuples in Test 12 is higher than the number of aligned state tuples in Test 13, as shown in Table 5.4. This situation was affected by the submodels and the structure of the composition.

To summarise, we applied our tool to a range of constructed model sizes, using artificial BNs to evaluate its performance. The results seem to be very promising, and it appears that the tool will scale to allow large models to be considered. Despite these promising results, the second experiment indicated that further improvements are needed for our tool to be faster when applied to a complex structure of submodels and

compositions. In addition, testing our tool for larger constructed models is desirable in order to further assess the efficiency of this tool.

5.3.2 Performance Comparison

After conducting the experiments described above, we compared the performance of our developed tool to three existing tools utilised for attractor identification: *BoolNet* [55], *BNS* [35] and *BoolSim* [56].

- *BoolNet* [55] is an R package used for the construction, simulation and analysis of BNs. It provides methods to identify and analyse the attractors of synchronous, asynchronous and probabilistic BNs. In the 'exhaustive' attractor identification method, the maximum number of entities allowed is 29. In our experiment, we use the 'sat.exhaustive' method, which is based on the satisfiability problem.
- *BNS* is a tool based on the algorithm presented in [35] for finding Attractors in synchronous BNs. They use MiniSAT SAT-solver [204] in their implementation.
- *BoolSim* is software that provides algorithms based on reduced ordered binary decision diagrams (ROBDDs) to evaluate attractors and perform perturbation experiments on BNs, using synchronous or asynchronous network dynamics [56]. In our experiment, the $BoolSim$ is configured in the synchronous mode.

We demonstrated the efficiency of these tools by conducting experiments on the last 10 constructed models employed in the first experiment.

Table 5.7 reports the runtime in seconds of those tools when finding attractors. Columns 1, 2, and 3 present the number of the test, the number of entities in the composed model, and the number and length of attractors computed by the tool, respectively. To clarify how our tool compares to the existing tools, we include the runtime for $findAtt$, $BoolNet$, $BoolSim$ and BNS to find all attractors in columns 4, 5, 6 and 7, respectively.

The data of Table 5.7 is illustrated in Figure 5.7, where the horizontal axis represents the number of entities in each test, and the corresponding runtime in seconds is on the vertical axis. The results showed that our tool was faster than $BoolNet$ overall. Moreover, $findAtt$ was faster than BNS and $BoolSim$ in Tests 1–13 and faster than $BoolSim$ in Test 15, but the tool was slower than BNS and $BoolSim$ in Test 14 and

Table 5.7 Performance comparison between four tools $findAtt$, $BoolNet$, $BoolSim$ and BNS, where the test models are based on the first experiment.

No	No of nodes	No of attractors x length of attractors	$findAtt$ (time,sec)	$BoolNet$ (time,sec)	BNS (time,sec)	$BoolSim$ (time,sec)
5	54	12 x 2	0.037	0.154	0.166	0.162
6	62	6 x 2	0.044	0.208	0.106	0.187
7	70	14 x 2	0.080	0.361	0.180	0.260
8	80	12 x 2	0.094	0.571	0.183	0.211
9	90	16 x 2	0.182	1.088	0.319	0.329
10	102	24 x 2	0.361	1.313	0.553	0.402
11	112	24 x 2	0.438	1.700	0.628	0.556
12	122	28 x 2	0.575	2.231	0.693	0.919
13	130	32 x 2	0.740	2.451	0.794	1.100
14	142	48 x 2	1.330	2.766	0.954	1.150
15	152	96 x 2	1.594	3.610	1.480	2.170
16	167	32 x 2	2.488	4.243	0.963	1.74
17	182	32 x 2	3.702	5.317	1.2	1.76
18	197	32 x 2	5.592	5.942	1.12	4.32
19	215	24 x 2	6.682	7.605	0.983	6.23
20	233	24 x 2	9.230	6.991	0.547	5.39

slower than BNS in Test 15. We notice that in the larger models in Tests 16–20, BNS and $BoolSim$ are faster than our tool. It appears that BNS is affected more by the number of attractors than the amount of entities in the models, while $BoolSim$ is impacted by the number of attractors and the structure of the models. On the other hand, the runtime of $findAtt$ and $BoolNet$ increase as the number of entities increases.

In general, the results of the comparison indicate that our tool performs extremely well against the three mature tools. The results also indicate that our tool works well with large models, although the outcome was slightly behind that of $BoolSim$ and BNS in Tests 16-20. Thus, the tool's efficiency needs to be further refined.

In the future, we will apply $BoolNet$, $BoolSim$ and BNS to the models in the second experiment in which the structure of SCCs are more complex than in the first experiment, because the performance of solvers relates to the network's structure, and it is not proportional to the number of nodes. A potential future work could involve creating larger models, and applying our tool in those contexts as well. Comparing our tool with decomposition approaches (e.g. [36]) is also planned for the future.

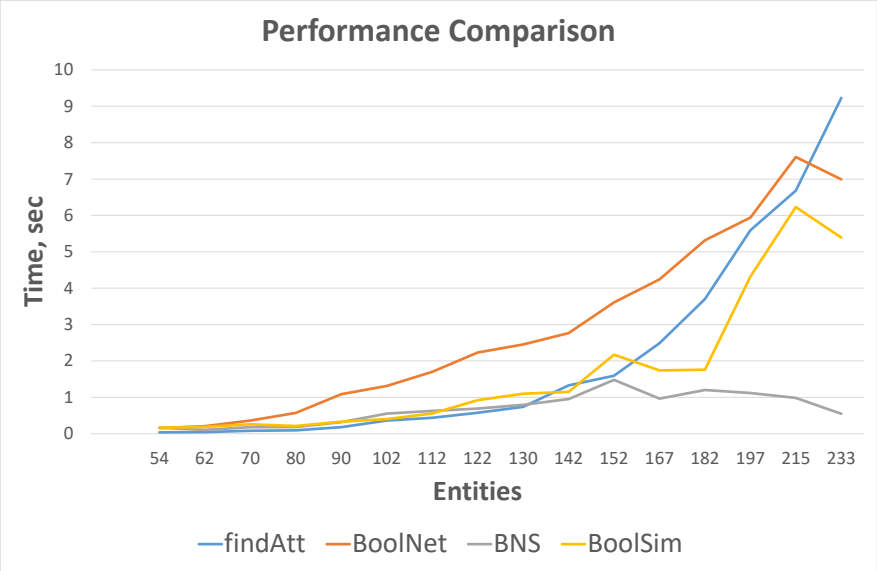


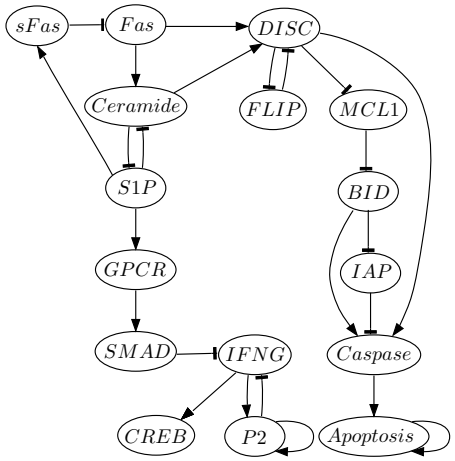
Figure 5.7 Graph to represent the performance comparison between four tools, $findAtt$, $BoolNet$, $BoolSim$ and BNS, based on the results summarised in Table 5.7.

5.4 Case Study

In this section, we illustrate the application of the tools and techniques developed for attractor identification by applying them to an example biological regulatory network. We consider the signalling pathway underpinning T cell surval in *T cell large granular lymphocyte leukemia (T-LGL)* [54], and use a BN based on the model presented in [7]. We begin by decomposing this BN and then applying our techniques to the resulting composition to identify all of the model's attractors.

Our aim here is to simply illustrate the developed techniques for a biological model, and that does not involve discovering new biological insights into the T-LGL pathway. Although the model is small, it shows the practical application of our approach and its correctness. Furthermore, it provides important insight into how a model could be decomposed. At present, we have decomposed the model manually, but in the future, we will examine decomposing models automatically. We conducted the case study in the early stages of our research, and we found it to be insightful. Thus, we have included it here as an important illustration. In the future, we intend to apply our approach to a larger biological model.

A BN, modelling the signalling pathways involved in T-LGL based on the model constructed in [7], is presented in Figure 5.8. The model focuses on T cell survival, and the key output determines whether or not *apoptosis* [54] (cell death) occurs. The model contains 16 entities, and so has 65536 global states.



$$
\begin{aligned}
&CREB = IFNG\\
&IFNG = \overline{(SMAD \vee P2)}\\
&P2 = IFNG \vee P2\\
&GPCR = S1P\\
&SMAD = GPCR\\
&Fas = \overline{sFas}\\
&sFas = S1P\\
&Ceramide = Fas \wedge \overline{S1P}\\
&DISC = (Ceramide \vee (Fas \wedge \overline{FLIP}))\\
&Caspase = ((BID \wedge \overline{IAP}) \vee DISC)\\
&FLIP = \overline{DISC}\\
&BID = \overline{MCL1}\\
&IAP = \overline{BID}\\
&MCL1 = \overline{DISC}\\
&S1P = \overline{Ceramide}\\
&Apoptosis = Caspase \vee Apoptosis
\end{aligned}
$$

Figure 5.8 A Boolean network $\mathcal{BN}_{TLGL}$ modelling the signalling pathways involved in T-LGL based on the model constructed in [7].

In order to apply our tool, we first need to decompose the Boolean network $\mathcal{BN}_{TLGL}$ (Figure 5.8) in to a composition. The BN naturally decomposes into the three subnetworks presented in Figure 5.9, where the entities $S1P$ and $DISC$ in $\mathcal{BN}_{TLGL}$ are decomposed into $S1P_1$, $S1P_2$ and $DISC_1$, $DISC_2$, respectively.

Note that since the entities $S1P_1$ and $DISC_1$ have no inputs in their subnetworks $\mathcal{BN}^1_{TLGL}$ and $\mathcal{BN}^3_{TLGL}$, respectively, we use the constant function which always returns 1. The reason for this approach is that 1 is the identity value for conjunction, and so this ensures the composition of $S1P_1$ and $S1P_2$, and $DISC_1$ and $DISC_2$ results in the original next state functions for $S1P$ and $DISC$, respectively. Indeed, it is straightforward to show that $BN(\Sigma_{TLGL}) = \mathcal{BN}_{TLGL}$. It is important to note that although $S1P_1$ and $DISC_1$ have their next states fixed, we are able to capture all the possible behaviours for these entities due to incorporating interference. Recall that a state transition to 1 can always be interfered with to result in a transition to 0 and this behaviour is captured in a BN's interference state graph.

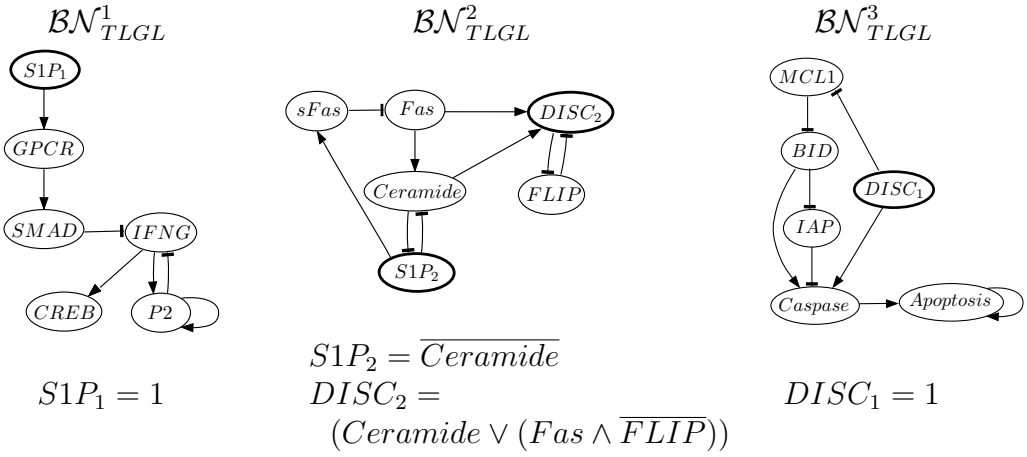


Figure 5.9 The composition Σ_{TLGL} consisting of the three Boolean networks $\mathcal{BN}^1_{TLGL}$, $\mathcal{BN}^2_{TLGL}$, and $\mathcal{BN}^3_{TLGL}$ and the set of composed entities $\{\{S1P_1, S1P_2\}, \{DISC_1, DISC_2\}\}$.

Now, we apply our tool to the submodels' state graphs. The tool then identifies all the SCCs in the interference state graphs for $\mathcal{BN}^1_{TLGL}$, $\mathcal{BN}^2_{TLGL}$ and $\mathcal{BN}^3_{TLGL}$ given the composition Σ_{TLGL}. We summarise information about the SCCs that are found for these subnetworks in Table 5.8, and the SCCs identified for subnetworks are presented in Figure 5.10.

Table 5.8 Summary of the SCCs identified for $\mathcal{BN}^1_{TLGL}$, $\mathcal{BN}^2_{TLGL}$, and $\mathcal{BN}^3_{TLGL}$ based on the composition Σ_{TLGL}.

$\mathcal{BN}^1_{TLGL}$		$\mathcal{BN}^2_{TLGL}$		$\mathcal{BN}^3_{TLGL}$	
SCC	Size	SCC	Size	SCC	Size
φ^1_1	8	φ^2_1	4	φ^3_1	20
φ^1_2	1	φ^2_2	10	φ^3_2	1
		φ^2_3	6		
		φ^2_4	10		

The identified SCCs for Σ_{TLGL} result in 384000 possible aligned state tuples but only 1400 of these align. The *findAtt* function identifies the seven attractors for $\mathcal{BN}_{TLGL}$ as follows: (where the genes are encoded in the following order: $S1P$, $DISC$, $CREB$, $IFNG$, $P2$, $GPCR$, $SMAD$, Fas, $sFas$, $Ceramide$, $Caspase$, FLI, BID, IAP, $MCL1$, $Apoptosis$).

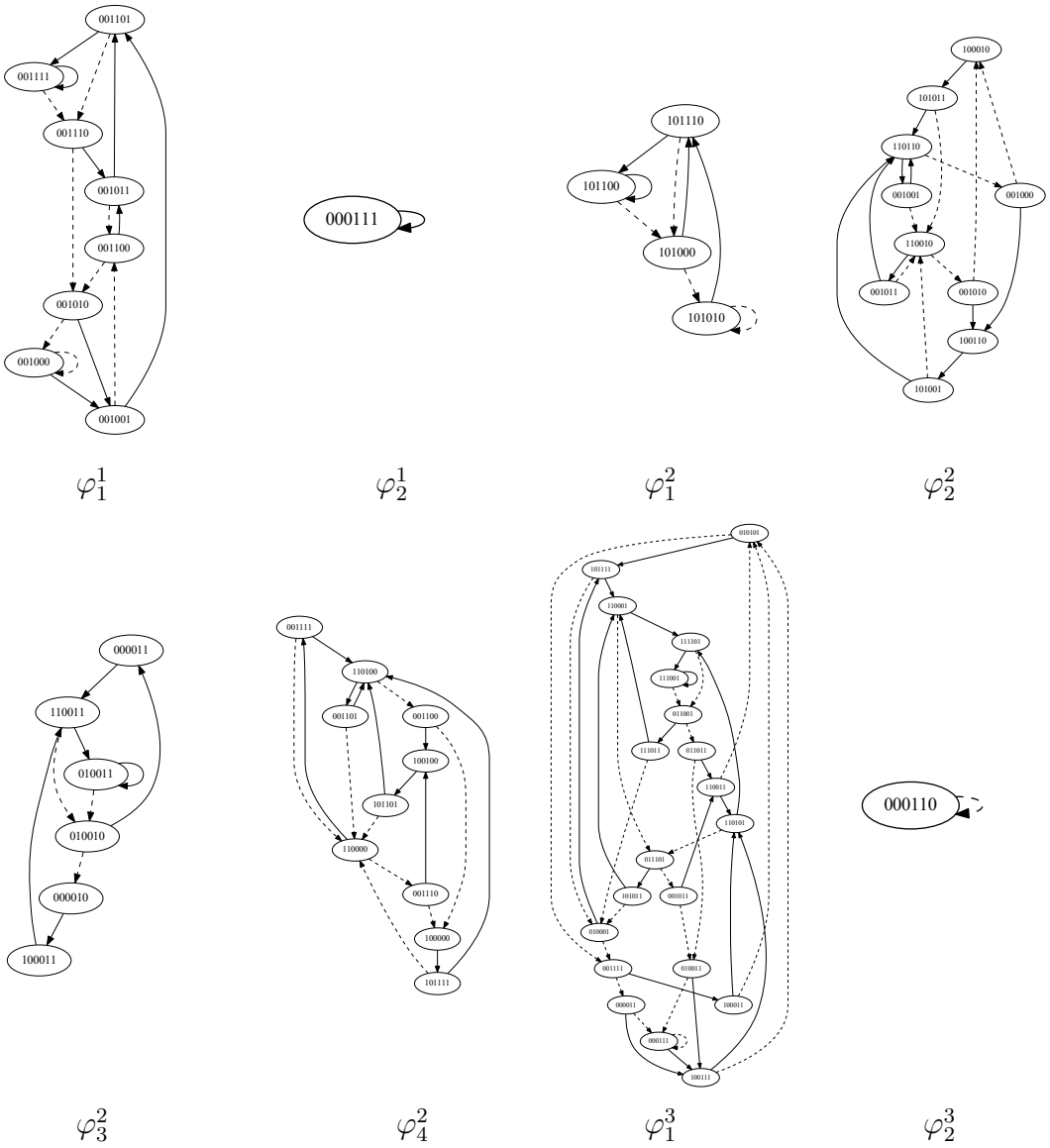


Figure 5.10 The SCCs identified for $\mathcal{BN}^1_{TLGL}$, $\mathcal{BN}^2_{TLGL}$, and $\mathcal{BN}^3_{TLGL}$ based on the composition Σ_{TLGL}.

1) Suppose $ST = (001010, 100100, 110001)$. Then, the following interference aligned next state tuples are generated:

$$\begin{aligned} listST \quad = \quad & [(001010, 100100, 110001), (001001, 101101, 111101), \\ & (001100, 110100, 111001), (001011, 001101, 111001), (001100, 110100, 111001)] \end{aligned}$$

Applying $extCP$ to $list$ returns the attractor
$[0100110110011001, 1100101001011001, 0100110110011001]$

2) Suppose $ST = (001010, 101100, 110001)$. Then, the following interference aligned next state tuples are generated:

$$\begin{aligned} listST &= [(001010, 101100, 110001), (001000, 101100, 111101), \\ &\quad (001000, 101100, 111001), (001000, 101100, 111001)] \end{aligned}$$

Applying $extCP$ to $list$ returns the attractor $[0100100101011001, 0100100101011001]$.

3) Suppose $ST = (001010, 100110, 110001)$. Then, the following interference aligned next state tuples are generated:

$$\begin{aligned} listST &= [(001010, 100110, 110001), (001001, 101001, 011101) \\ &\quad (001100, 110110, 101011), (001011, 001001, 010001) \\ &\quad (001100, 110110, 101111), (001011, 001001, 010001)] \end{aligned}$$

Applying $extCP$ to $list$ returns the attractor $[1000101001010001, 0100110110101111, 1000101001010001]$.

4) Suppose $ST = (001010, 010010, 000111)$. Then, the following interference aligned next state tuples are generated:

$$\begin{aligned} listST &= [(001010, 010010, 000111), (001001, 000011, 000111) \\ &\quad (001101, 110011, 000111), (001111, 010011, 000111) \\ &\quad (001111, 010011, 000111)] \end{aligned}$$

Applying $extCP$ to $list$ returns the attractor $[1000111010100111, 1000111010100111]$.

5) Suppose $ST = (001010, 010010, 000110)$. Then, the following interference aligned next state tuples are generated:

$$\begin{aligned} listST &= [(001010, 010010, 000110), (001001, 000011, 000110) \\ &\quad (001101, 110011, 000110), (001111, 010011, 000110) \\ &\quad (001111, 010011, 000110)] \end{aligned}$$

Applying $extCP$ to $list$ returns the attractor $[1000111010100110, 1000111010100110]$.

6) Suppose $ST = (000111, 100011, 000111)$. Then, the following interference aligned next state tuples are generated:

$$\begin{aligned} listST &= [(000111, 100011, 000111), (000111, 110011, 000111) \\ &\quad (000111, 010011, 000111), (000111, 010011, 000111)] \end{aligned}$$

Applying $extCP$ to $list$ returns the attractor $[1000011010100111, 1000011010100111]$.

7) Suppose $ST = (000111, 100011, 000110)$. Then, the following interference aligned next state tuples are generated:

$$\begin{aligned} listST &= [(000111, 100011, 000110), (000111, 110011, 000110) \\ &\quad (000111, 010011, 000111), (000111, 010011, 000111)] \end{aligned}$$

Applying $extCP$ to $list$ returns the attractor $[1000011010100110, 1000011010100110]$.

5.5 Conclusions

In this chapter, we set out to develop tool support for the new compositional approach of attractor identification detailed in Chapter 4. This involved developing a new algorithm based on extending the initial algorithm presented in Chapter 3. We carefully considered implementing the key functions of generating the set $alignSet(\Phi_E)$ of all aligned tuples and computing the interference aligned next state tuple $doStep((S_1, ..., S_n), \Phi_\Sigma)$ for a given state tuple $(S_1, ..., S_n)$ efficiently. In turn, we developed a prototype support tool based on the algorithm. Then, we evaluated it by undertaking a range of performance tests and comparing it to three existing attractor identification tools. In particular, this chapter covered four main areas, which we discuss below.

5.5.1 Algorithm

Using the definitions and results developed in Section 4.3, we formulated an extended algorithm named $findAtt$ to identify attractors related to the given SCCs in the new formation of a composition. The algorithmic approach uses the new notion of *aligned state tuples (for E)*, which refers to a collection of states for the underlying BNs in the composition that are aligned on composed entities. Given the possibility of having several next states for each global state in each submodel, we generalised a definition of a step based version of *interference alignment*. We proved that we could have at most one interference aligned next state tuple. That outcome provides a basis for developing tool support to compositionally identify attractors.

The crucial step in the developed algorithm is its ability to generate aligned state tuples efficiently. We focused on key states to generate the values of the composed entities, knowing that the heuristic order of BNs would impact the efficiency of our approach.

The counter idea used to compute interference aligned state tuples appears to have improved the algorithm's performance significantly. Adding operations based on the pre-processed state labels seems to be rather quick and that affects the support tool's performance.

The algorithm's overall performance is impacted by the size and the structure of SCCs associated with a composition and the number of composed entities (for more detail, see Section 5.2.4).

5.5.2 A Prototype Support Tool

The support tool was crucial in making the described techniques practical. We implemented the $findAtt$ algorithm using Python by taking submodels' state graphs, and composed entities as inputs and returning the composed model attractors.

Initially, the performance results of the developed tool were poor for large models. We had to improve the tool's efficiency through several iterations. During the test, we noticed that by changing the build-in function $deepCopy()$ in Python to our defined function, the performance of generating aligned state tuples increased. In addition, computing the interference aligned next state tuple involved several steps to improve its performance. Moreover, choosing data structures also required careful consideration during the implementation. Several challenges were discussed in Section 5.2.6.

Further improving the tool's efficiency and features should be considered in the future. For example, we could change the idea of copying A to undoing its change to increase the performance of generating aligned state tuples. In addition, it would be interesting to have a graphical user interface (GUI) to allow users to construct, analyse and visualise models. One constraint in the implementation, which can be relaxed in the future, is the assumption that the composed entity g^c contains one entity from each BN.

5.5.3 Performance Testing and Evaluation

We investigated the practical application of our technique and tools by conducting three experiments, with 31 tests, each using compositional models that range between 12 and 233 entities. The results clearly indicate that our tool performs well, and it appears that it could be scaled to large models. Our tool exhibited greater efficiency in the first experiment than in the second one, which suggests that its efficiency largely relies on the structure and size of the SCCs, the number of aligned states tuples and the composition's structure.

We compared the results of the last 16 tests in the first experiment to results obtained with three advanced tools used to identify attractors in synchronous BNs. Our tool performed well compared with those tools. However, the tests 16–20 indicate that further improvement for the tool's efficiency is needed for large models. The third experiment suggests that the size of attractors does not affect the performance. The structure of the SCCs and the number of aligned state tuples heavily affect the tool's performance.

Despite those promising results, we need to apply our tool to larger composed models and compare the results to the three above mentioned tools. Due to time constraints, we did not consider other tools based on decomposition approaches (e.g. [36]); however, we intend to include them in a performance comparison in the future. We also plan to perform intensive experiments involving large constructed models and real biological networks.

5.5.4 Case Study and Decomposition

We illustrated the application of the developed techniques for compositional attractor analysis by considering a biologically inspired case study based on a BN for the signalling pathway involved in *T cell large granular lymphocyte leukemia (T-LGL)* [7]. The case study considered the early stages of this work, and it provides good insights into how our approach can be used in submodels generated by a decomposition approach. In fact, this case study is small; however, we intend to apply our tool to a large biological network, either by decomposing it into submodels or constructing it from subparts.

In the future, it would be interesting to develop automatic decomposition techniques and tools based on the existing compositional framework. Work has already started

to investigate what could make a good decomposition, and we have developed several metrics to measure the quality of decompositions. Anther issue to be considered concerns ways of partitioning a BN's entities, which involves identifying a suitable set of entities for the partition and, in turn, splitting those entities by decomposing their Boolean functions. That task will likely be challenging and require techniques from graph theory. Another aspect of partitioning entities that warrants consideration is having a BN with a disconnected partition. Decomposing entities' Boolean functions is another important issue, one that may be approached in various ways, including the use of decomposition charts [205] or a binary decision diagram (BDD) [206].

Chapter 6

Concluding Remarks

6.1 Summary

In this thesis, we have set out to extend and strengthen an existing compositional framework for BNs based on using logical connectives to merge entities [48, 49, 4]. One key area to be addressed was attractor analysis to support BNs' practical application. The attractors are the stable and robust operating mechanisms of functioning a cell—such as cell types or cell behavioural modes like growth, quiescence, differentiation and apoptosis—that result from the constraints within the regulatory networks. Due to the importance of identifying attractors in biology [28, 7], we developed theoretical results and techniques to identify attractors compositionally in stages. First, we developed a novel approach for identifying attractors based on merging the cyclic paths identified in each subnetwork's interference state graph. We used SCCs to identity these cyclic paths, and merged them based on a key property called *interference alignment*. We provided a formal correctness argument, where we formally showed that our approach was correct. In our argument, we proved the soundness and completeness of our approach. We formulated an algorithm based on the techniques that we developed for compositional attractor identification. Then, we used the algorithm to develop a prototype support tool. We illustrated the practical application of the developed techniques and tools by applying them to a case study based on a regulatory network for cell differentiation [5, 6] found in the bacteria *Caulobacter crescentus* [199].

In order to make our compositional attractor identification technique more practically applicable, we extended our approach to apply it to an arbitrary form of composition. In fact, the existing framework provides a restrictive way of composing

multiple BNs, and its notations and definitions limit the extension of the results due to their complexity [4]. Therefore, we provide a new formulation and definitions of a composition to allow arbitrary compositions based on an underlying graph structure. This is a significant step forward in the original compositional framework that supports constructing models and decomposing them to aid analysis. Then, we extended our new approach and its results to identify attractors compositionally based on the new formulation. In an arbitrary composition, we analysed multiple interference state graphs that represent interference occurring in a composition to identify SCCs. We formally extended the key property *interference alignment* to be used in merging cyclic paths to identify attractors. We formally proved that the extended approach was correct by showing it is sound and complete.

A significant focus of this work was the development of a prototype support tool for our developed techniques and results in order to automate and evaluate them. We formulated the algorithm $findAtt$ for attractor analysis in an arbitrary composition. We considered efficiently implementing the functions of generating aligned state tuples and computing interference aligned next state tuples. We proved that we have, at most, one interference aligned next state tuple. We implemented the formulated algorithm using Python to automate and evaluate our results related to identifying attractors compositionally in an arbitrary composition. We considered applying our techniques to an existing biological system.

We evaluated the support tool by conducting a series of performance tests. We addressed this by defining nine artificial BNs and using them to construct models ranging from 12 to 233 entities. We conducted three experiments, and, in the second one, we aimed to construct complex models by having SCCs with a high number of cycles. We obtained higher performance results in the first experiment than in the second one. The third experiments attempted to observe the performance of the tool when the generated attractors are in a big size. The test results indicated that our tool was affected by the size and structure of the submodels' SCCs and the number of the aligned state tuples. We noticed also that the submodels' entities used in the composition, and the entities that merged together can affect performance. We compared the runtime of the first experiment to three advanced tools for identifying attractors. Overall, we achieved superior results with our tool, but the last five tests (16 to 20) indicated that more efficiency improvement for the tool is needed.

6.2 What has been achieved

This thesis aimed to develop compositional techniques and tools to support the practical analysis and engineering of BNs by addressing the limitations in the existing compositional framework [48, 49, 4]. Therefore, we believe that the research work presented in this thesis represents an original and significant contribution to this aim.

This research has resulted in the following key contributions, which are linked to the objectives listed in Section 1.4:

1. **Developed compositional attractor identification techniques**
   A key contribution of this work is developing compositional techniques for attractor analysis in an arbitrary structure of a composition. The approach presented is based on using the SCCs of a subnetwork's *interference state graph* to identify potential cyclic behaviour based on an important property called *interference alignment*, which is then used to construct the attractors of the composed model. We proved that this property captures instances in which paths in interference state graphs can be merged to create a path in a composed model. Since multiple entities in a BN can experience interference under composition, we provided generalised definitions for normal and interference edges added in an interference state graph. We formally proved the correctness of this approach by showing its *soundness* and *completeness*.

   It was crucial to initially developing attractor analysis techniques for a basic composition involving two BNs, as presented in Chapter 3. Moreover, we extended the attractor identification approach to a composition consisting of three BNs, and we partially extended it into the composition structure of a pairwise sequence of BNs presented in [49, 4]. The results and proofs became too complex based on the existing definition provided. Therefore, it became clear that developing a more general arbitrary compositional theory was needed. This work provided a unique insight into the approach and without this step we could not have extended it for a general composition, defined in the next contribution.

   The work above was conducted in Chapter 3 and Chapter 4, fulfilling *Objective 2.1* and *Objective 2.3*.

2. **Generalised the compositional definition to allow arbitrary compositions**
   Another key contribution of this thesis is the development of a new general definition of a composition to allow an arbitrary composition involving multiple BNs based on an underlying graph structure. The motivation to generalise the composition

formulation emerged from the difficulties which arose during the extending of attractor identification techniques. In fact, this work represents a significant step forward in the original compositional framework, because it made the composition approach more practical and simplified the representation of definitions and results. Consequently, it allowed us to develop a general compositional approach for attractor identification in an arbitrary composition.

We provided new definitions and notions to facilitate the theoretical results developed for the extended attractor identification techniques. One of them used the alternative approach of representing global states based on functions instead of tuples. Another important definition involved naming the composed entity in a BN that results from a composition. We updated the definition interference state graph [4] based on our new definitions and notations. In addition, we provided an updated version for an important theorem showing that an interference state graph captures all the possible behaviour that can result from a BN if it is used in a composition.

This work was conducted in Chapter 4, and fulfilled *Objective 2.2*.

3. **Developed tool support for the attractor identification approach**
   It was a crucial step to develop tool support to automate and illustrate the practical application of the compositional attractor identification techniques and results that we have developed. We formulated the algorithm $findAtt$ to identify attractors from a set of BNs based on the theoretical approach developed. The initial idea of the algorithm came from formulating and implementing the *findAttTwo* algorithm to identify attractors in two submodels. The $findAtt$ algorithm takes the submodels' SCCs data structure and returns the attractors in the resulting composed model. The algorithm repeats for each aligned state tuple, finding an interference aligned next state tuple until the repeated state tuple appears again, indicating that a set of interference aligned cyclic paths have been found.

   We carefully considered implementing its key functions efficiently. Therefore, to generate aligned state tuples efficiently, we formulated the algorithm $genAST$, which recursively iterates through key states to assign the values to composed entities if they are consistent (*i.e.* the values of the entities merged together have the same value). Interestingly, ordering BNs heuristically by calculating their *impact factor* leads to much better performance. Thus, we considered key states based on the resulting order. In addition, we produced an interesting idea to compute interference aligned state tuples, rather than considering each possible next state tuple. The idea is to count the labels for each composed entity $gc \in \Delta(\Sigma)$ in each state in the

tuple to generate the next state values for each composed entity and use them to determine the interference aligned next state tuples.

We implemented the formulated algorithms to identify attractors compositionally using Python. The tool takes the BNs' state graphs in a DOT format, which can be generated with existing tools such as *GinSim* [71, 125, 3], and the details of composed entities. The main implemented functions of the tool are generating interference state graphs, finding SCCs, labelling states, ordering BNs, generating aligned state tuples, finding interference aligned next state tuples, and generating composed model attractors.

We applied the tool to two biological case studies from the literature. The first one was a biologically relevant case study for cell differentiation in the bacteria *Caulobacter crescentus* [5, 6]. The second case study that we considered was the signalling pathway underpinning T cell surval in *T cell large granular lymphocyte leukemia (T-LGL)* [54]. These case studies provide important insight for anyone who would like to apply the developed techniques to biological networks and investigate the decomposition approach.

These achievements were realised in Chapter 3 and Chapter 5, and linked to *Objective 3.1* and *Objective 3.2*.

4. **Practical evaluation of the tool support**
   We developed a scalable test model to conduct a formal investigation into the developed tool's performance. We started our analysis using three submodels generating 12 entities in a composed model. We then scaled our model by constructing larger models, with up to 233 entities resulting from the composition. We conducted three experiments to test the performance, and to investigate how the SCCs and the size of resulted attractors influenced the tool. The results were promising, and revealed that our tool was affected by the SCCs' size and structure, the number of aligned state tuples and the composition structure. During the test, we discovered performance issues in some components, and we worked to improve their speed through several iterations. We compared the final tool's performance in terms of speed to three existing tools *BoolNet* [55], *BNS* [35] and *BoolSim* [56]. The comparison results show that the tool performs well against them, but further efficiency improvement is needed for large models.

   This was considered in Chapter 5, and fulfilled *Objective 3.3*.

6.3 Challenges

During the research reported in this thesis, many interesting challenges arose that needed to be addressed. Some important challenges are discussed below.

- **Moving from a basic composition of two BNs to an arbitrary version**
  Dealing with the complexity of moving from a simple composition, involving two BNs, to an arbitrary composition was challenging. The existing notations and definitions for the basic compositional framework became complicated when we extended the attractor identification approach to a set of three BNs and a set of multiple BNs in a composition. We were not able to represent arbitrary compositions using the original definitions that are in the existing theory. Therefore, the challenge was to develop a new definition for a composition that allows arbitrary compositions over multiple BNs. This new definition was needed to provide a basis for developing attractor identification techniques for a general composition. It was addressed by developing a new form of a composition, based on using a graph structure and a set of new definitions. This new formulation simplified the representation of definitions and results, and it was a key step to extend our developed approach for identifying attractors compositionally.

- **Extending attractor analysis results to an arbitrary composition**
  Dealing with the complexity of these arbitrary compositions and constructing proofs was challenging. The arbitrary composition has a large number of models and interference state graphs; thus, the proofs need to be carefully formulated. Changing the way of representing global states was associated with that. To explain that, instead of using them as tuples, we treated them in the new general formulation as functions and that was a crucial change which eased the process of producing results. Furthermore, the newly composed entities that were generated by merging multiple entities from different BNs were difficult to track. Thus, we named the entities used in the composition as $\Delta(g)$, and identify a set of entities that merged with any given entity used in a composition. It was also important to retrieve the BN to which a given entity belongs, and so we defined $\lambda(g)$ to be the index of the BN entity to which g belongs. The approach was based on analysing the submodels' interference state graphs, which have additional edges to represent interference. Hence, we need to distinguish between normal edges and interference edges for a given composed entity. This step is important in identifying the interference aligned cyclic paths to

be merged be merged. Therefore, we extended the *interference alignment* definition for multiple paths involved in a composition.

- **Developing the efficiency of tool support**
  Initially, the performance of the tool in terms of speed was poor, and showed that further work to improve its speed efficiency was required. In fact, the tool development went through several iterations to improve its efficiency, especially in generating aligned state tuples and computing interference aligned next state tuples. Initially, we considered the aligned tuples in all SCCs combinations to identify attractors. By observing this process, we found that we had repeated the exact aligning process for different states. To solve this issue, we avoided passing all possible combinations of SCCs, and we only focused on key states in the SCCs data structure to generate aligned state tuples based on the ordered BNs. Meanwhile, computing the interference aligned state tuple took a long time, as we considered all possible next state tuples. Then, we discovered the idea of computing the interference aligned next state tuple by examining the pre-processed labels to determine the next state values for composed entities. Choosing a data structure required careful consideration, and involved several iterations to produce data structures. During performance comparison, we wrote files in $BoolNet$ format, and we had to be careful when we composed functions to create a composed model for each test. Then, we converted the format of $BoolNet$ files to $GinSim$ files, and passed them to BNS and $BoolSim$ tools for analysis.

6.4 Future Work

This thesis has focused on extending and strengthening an existing compositional framework for BNs by developing attractor identification techniques and generalising the composition definition. With our developed tools and techniques, a number of interesting areas of future work are required to addressed in order to take our work forward.

- **Generalising the Boolean operators**
  While this thesis focuses on using conjunction to merge the behaviour of entities in a composition, all the results presented can straightforwardly be adapted to the use of disjunction. It turns out that the key property for the results is idempotency, and so any for Boolean operator which is idempotence, the result could be proved or

developed for it. Note that four binary Boolean operators are idempotent: transfer, complement, OR and AND. It would be interesting in future work to consider fully integrated conjunction and disjunction in a composition, so that both can be used.

- **Extending compatibility results to the new formulation of composition**
  There is a range of existing results around behaviour preservation in the basic composition framework [48, 49, 4], which appear to hold in a new compositional approach. It would be interesting to formally prove these results for the new general framework.

- **Decompositional framework based on our generalised composition**
  The current work supports the engineering of biological systems by constructing them from subparts. It would be beneficial to use it as a basis for developing techniques to decompose large models to aid analysis [207, 208, 191]. Constructing a decomposition of a BN raises some issues that need to be considered. One key issue is how to partition the set of entities and find suitable entities for this partition. Another issue is decomposing the Boolean functions of the entities that need to be split between partitions. It is clear that there may be more than one possible decomposition for a BN. Therefore, the third issue is how to measure the quality of the decomposition. Defining quality measures for the decomposition could help in identifying the optimal decompositions.

- **Developing compositional attractor analysis techniques for asynchronous BNs**
  Another interesting area to consider in the future is the development of an attractor identification approach for *asynchronous* BNs [67]. Prior to undertaking such a task, the development of a compositional theory for asynchronous BNs is required. In addition, an updated version of an interference state graph, which is a key element in the attractor identification algorithm is needed. It is generally thought to be more realistic to model biological networks using asynchronous BNs due to the distinct kinetics between nodes. However, asynchronous BNs are complex, and many algorithms of attractor identification focus on synchronous BNs. An interference state graph provides an initial insight into non-deterministic in an asynchronous semantics update. This topic is ongoing research work at Newcastle University and is being prepared for publication (therefore the detail of this work cannot be discussed in detail here).

- **Creating a library of parts**
  Biologists use biological parts to build genetic devices and systems, and they refer to this as *synthetic biology* [175–177]. There are libraries for biological parts to be used in constructing new models such as *GenoLIB* [209]. Designing and constructing synthetic a gene circuit helps in understanding the behaviour of biological systems. The ability to construct biological systems provides powerful tools to address various needs in the areas of health, energy and environment. For example, researchers have engineered immune cells to be safer and easier to produce the therapy to attack tumours [210], and engineered gene networks to provide treatment strategies for obesity [211] and diabetes [212]. One technique in synthetic biology which has gained attention is the Biobricks method for the physical composition of biological parts [213], which allows the composition of systems using standard biological parts. A standard biological part is a genetically encoded biological function that meets specific rules and guidance to support composition [214].

  It would be interesting to create a library of BNs models representing biological parts that can be used in the composition. These models are important in investigating further applications of our techniques and tools. This library would provide information on the parts before composing them. For example, by using a part's interference state graph, we can know the maximum behaviour in a composition [49, 4]. Interference state graphs do not show behaviour that does not occur in the composed model. Thus, if we want some behaviour to occur in the composed model, an interference state graph must show this behaviour. These are some ideas that can be investigated in the future.

- **Model synthesis**
  Model synthesis [215–217] is an important area that involves creating models that have certain behaviour. For example, several works have already considered synthesising models that have a certain type of attractors (e.g. [218–220]). In future work we would like to consider applying our techniques to this area. We can use submodels to predict behaviour that is going to occur. For example, we might need to build a model with particular attractors. By using our techniques, we can identify the final attractors from the behaviour of submodels.

- **Tool support**
  Our tool produces good results, but we would like to further increase its speed in the future. In addition, it would be interesting to have a GUI interface with which to construct BNs models using their functions or truth tables, or by importing

other known file formats such as *GinSim* [71, 125, 3], *BoolSim* [56] and *Systems Biology Markup Language (SBML)* [183]. We would like to add more features to the tool support, such as composing, analysing, simulating and visualising models. In addition, we could like to create a repository of BN models to facilitate applying the results in the tool.

- **Case studies**
  Further testing in large case studies is required in order to investigate the practical application of the tool. We would like to identify a proper case study based on a real biological system in order to perform further tests for the techniques that have been developed in this study.

Bibliography

[1] L. Kaderali and N. Radde, "Inferring gene regulatory networks from expression data," in *Computational intelligence in bioinformatics*, pp. 33–74, Springer, 2008.

[2] M. I. Davidich and S. Bornholdt, "Boolean network model predicts cell cycle sequence of fission yeast," *PloS one*, vol. 3, p. e1672, 2008 2008.

[3] A. Naldi, C. Hernandez, W. Abou-Jaoudé, P. T. Monteiro, C. Chaouiya, and D. Thieffry, "Logical modeling and analysis of cellular regulatory networks with ginsim 3.0," *Frontiers in physiology*, vol. 9, p. 646, 2018.

[4] H. Alkhudhayr, *Developing a Compositional Framework for the Construction and Analysis of Boolean Networks*. PhD thesis, School of Computing, Newcastle University, 2020.

[5] I. Sánchez-Osorio, C. A. Hernández-Martínez, and A. Martínez-Antonio, "Modeling asymmetric cell division in caulobacter crescentus using a boolean logic approach," in *Asymmetric Cell Division in Development, Differentiation and Cancer*, pp. 1–21, Springer, 2017.

[6] C. Quiñones-Valles, I. Sánchez-Osorio, and A. Martínez-Antonio, "Dynamical modeling of the cell cycle and cell fate emergence in caulobacter crescentus," *PloS one*, vol. 9, no. 11, p. e111116, 2014.

[7] A. Saadatpour, R.-S. Wang, A. Liao, X. Liu, T. P. Loughran, I. Albert, and R. Albert, "Dynamical and structural analysis of a T cell survival network identifies novel candidate therapeutic targets for large granular lymphocyte leukemia," *PLOS Computational Biology*, vol. 7, no. 11, p. e1002267, 2011.

[8] H. de Jong, "Modeling and simulation of genetic regulatory systems: a literature review," *Journal of Computational Biology*, vol. 9, pp. 67–103, 2002.

[9] R. Barbuti, R. Gori, P. Milazzo, and L. Nasti, "A survey of gene regulatory networks modelling methods: from differential equations, to boolean and qualitative bioinspired models," *Journal of Membrane Computing*, vol. 2, pp. 207–226, 2020.

[10] L. J. Steggles, R. Banks, O. Shaw, and A. Wipat, "Qualitatively modelling and analysing genetic regulatory networks: a Petri net approach," *Bioinformatics*, vol. 23, no. 3, pp. 336–343, 2007.

[11] T. Helikar, J. Konvalina, J. Heidel, and J. A. Rogers, "Emergent decision-making in biological signal transduction networks," *Proceedings of the National Academy of Sciences*, vol. 105, no. 6, pp. 1913–1918, 2008.

[12] A. Saadatpour, I. Albert, and R. Albert, "Attractor analysis of asynchronous boolean models of signal transduction networks," *Journal of theoretical biology*, vol. 266, no. 4, pp. 641–656, 2010.

[13] K. Voss, M. Heiner, and I. Koch, "Steady state analysis of metabolic pathways using Petri nets," *In silico biology*, vol. 3, no. 3, pp. 367–387, 2003.

[14] T. Akutsu, S. Miyano, and S. Kuhara, "Inferring qualitative relations in genetic networks and metabolic pathways," *Bioinformatics*, vol. 16, no. 8, pp. 727–734, 2000.

[15] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22(3), pp. 437–467, 1969.

[16] S. A. Kauffman, *The origins of order: Self organization and selection in evolution*. Oxford University Press, USA, 1993.

[17] R. Thomas, "Regulatory networks seen as asynchronous automata: a logical description," *Journal of theoretical biology*, vol. 153, no. 1, pp. 1–23, 1991.

[18] L. J. Steggles, "Abstracting asynchronous multi-valued networks: An initial investigation," *arXiv preprint arXiv:1108.3433*, 2011.

[19] M. A. Schaub, T. A. Henzinger, and J. Fisher, "Qualitative networks: a symbolic approach to analyze biological signaling networks," *BMC systems biology*, vol. 1, no. 1, pp. 1–21, 2007.

[20] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[21] A. Carl, "Petri. kommunikation mit automaten," *PhD, University of Bonn, West Germany*, 1962.

[22] M. Heiner, D. Gilbert, and R. Donaldson, "Petri nets for systems and synthetic biology," in *International school on formal methods for the design of computer, communication and software systems*, pp. 215–264, Springer, 2008.

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

[24] R. Thomas and M. Kaufman, "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*, vol. 11, no. 1, pp. 180–195, 2001.

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

[26] J. Saez-Rodriguez, L. Simeoni, J. A. Lindquist, R. Hemenway, U. Bommhardt, B. Arndt, U.-U. Haus, R. Weismantel, E. D. Gilles, S. Klamt, *et al.*, "A logical model provides insights into t cell receptor signaling," *PLoS Comput Biol*, vol. 3, no. 8, p. e163, 2007.

[27] P. Bloomingdale, J. Niu, D. E. Mager, *et al.*, "Boolean network modeling in systems pharmacology," *Journal of pharmacokinetics and pharmacodynamics*, vol. 45, no. 1, pp. 159–180, 2018.

[28] S. Huang and D. E. Ingber, "Shape-dependent control of cell growth, differentiation, and apoptosis: switching between attractors in cell regulatory networks," *Experimental cell research*, vol. 261, no. 1, pp. 91–103, 2000.

[29] R. Thomas, "Boolean formalization of genetic control circuits," *Journal of theoretical biology*, vol. 42, no. 3, pp. 563–585, 1973.

[30] S. Bornholdt and T. Rohlf, "Topological evolution of dynamical networks: Global criticality from local dynamics," *Physical Review Letters*, vol. 84, no. 26, p. 6114, 2000.

[31] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang, "The yeast cell-cycle network is robustly designed," *Proceedings of the National Academy of Sciences*, vol. 101, no. 14, pp. 4781–4786, 2004.

[32] L. Mendoza and I. Xenarios, "A method for the generation of standardized qualitative dynamical systems of regulatory networks," *Theoretical Biology and Medical Modelling*, vol. 3, no. 1, pp. 1–18, 2006.

[33] J. Aracena, E. Goles, A. Moreira, and L. Salinas, "On the robustness of update schedules in boolean networks," *Biosystems*, vol. 97, no. 1, pp. 1–8, 2009.

[34] F. Dellaert and R. D. Beer, "Toward an evolvable model of development for autonomous agent synthesis," in *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pp. 246–257, Citeseer, 1994.

[35] E. Dubrova and M. Teslenko, "A SAT-based algorithm for finding attractors in synchronous boolean networks," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 8, no. 5, pp. 1393–1399, 2011.

[36] A. Mizera, J. Pang, H. Qu, and Q. Yuan, "A new decomposition method for attractor detection in large synchronous boolean networks," in *International Symposium on Dependable Software Engineering: Theories, Tools, and Applications. SETTA 2017*, vol. LNCS 10606, pp. 232–249, Springer, 2017.

[37] S. Kochemazov and A. Semenov, "Using synchronous boolean networks to model several phenomena of collective behavior," *Plos one*, vol. 9, no. 12, p. e115156, 2014.

[38] S. Huang, G. Eichler, Y. Bar-Yam, and D. E. Ingber, "Cell fates as high-dimensional attractor states of a complex gene regulatory network," *Physical review letters*, vol. 94, no. 12, p. 128701, 2005.

[39] S. Huang, I. Ernberg, and S. Kauffman, "Cancer attractors: a systems view of tumors from a gene network dynamics and developmental perspective," in *Seminars in cell & developmental biology*, vol. 20, pp. 869–876, Elsevier, 2009.

[40] M. Choi, J. Shi, S. H. Jung, X. Chen, and K.-H. Cho, "Attractor landscape analysis reveals feedback loops in the p53 network that control the cellular response to DNA damage," *Science signaling*, vol. 5, no. 251, pp. ra83–ra83, 2012.

[41] S. Von der Heyde, C. Bender, F. Henjes, J. Sonntag, U. Korf, and T. Beissbarth, "Boolean ErbB network reconstructions and perturbation simulations reveal individual drug response in different breast cancer cell lines," *BMC systems biology*, vol. 8, no. 1, pp. 1–22, 2014.

[42] S. Huang, "Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery," *Journal of molecular medicine*, vol. 77, no. 6, pp. 469–480, 1999.

[43] S. Huang, "Genomics, complexity and drug discovery: insights from boolean network models of cellular regulation," *Pharmacogenomics*, vol. 2, no. 3, pp. 203–222, 2001.

[44] Y.-E. Sanchez-Corrales, E. R. Alvarez-Buylla, and L. Mendoza, "The arabidopsis thaliana flower organ specification gene regulatory network determines a robust differentiation process," *Journal of theoretical biology*, vol. 264, no. 3, pp. 971–983, 2010.

[45] R. Edwards and L. Glass, "Combinatorial explosion in model gene networks," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 10, no. 3, pp. 691–704, 2000.

[46] F. Ay, F. Xu, and T. Kahveci, "Scalable steady state analysis of Boolean biological regulatory networks," *PloS one*, vol. 4, no. 12, p. e7992, 2009.

[47] Y. Zhao, J. Kim, and M. Filippone, "Aggregation algorithm towards large-scale boolean network analysis," *IEEE Transactions on Automatic Control*, vol. 58, no. 8, pp. 1976–1985, 2013.

[48] H. Alkhudhayr and J. Steggles, "A formal framework for composing qualitative models of biological systems," in *Theory and Practice of Natural Computing, TPNC2017, LNCS 10687* (C. Martin-Vide, R. Neruda, and M. Vega-Rodriguez, eds.), pp. 25–36, Springer, 2017.

[49] H. Alkhudhayr and J. Steggles, "A compositional framework for boolean networks," *Biosystems*, vol. 186, p. 103960, 2019.

[50] N. D. Mendes, F. Lang, Y.-S. Le Cornec, R. Mateescu, G. Batt, and C. Chaouiya, "Composition and abstraction of logical regulatory modules: application to multicellular systems," *Bioinformatics*, vol. 29, no. 6, pp. 749–757, 2013.

[51] W. Guo, G. Yang, W. Wu, L. He, and M. Sun, "A parallel attractor finding algorithm based on boolean satisfiability for genetic regulatory networks," *PloS one*, vol. 9, no. 4, p. e94258, 2014.

[52] B. W. Miller and D. L. Ranum, *Problem Solving with Algorithms and Data Structures Using Python*. Franklin, Beedle and Associates Inc, 2 ed., 2011.

[53] E. Koutsofios and S. C. North, "Drawing graphs with dot," 1996.

[54] R. Zhang, M. V. Shah, J. Yang, S. B. Nyland, X. Liu, J. K. Yun, R. Albert, and T. P. Loughran, "Network model of survival signaling in large granular lymphocyte leukemia," *Proceedings of the National Academy of Sciences*, vol. 105, no. 42, pp. 16308–16313, 2008.

[55] C. Müssel, M. Hopfensitz, and H. A. Kestler, "BoolNet—an R package for generation, reconstruction and analysis of boolean networks," *Bioinformatics*, vol. 26, no. 10, pp. 1378–1380, 2010.

[56] A. Garg, A. Di Cara, I. Xenarios, L. Mendoza, and G. De Micheli, "Synchronous versus asynchronous modeling of gene regulatory networks," *Bioinformatics*, vol. 24, no. 17, pp. 1917–1925, 2008.

[57] M. Ilea, M. Turnea, M. Rotariu, *et al.*, "Ordinary differential equations with applications in molecular biology," *Rev Med Chir Soc Med Nat Iasi*, vol. 116, no. 1, pp. 347–352, 2012.

[58] M. L. Wynn, N. Consul, S. D. Merajver, and S. Schnell, "Logic-based models in systems biology: a predictive and parameter-free network analysis method," *Integrative biology*, vol. 4, no. 11, pp. 1323–1337, 2012.

[59] A. Saadatpour and R. Albert, "A comparative study of qualitative and quantitative dynamic models of biological regulatory networks," *EPJ Nonlinear Biomedical Physics*, vol. 4, no. 1, pp. 1–13, 2016.

[60] J. M. Bower and H. Bolouri, *Computational modeling of genetic and biochemical networks*. MIT press, 2001.

[61] R. L. Rudell and A. Sangiovanni-Vincentelli, "Multiple-valued minimization for PLA optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 5, pp. 727–750, 1987.

[62] R. Thomas and R. d'Ari, *Biological feedback*. CRC press, 1990.

[63] P. Baldan, N. Cocco, A. Marin, and M. Simeoni, "Petri nets for modelling metabolic pathways: a survey," *Natural Computing*, vol. 9, no. 4, pp. 955–989, 2010.

[64] W. Reisig, *Petri nets: an introduction*, vol. 4. Springer Science & Business Media, 2012.

[65] J. A. Bergstra and J. W. Klop, "Process algebra for synchronous communication," *Information and control*, vol. 60, no. 1-3, pp. 109–137, 1984.

[66] R. Thomas, A.-M. GATHOYE, and L. Lambert, "A complex control circuit: Regulation of immunity in temperate bacteriophages," *European Journal of Biochemistry*, vol. 71, no. 1, pp. 211–227, 1976.

[67] I. Harvey and T. Bossomaier, "Time out of joint: Attractors in asynchronous random boolean networks," in *Proceedings of the Fourth European Conference on Artificial Life*, pp. 67–75, MIT Press, Cambridge, 1997.

[68] S. Pandey, R.-S. Wang, L. Wilson, S. Li, Z. Zhao, T. Gookin, S. Assmann, and R. Albert, "Boolean modeling of transcriptome data reveals novel modes of heterotrimeric g-protein action," *Molecular Systems Biology*, vol. 6(1), pp. 2375–2387, 2010.

[69] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks," *Bioinformatics*, vol. 18, no. 2, pp. 261–274, 2002.

[70] D. Thieffry and R. Thomas, "Dynamical behaviour of biological regulatory networks—II. Immunity control in bacteriophage lambda," *Bulletin of Mathematical Biology*, vol. 57, no. 2, pp. 277–297, 1995.

[71] A. G. Gonzalez, A. Naldi, L. Sanchez, D. Thieffry, and C. Chaouiya, "GINsim: a software suite for the qualitative modelling, simulation and analysis of regulatory networks," *Biosystems*, vol. 84, no. 2, pp. 91–100, 2006.

[72] R. Banks and L. Steggles, "A high-level petri net framework for multi-valued genetic regulatory networks," *School of Computing Science Technical Report Series*, 2007.

[73] A. Mishchenko and R. Brayton, "Simplification of non-deterministic multi-valued networks," in *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pp. 557–562, 2002.

[74] R. Banks and L. J. Steggles, "An abstraction theory for qualitative models of biological systems," *Theoretical Computer Science*, vol. 431, pp. 207–218, 2012.

[75] A. Naldi, E. Remy, D. Thieffry, and C. Chaouiya, "A reduction of logical regulatory graphs preserving essential dynamical properties," in *International Conference on Computational Methods in Systems Biology*, pp. 266–280, Springer, 2009.

[76] A. Veliz-Cuba, "Reduction of Boolean network models," *Journal of theoretical biology*, vol. 289, pp. 167–172, 2011.

[77] C. A. Petri, "Kommunikation mit automaten," 1962.

[78] D. Gilbert and M. Heiner, "From Petri nets to differential equations–an integrative approach for biochemical network analysis," in *International Conference on Application and Theory of Petri Nets*, pp. 181–200, Springer, 2006.

[79] I. Koch and M. Heiner, "Petri Nets, in Junker BH and Schreiber, F.(eds.), Biological Network Analysis, chapter 7," 2008.

[80] J.-P. Comet, H. Klaudel, and S. Liauzu, "Modeling multi-valued genetic regulatory networks using high-level Petri nets," in *International Conference on Application and Theory of Petri Nets*, pp. 208–227, Springer, 2005.

[81] P. J. Goss and J. Peccoud, "Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets," *Proceedings of the National Academy of Sciences*, vol. 95, no. 12, pp. 6750–6755, 1998.

[82] O. Roig, J. Cortadella, and E. Pastor, "Verification of asynchronous circuits by BDD-based model checking of petri nets," in *International Conference on Application and Theory of Petri Nets*, pp. 374–391, Springer, 1995.

[83] B. Jasiul, M. Szpyrka, and J. Śliwa, "Detection and modeling of cyber attacks with petri nets," *Entropy*, vol. 16, no. 12, pp. 6602–6623, 2014.

[84] T. M. Chen, J. C. Sanchez-Aarnoutse, and J. Buford, "Petri net modeling of cyber-physical attacks on smart grid," *IEEE Transactions on smart grid*, vol. 2, no. 4, pp. 741–749, 2011.

[85] J. Billington and M. Diaz, *Application of Petri nets to communication networks: Advances in Petri nets*. No. 1605, Springer Science & Business Media, 1999.

[86] K. Jensen, *Coloured Petri nets: basic concepts, analysis methods and practical use*, vol. 1. Springer Science & Business Media, 1997.

[87] M. K. Molloy, "On the integration of delay and throughput measures in distributed processing models.," 1982.

[88] P. Merlin and D. Farber, "Recoverability of communication protocols-implications of a theoretical study," *IEEE transactions on Communications*, vol. 24, no. 9, pp. 1036–1043, 1976.

[89] C. Talcott and D. L. Dill, "The pathway logic assistant," in *Third International Workshop on Computational Methods in Systems Biology*, vol. 3, pp. 228–239, Citeseer, 2005.

[90] M. Heiner, M. Herajy, F. Liu, C. Rohr, and M. Schwarick, "Snoopy–a unifying Petri net tool," in *International Conference on Application and Theory of Petri Nets and Concurrency*, pp. 398–407, Springer, 2012.

[91] S. Baarir, M. Beccuti, D. Cerotti, M. De Pierro, S. Donatelli, and G. Franceschinis, "The GreatSPN tool: recent enhancements," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 4, pp. 4–9, 2009.

[92] L. Napione, D. Manini, F. Cordero, A. Horváth, A. Picco, M. De Pierro, S. Pavan, M. Sereno, A. Veglio, F. Bussolino, *et al.*, "On the use of stochastic Petri nets in the analysis of signal transduction pathways for angiogenesis process," in *International Conference on Computational Methods in Systems Biology*, pp. 281–295, Springer, 2009.

[93] A. Regev, W. Silverman, and E. Shapiro, "Representation and simulation of biochemical processes using the π-calculus process algebra," in *Biocomputing 2001*, pp. 459–470, World Scientific, 2000.

[94] M. Calder, S. Gilmore, and J. Hillston, "Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA," in *Transactions on computational systems biology VII*, pp. 1–23, Springer, 2006.

[95] O. Tymchyshyn and M. Kwiatkowska, "Combining intra-and inter-cellular dynamics to investigate intestinal homeostasis," in *International Workshop on Formal Methods in Systems Biology*, pp. 63–76, Springer, 2008.

[96] M. Bernardo, P. Degano, and G. Zavattaro, *Formal Methods for Computational Systems Biology: 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2008 Bertinoro, Italy, June 2-7, 2008*, vol. 5016. Springer, 2008.

[97] R. Milner *et al.*, "A calculus of communicating systems," 1980.

[98] C. A. R. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, no. 8, pp. 666–677, 1978.

[99] C.-C. Jou and S. A. Smolka, "Equivalences, congruences, and complete axiomatizations for probabilistic processes," in *International Conference on Concurrency Theory*, pp. 367–383, Springer, 1990.

[100] R. Pucella, "Communicating and mobile systems: The π-calculus," 2000.

[101] P. Degano, D. Prandi, C. Priami, and P. Quaglia, "Beta-binders for biological quantitative experiments," *Electronic Notes in Theoretical Computer Science*, vol. 164, no. 3, pp. 101–117, 2006.

[102] J. Hillston, "A compositional approach to performance modelling," 1996.

[103] H. Garavel, F. Lang, and R. Mateescu, "An overview of CADP 2001," 2001.

[104] F. Moller and P. Stevens, "Edinburgh concurrency workbench user manual (version 7.1)," 1999.

[105] P. D'haeseleer, S. Liang, and R. Somogyi, "Genetic network inference: from co-expression clustering to reverse engineering," *Bioinformatics*, vol. 16, no. 8, pp. 707–726, 2000.

[106] P. E. Dunne, *The complexity of Boolean networks*. Academic Press Professional, Inc., 1988.

[107] R. J. Tocci, *Digital systems: principles and applications*. Pearson Education India, 1991.

[108] B. Drossel, T. Mihaljev, and F. Greil, "Number and length of attractors in a critical Kauffman model with connectivity one," *Physical Review Letters*, vol. 94, no. 8, p. 088701, 2005.

[109] I. Shmulevich and S. A. Kauffman, "Activities and sensitivities in Boolean network models," *Physical review letters*, vol. 93, no. 4, p. 048701, 2004.

[110] J. E. Socolar and S. A. Kauffman, "Scaling in ordered and critical random Boolean networks," *Physical review letters*, vol. 90, no. 6, p. 068702, 2003.

[111] D. Wright, T. Stocker, and L. Mysak, "A note on quaternary climate modelling using Boolean delay equations," *Climate dynamics*, vol. 4, no. 4, pp. 263–267, 1990.

[112] I. Zaliapin, V. Keilis-Borok, and M. Ghil, "A boolean delay equation model of colliding cascades. Part I: Multiple seismic regimes," *Journal of Statistical Physics*, vol. 111, no. 3, pp. 815–837, 2003.

[113] G. Easton, R. J. Brooks, K. Georgieva, and I. Wilkinson, "Understanding the dynamics of industrial networks using Kauffman boolean networks," *Advances in Complex Systems*, vol. 11, no. 01, pp. 139–164, 2008.

[114] J.-W. Gu, W.-K. Ching, T.-K. Siu, and H. Zheng, "On modeling credit defaults: A probabilistic Boolean network approach," *Risk and decision analysis*, vol. 4, no. 2, pp. 119–129, 2013.

[115] B. Coluzzi, M. Ghil, S. Hallegatte, and G. Weisbuch, "Boolean delay equations on networks in economics and the geosciences," *International Journal of Bifurcation and Chaos*, vol. 21, no. 12, pp. 3511–3548, 2011.

[116] A. Roli, M. Manfroni, C. Pinciroli, and M. Birattari, "On the design of Boolean network robots," in *European Conference on the Applications of Evolutionary Computation*, pp. 43–52, Springer, 2011.

[117] A. Roli, M. Villani, R. Serra, S. Benedettini, C. Pinciroli, and M. Birattari, "Dynamical properties of artificially evolved Boolean network robots," in *Congress of the Italian Association for Artificial Intelligence*, pp. 45–57, Springer, 2015.

[118] R. A. Banks, *Qualitatively modelling genetic regulatory networks: Petri net techniques and tools*. PhD thesis, Newcastle University, 2009.

[119] J. L. Apostel, "Classification of random boolean networks," *Artificial Life 8*, vol. 8, p. 1, 2003.

[120] A. Poret, C. M. Sousa, and J.-P. Boissel, "Enhancing boolean networks with continuous logical operators and edge tuning," *arXiv preprint arXiv:1407.1135*, 2014.

[121] M. Hopfensitz, C. Müssel, M. Maucher, and H. A. Kestler, "Attractors in boolean networks: a tutorial," *Computational Statistics*, vol. 28, no. 1, pp. 19–36, 2013.

[122] Q. Yuan, H. Qu, J. Pang, and A. Mizera, "Improving BDD-based attractor detection for synchronous boolean networks," *Science China Information Sciences*, vol. 59, no. 8, pp. 1–16, 2016.

[123] Q. Yuan, A. Mizera, J. Pang, and H. Qu, "A new decomposition-based method for detecting attractors in synchronous boolean networks," *Science of Computer Programming*, vol. 180, pp. 18–35, 2019.

[124] L. E. Chai, S. K. Loh, S. T. Low, M. S. Mohamad, S. Deris, and Z. Zakaria, "A review on the computational approaches for gene regulatory network construction," *Computers in biology and medicine*, vol. 48, pp. 55–65, 2014.

[125] A. Naldi, D. Berenguier, A. Fauré, F. Lopez, D. Thieffry, and C. Chaouiya, "Logical modelling of regulatory networks with GINsim 2.3," *Biosystems*, vol. 97, no. 2, pp. 134–139, 2009.

[126] A. Naldi, D. Thieffry, and C. Chaouiya, "Decision diagrams for the representation and analysis of logical models of genetic networks," in *International Conference on Computational Methods in Systems Biology*, pp. 233–247, Springer, 2007.

[127] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz—open source graph drawing tools," in *International Symposium on Graph Drawing*, pp. 483–484, Springer, 2001.

[128] A. J. Enright and C. A. Ouzounis, "Biolayout—an automatic graph layout algorithm for similarity visualization," *Bioinformatics*, vol. 17, no. 9, pp. 853–854, 2001.

[129] M. E. Smoot, K. Ono, J. Ruscheinski, P.-L. Wang, and T. Ideker, "Cytoscape 2.8: new features for data integration and network visualization," *Bioinformatics*, vol. 27, no. 3, pp. 431–432, 2011.

[130] D. Bérenguier, C. Chaouiya, P. T. Monteiro, A. Naldi, E. Remy, D. Thieffry, and L. Tichit, "Dynamical modeling and analysis of large cellular regulatory networks," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 23, no. 2, p. 025114, 2013.

[131] W. J. Longabaugh, E. H. Davidson, and H. Bolouri, "Computational representation of developmental genetic regulatory networks," *Developmental biology*, vol. 283, no. 1, pp. 1–16, 2005.

[132] V. Batagelj and A. Mrvar, "Pajek-program for large network analysis," *Connections*, vol. 21, no. 2, pp. 47–57, 1998.

[133] M. Bock, T. Scharp, C. Talnikar, and E. Klipp, "BooleSim: an interactive boolean network simulator," *Bioinformatics*, vol. 30, no. 1, pp. 131–132, 2014.

[134] F. Krause, M. Schulz, B. Ripkens, M. Flöttmann, M. Krantz, E. Klipp, and T. Handorf, "Biographer: web-based editing and rendering of SBGN compliant biochemical networks," *Bioinformatics*, vol. 29, no. 11, pp. 1467–1468, 2013.

[135] J. Schwab, A. Burkovski, L. Siegle, C. Müssel, and H. A. Kestler, "ViSiBooL—visualization and simulation of boolean networks with temporal constraints," *Bioinformatics*, vol. 33, no. 4, pp. 601–604, 2016.

[136] T. M. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[137] T. Kamada, S. Kawai, *et al.*, "An algorithm for drawing general undirected graphs," *Information processing letters*, vol. 31, no. 1, pp. 7–15, 1989.

[138] K. Sugiyama, S. Tagawa, and M. Toda, "Methods for visual understanding of hierarchical system structures," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 2, pp. 109–125, 1981.

[139] W. Peckham, "A support tool for boolean network," 2017.

[140] H. De Jong, J. Geiselmann, C. Hernandez, and M. Page, "Genetic Network Analyzer: qualitative simulation of genetic regulatory networks," *Bioinformatics*, vol. 19, no. 3, pp. 336–344, 2003.

[141] A. Di Cara, A. Garg, G. De Micheli, I. Xenarios, and L. Mendoza, "Dynamic simulation of regulatory networks using squad," *BMC bioinformatics*, vol. 8, no. 1, pp. 1–10, 2007.

[142] S. Klamt, J. Saez-Rodriguez, and E. D. Gilles, "Structural and functional analysis of cellular networks with cellnetanalyzer," *BMC systems biology*, vol. 1, no. 1, pp. 1–13, 2007.

[143] S. Karl and T. Dandekar, "Jimena: efficient computing and system state identification for genetic regulatory networks," *BMC bioinformatics*, vol. 14, no. 1, pp. 1–11, 2013.

[144] S. Li, S. M. Assmann, and R. Albert, "Predicting essential components of signal transduction networks: a dynamic model of guard cell abscisic acid signaling," *PLoS biology*, vol. 4, no. 10, p. e312, 2006.

[145] J. Thakar, M. Pilione, G. Kirimanjeswara, E. T. Harvill, and R. Albert, "Modeling systems-level regulation of host immune responses," *PLoS computational biology*, vol. 3, no. 6, p. e109, 2007.

[146] G. Karlebach and R. Shamir, "Modelling and analysis of gene regulatory networks," *Nature reviews Molecular cell biology*, vol. 9, no. 10, pp. 770–780, 2008.

[147] W.-P. Lee and W.-S. Tzou, "Computational methods for discovering gene networks from expression data," *Briefings in bioinformatics*, vol. 10, no. 4, pp. 408–423, 2009.

[148] J. Saez-Rodriguez, L. G. Alexopoulos, M. Zhang, M. K. Morris, D. A. Lauffenburger, and P. K. Sorger, "Comparing signaling networks between normal and transformed hepatocytes using discrete logical models," *Cancer research*, vol. 71, no. 16, pp. 5400–5411, 2011.

[149] J. Saez-Rodriguez, L. G. Alexopoulos, J. Epperlein, R. Samaga, D. A. Lauffenburger, S. Klamt, and P. K. Sorger, "Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction," *Molecular systems biology*, vol. 5, no. 1, p. 331, 2009.

[150] J. Collado-Vides and R. Hofestädt, *Gene regulation and metabolism: postgenomic computational approaches*. MIT Press, 2004.

[151] X. Zhang, X.-M. Zhao, K. He, L. Lu, Y. Cao, J. Liu, J.-K. Hao, Z.-P. Liu, and L. Chen, "Inferring gene regulatory networks from gene expression data by path consistency algorithm based on conditional mutual information," *Bioinformatics*, vol. 28, no. 1, pp. 98–104, 2012.

[152] B. Alberts, A. Johnson, J. Lewis, D. Morgan, M. Raff, K. Roberts, P. Walter, J. Wilson, and T. Hunt, *Molecular biology of the cell*. WW Norton & Company, 2017.

[153] B. Ristevski, "A survey of models for inference of gene regulatory networks," *Nonlinear Analysis: Modelling and Control*, vol. 18, no. 4, pp. 444–465, 2013.

[154] S. A. Kauffman, "Antichaos and adaptation," *Scientific American*, vol. 265, no. 2, pp. 78–85, 1991.

[155] R. Somogyi and C. A. Sniegoski, "Modeling the complexity of genetic networks: understanding multigenic and pleiotropic regulation," *complexity*, vol. 1, no. 6, pp. 45–63, 1996.

[156] Z. Szallasi and S. Liang, "Modeling the normal and neoplastic cell cycle with 'realistic Boolean genetic networks': Their application for understanding carcinogenesis and assessing therapeutic strategies," in *Pacific Symposium on Biocomputing*, vol. 3, pp. 66–76, Citeseer, 1998.

[157] G. Weisbuch, "Networks of automata and biological organization," *Journal of theoretical Biology*, vol. 121, no. 3, pp. 255–267, 1986.

[158] D. Irons, "Logical analysis of the budding yeast cell cycle," *Journal of theoretical biology*, vol. 257, no. 4, pp. 543–559, 2009.

[159] S. Kauffman, "Homeostasis and differentiation in random genetic control networks," *Nature*, vol. 224, no. 5215, pp. 177–178, 1969.

[160] G. d. Anda-Jáuregui, J. Espinal-Enríquez, S. Sandoval-Motta, and E. Hernández-Lemus, "A boolean network approach to estrogen transcriptional regulation," *Complexity*, vol. 2019, 2019.

[161] M. E. Martinez-Sanchez, M. Hiriart, and E. R. Alvarez-Buylla, "The CD4+ T cell regulatory network mediates inflammatory responses during acute hyperinsulinemia: a simulation study," *BMC systems biology*, vol. 11, no. 1, pp. 1–12, 2017.

[162] Y.-K. Kwon and K.-H. Cho, "Boolean dynamics of biological networks with multiple coupled feedback loops," *Biophysical Journal*, vol. 92, no. 8, pp. 2975–2981, 2007.

[163] I. Shmulevich, E. R. Dougherty, and W. Zhang, "From Boolean to probabilistic Boolean networks as models of genetic regulatory networks," *Proceedings of the IEEE*, vol. 90, no. 11, pp. 1778–1792, 2002.

[164] J. Barnat, N. Beneš, L. Brim, M. Demko, M. Hajnal, S. Pastva, and D. Šafránek, "Detecting attractors in biological models with uncertain parameters," in *International Conference on Computational Methods in Systems Biology*, pp. 40–56, Springer, 2017.

[165] A. Garg, I. Xenarios, L. Mendoza, and G. DeMicheli, "An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments," in *Annual International Conference on Research in Computational Molecular Biology*, pp. 62–76, Springer, 2007.

[166] D. Zheng, G. Yang, X. Li, Z. Wang, F. Liu, and L. He, "An efficient algorithm for computing attractors of synchronous and asynchronous boolean networks," *PloS one*, vol. 8, no. 4, p. e60593, 2013.

[167] E. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded model checking using satisfiability solving," *Formal methods in system design*, vol. 19, no. 1, pp. 7–34, 2001.

[168] A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu, "Symbolic model checking using SAT procedures instead of BDDs," in *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, pp. 317–320, 1999.

[169] E. Dubrova and M. Teslenko, "Compositional properties of random boolean networks," *Physical Review E*, vol. 71, no. 5, p. 056116, 2005.

[170] K. C. Chen, L. Calzone, A. Csikasz-Nagy, F. R. Cross, B. Novak, and J. J. Tyson, "Integrative analysis of cell cycle control in budding yeast," *Molecular biology of the cell*, vol. 15, no. 8, pp. 3841–3862, 2004.

[171] R. Randhawa, C. A. Shaffer, and J. J. Tyson, "Model aggregation: a building-block approach to creating large macromolecular regulatory networks," *Bioinformatics*, vol. 25, no. 24, pp. 3289–3295, 2009.

[172] G. Buzi, U. Topcu, and J. C. Doyle, "Compositional analysis of autocatalytic networks in biology," in *Proceedings of the 2010 American Control Conference*, pp. 5929–5935, IEEE, 2010.

[173] E. Klipp, B. Nordlander, R. Krüger, P. Gennemark, and S. Hohmann, "Integrative model of the response of yeast to osmotic shock," *Nature biotechnology*, vol. 23, no. 8, pp. 975–982, 2005.

[174] R. Randhawa, C. Shaffer, and J. Tyson, "Model composition for macromolecular regulatory networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 7, no. 2, pp. 278–287, 2008.

[175] M. Heinemann and S. Panke, "Synthetic biology—putting engineering into biology," *Bioinformatics*, vol. 22, no. 22, pp. 2790–2799, 2006.

[176] L. Serrano, "Synthetic biology: promises and challenges," 2007.

[177] K. Channon, E. H. Bromley, and D. N. Woolfson, "Synthetic biology through biomolecular design and engineering," *Current opinion in structural biology*, vol. 18, no. 4, pp. 491–498, 2008.

[178] L. Hartwell, J. Hopfield, S. Leibler, and A. Murray, "Nature 402 suppl," *C47–C52*, 1999.

[179] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, "Hierarchical organization of modularity in metabolic networks," *science*, vol. 297, no. 5586, pp. 1551–1555, 2002.

[180] A. Fauré, A. Naldi, F. Lopez, C. Chaouiya, A. Ciliberto, and D. Thieffry, "Modular logical modelling of the budding yeast cell cycle," *Molecular BioSystems*, vol. 5, no. 12, pp. 1787–1796, 2009.

[181] L. Sánchez, C. Chaouiya, and D. Thieffry, "Segmenting the fly embryo: logical analysis of the role of the segment polarity cross-regulatory module," *International journal of developmental biology*, vol. 52, no. 8, pp. 1059–1075, 2002.

[182] R. Randhawa, C. A. Shaffer, and J. J. Tyson, "Fusing and composing macromolecular regulatory network models," in *Proceedings of the 2007 spring simulation multiconference-Volume 2*, pp. 337–344, Citeseer, 2007.

[183] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, *et al.*, "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, no. 4, pp. 524–531, 2003.

[184] C. Chaouiya, H. Klaudel, and F. Pommereau, "A modular, qualitative modeling of regulatory networks using Petri nets," in *Modeling in Systems Biology*, pp. 253–279, Springer, 2011.

[185] S. d. Putter and A. Wijs, "Compositional model checking is lively," in *International Conference on Formal Aspects of Component Software*, pp. 117–136, Springer, 2017.

[186] E. M. Clarke, D. E. Long, and K. L. McMillan, "Compositional model checking," 1989.

[187] J. J. Hooman and W.-P. de Roever, "An introduction to compositional methods for concurrency and their application to real-time," *Sadhana*, vol. 17, no. 1, pp. 29–73, 1992.

[188] H. Zheng, Z. Zhang, C. J. Myers, E. Rodriguez, and Y. Zhang, "Compositional model checking of concurrent systems," *IEEE Transactions on Computers*, vol. 64, no. 6, pp. 1607–1621, 2014.

[189] A. Varga and R. K. Moore, "Simultaneous recognition of concurrent speech signals using hidden markov model decomposition," in *Second European Conference on Speech Communication and Technology*, 1991.

[190] J. Saez-Rodriguez, S. Gayer, M. Ginkel, and E. D. Gilles, "Automatic decomposition of kinetic models of signaling networks minimizing the retroactivity among modules," *Bioinformatics*, vol. 24, no. 16, pp. i213–i219, 2008.

[191] J. Anderson, Y.-C. Chang, and A. Papachristodoulou, "Model decomposition and reduction tools for large-scale networks in systems biology," *Automatica*, vol. 47, no. 6, pp. 1165–1174, 2011.

[192] P. Holme, M. Huss, and H. Jeong, "Subnetwork hierarchies of biochemical pathways," *Bioinformatics*, vol. 19, no. 4, pp. 532–538, 2003.

[193] J. Saez-Rodriguez, A. Kremling, H. Conzelmann, K. Bettenbrock, and E. D. Gilles, "Modular analysis of signal transduction networks," *IEEE control systems*, vol. 24, no. 4, pp. 35–52, 2004.

[194] D. Del Vecchio and E. D. Sontag, "Engineering principles in bio-molecular systems: from retroactivity to modularity," in *Control Conference (ECC), 2009 European*, pp. 658–664, IEEE, 2009.

[195] D. Del Vecchio, A. J. Ninfa, and E. D. Sontag, "Modular cell biology: retroactivity and insulation," *Molecular systems biology*, vol. 4, no. 1, p. 161, 2008.

[196] K. Perrot, P. Perrotin, and S. Sené, "A framework for (de) composing with Boolean automata networks," in *International Conference on Machines, Computations, and Universality*, pp. 121–136, Springer, 2018.

[197] C. Hong, J. Hwang, K.-H. Cho, and I. Shin, "An efficient steady-state analysis method for large Boolean networks with high maximum node connectivity," *PLOS ONE*, vol. 10, no. 12, p. e0145734, 2015.

[198] K. Perrot, P. Perrotin, and S. Sené, "Optimising attractor computation in boolean automata networks," in *International Conference on Language and Automata Theory and Applications*, pp. 68–80, Springer, 2021.

[199] M. T. Laub, L. Shapiro, and H. H. McAdams, "Systems biology of caulobacter," *Annu. Rev. Genet.*, vol. 41, pp. 429–441, 2007.

[200] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[201] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using NetworkX," tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[202] E. Nuutila and E. Soisalon-Soininen, "On finding the strongly connected components in a directed graph," *Information processing letters*, vol. 49, no. 1, pp. 9–14, 1994.

[203] M. Sharir, "A strong-connectivity algorithm and its applications in data flow analysis," *Computers & Mathematics with Applications*, vol. 7, no. 1, pp. 67–72, 1981.

[204] N. Een, "MiniSat: A SAT solver with conflict-clause minimization," in *Proc. SAT-05: 8th Int. Conf. on Theory and Applications of Satisfiability Testing*, pp. 502–518, 2005.

[205] N. Scott, "A new approach to the design of switching circuits," *Proceedings of the IEEE*, vol. 51, no. 2, pp. 413–413, 1963.

[206] C. Yang, V. Singhal, and M. Ciesielski, "BDD decomposition for efficient logic synthesis," in *Computer Design, 1999.(ICCD'99) International Conference on*, pp. 626–631, IEEE, 1999.

[207] D. J. Clancy and B. Kuipers, "Model decomposition and simulation: A component based qualitative simulation algorithm," in *AAAI/IAAI*, pp. 118–124, 1997.

[208] A. Varga and R. Moore, "Hidden Markov model decomposition of speech and noise," in *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pp. 845–848, IEEE, 1990.

[209] N. R. Adames, M. L. Wilson, G. Fang, M. W. Lux, B. S. Glick, and J. Peccoud, "GenoLIB: a database of biological parts derived from a library of common plasmid features," *Nucleic acids research*, vol. 43, no. 10, pp. 4823–4832, 2015.

[210] D. Chakravarti and W. W. Wong, "Synthetic biology in cell-based cancer immunotherapy," *Trends in biotechnology*, vol. 33, no. 8, pp. 449–461, 2015.

[211] K. Rössger, G. Charpin-El-Hamri, and M. Fussenegger, "A closed-loop synthetic gene circuit for the treatment of diet-induced obesity in mice," *Nature communications*, vol. 4, no. 1, pp. 1–9, 2013.

[212] H. Ye, M. Daoud-El Baba, R.-W. Peng, and M. Fussenegger, "A synthetic optogenetic transcription device enhances blood-glucose homeostasis in mice," *Science*, vol. 332, no. 6037, pp. 1565–1568, 2011.

[213] T. Knight, "Idempotent vector design for standard assembly of biobricks," 2003.

[214] B. Canton, A. Labno, and D. Endy, "Refinement and standardization of synthetic biological parts and devices," *Nature biotechnology*, vol. 26, no. 7, pp. 787–793, 2008.

[215] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, *Logic Synthesis for Asynchronous Controllers and Interfaces: With 146 Figures*, vol. 8. Springer Science & Business Media, 2002.

[216] S. Uchitel, J. Kramer, and J. Magee, "Synthesis of behavioral models from scenarios," *IEEE Transactions on Software Engineering*, vol. 29, no. 2, pp. 99–115, 2003.

[217] A. S. Koksal, Y. Pu, S. Srivastava, R. Bodik, J. Fisher, and N. Piterman, "Synthesis of biological models from mutation experiments," in *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 469–482, 2013.

[218] R. Pal, I. Ivanov, A. Datta, M. L. Bittner, and E. R. Dougherty, "Generating Boolean networks with a prescribed attractor structure," *Bioinformatics*, vol. 21, no. 21, pp. 4021–4025, 2005.

[219] X. Zhou, X. Wang, R. Pal, I. Ivanov, M. Bittner, and E. R. Dougherty, "A bayesian connectivity-based approach to constructing probabilistic gene regulatory networks," *Bioinformatics*, vol. 20, no. 17, pp. 2918–2927, 2004.

[220] K. Kobayashi and K. Hiraishi, "Design of Boolean networks based on prescribed singleton attractors," in *2014 European Control Conference (ECC)*, pp. 1504–1509, IEEE, 2014.

Appendix A

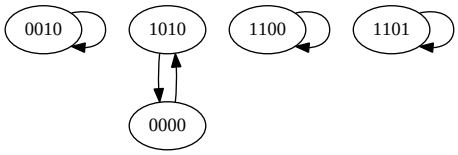
Test Models

In this section, we illustrate the SCCs of the submodels' interference state graphs used to identify attractors for the performance tests. After that, We summarise the compositional test models by illustrating the underlying BNs and the composed entities.

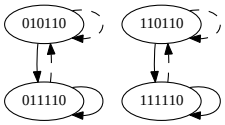
A.1 Submodels' SCCs

Figure A.1 presents the SCCs of the submodels' interference state graphs identified for $\mathcal{BN}^1_{Test}$, $\mathcal{BN}^2_{Test}$, $\mathcal{BN}^3_{Test}$, $\mathcal{BN}^4_{Test}$, $\mathcal{BN}^5_{Test}$ and $\mathcal{BN}^6_{Test}$ based on the entities used for a composition. The SCCs of the submodel's interference state graph identified for $\mathcal{BN}^7_{Test}$ are depicted in Figure A.4.

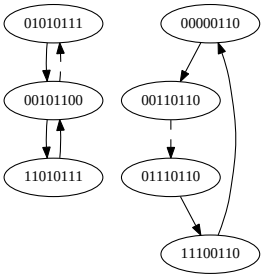
$\mathcal{BN}^1_{Test}$ composed on g^1_1



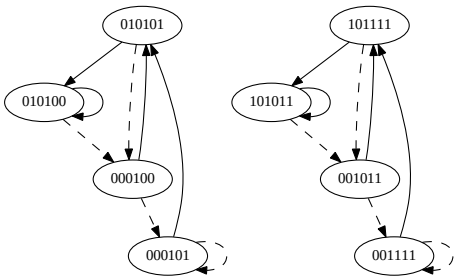
$\mathcal{BN}^2_{Test}$ composed on g^2_1



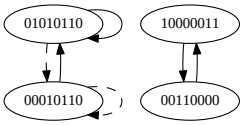
$\mathcal{BN}^3_{Test}$ composed on g^3_1 and g^3_3



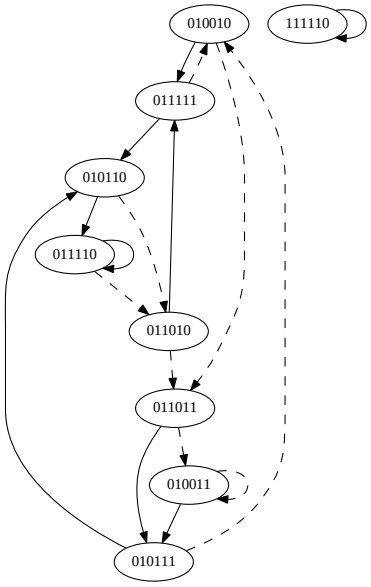
$\mathcal{BN}^6_{Test}$ composed on g^6_1



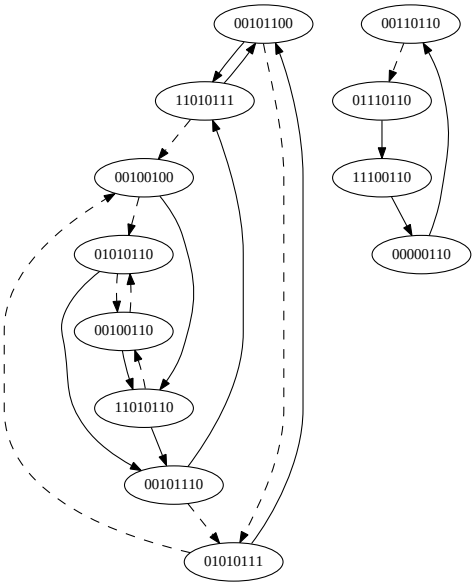
$\mathcal{BN}^4_{Test}$ composed on g^4_1 and g^4_2



$\mathcal{BN}^5_{Test}$ composed on g^5_1, g^5_2 and g^5_3



$\mathcal{BN}^3_{Test}$ composed on g^3_1 and g^3_4



$\mathcal{BN}^6_{Test}$ composed on g^6_1 and g^6_5

Figure A.1 The SCCs identified for $\mathcal{BN}^1_{Test}$, $\mathcal{BN}^2_{Test}$, $\mathcal{BN}^3_{Test}$, $\mathcal{BN}^4_{Test}$, $\mathcal{BN}^5_{Test}$ and $\mathcal{BN}^6_{Test}$.

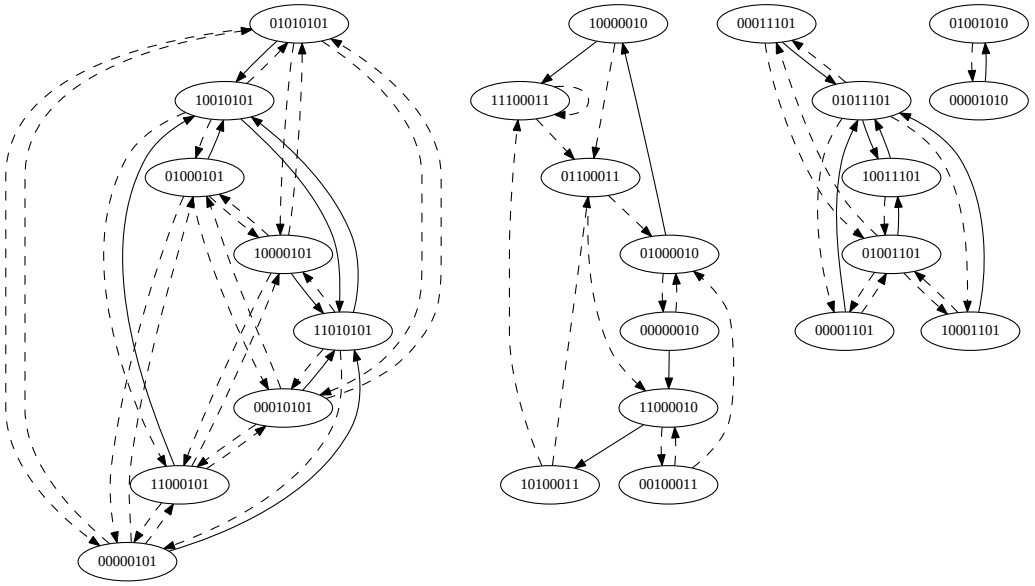


Figure A.2 The SCCs identified for $\mathcal{BN}^7_{Test}$, where the composed entities are g^7_1 and g^7_3.

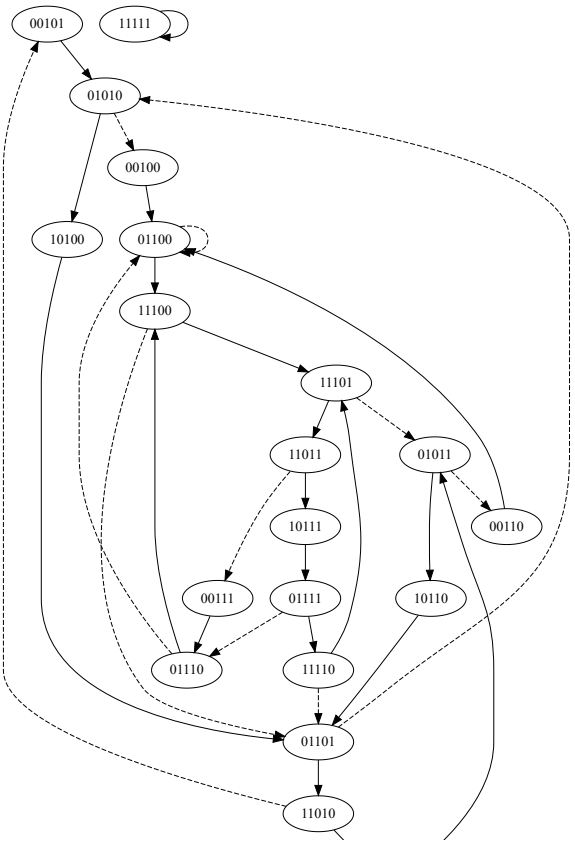


Figure A.3 The SCCs identified for $\mathcal{BN}^8_{Test}$, where the composed entities are g^8_1.

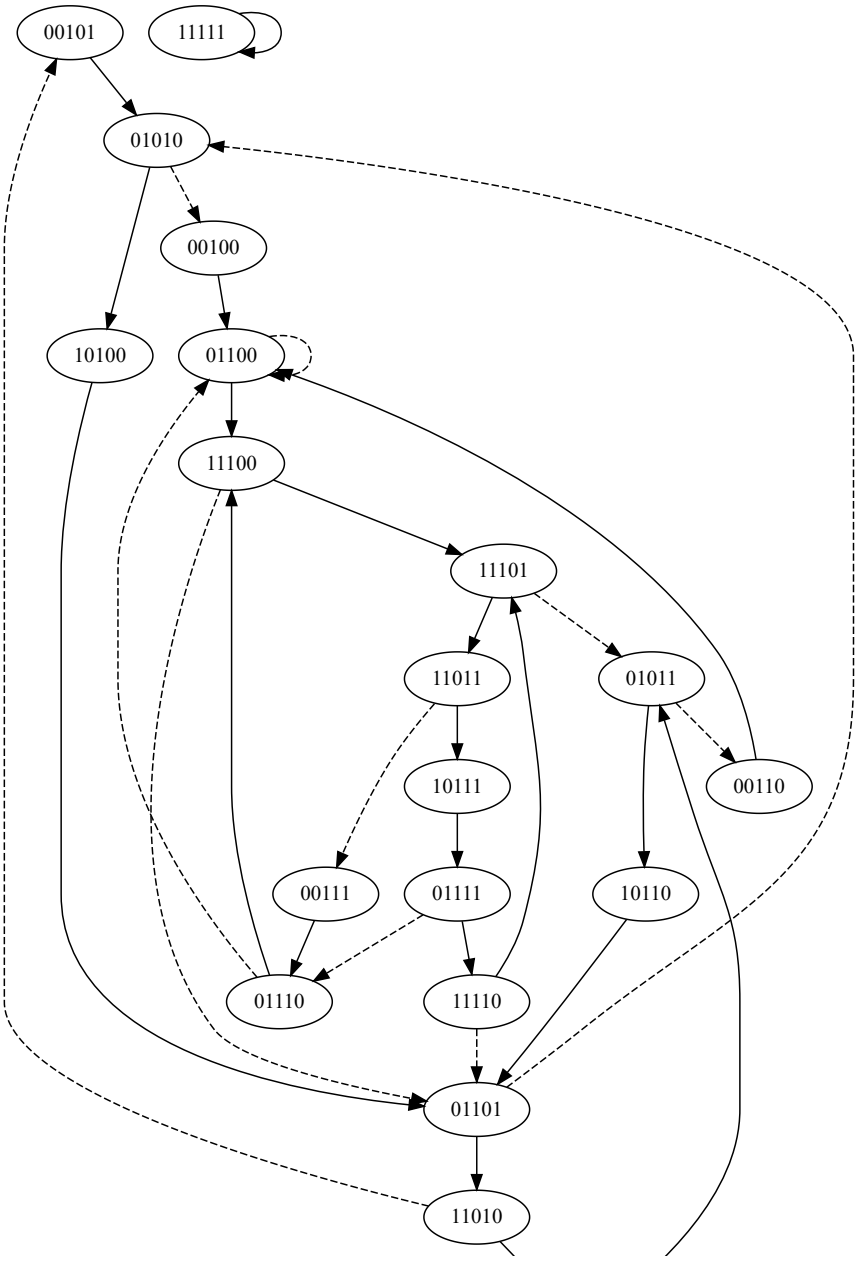


Figure A.4 The SCCs identified for $\mathcal{BN}^9_{Test}$, where the composed entities are g^9_1.

A.2 Compositions

Tables A.1 and A.2 summarise the composed test models in terms of their test numbers, the submodels used to construct the model, and the composed entities[1].

Table A.1 Test models used in the first experiment.

Test	Models	Composition
1	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$ and $BN_3(\mathcal{M}_{Ex3})$	$g_1^c = \{g_1^1, g_1^3\}$ $g_2^c = \{g_1^2, g_3^3\}$
2	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$ and $\mathcal{BN}_5(\mathcal{M}_{Ex5})$	$g_1^c = \{g_1^1, g_1^3\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4\}$
3	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$ and $BN_7(\mathcal{M}_{Ex3})$	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4\}$ $g_4^c = \{g_3^5, g_1^6\}$
4	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$ and $BN_9(\mathcal{M}_{Ex3})$	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_4^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$
5	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex1})$ and $BN_{11}(\mathcal{M}_{Ex6})$	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}\}$ $g_6^c = \{g_3^{10}, g_1^{11}\}$
6	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex2})$, $BN_{12}(\mathcal{M}_{Ex1})$ and $BN_{13}(\mathcal{M}_{Ex5})$	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}\}$

Continued on next page

[1]For more information about the test models; please email the authors.

Table A.1 – continued from previous page

Test	Models	Composition
7	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex2})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex3})$, $BN_{14}(\mathcal{M}_{Ex1})$ and $BN_{15}(\mathcal{M}_{Ex4})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}\}$ $g_7^c = \{g_2^{12}, g_1^{14}\}$ $g_8^c = \{g_2^{14}, g_1^{15}\}$
8	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex1})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex3})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex3})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex6})$, $BN_{16}(\mathcal{M}_{Ex1})$ and $BN_{17}(\mathcal{M}_{Ex3})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}\}$
9	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex5})$, $BN_5(\mathcal{M}_{Ex1})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex6})$, $BN_{11}(\mathcal{M}_{Ex3})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex3})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex5})(\mathcal{M}_{Ex4})$, $BN_{18}(\mathcal{M}_{Ex3})$ and $BN_{19}(\mathcal{M}_{Ex3})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_5^{10}, g_1^{11}, g_1^{12}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}\}$ $g_9^c = \{g_2^{13}, g_1^{19}\}$

Continued on next page

Table A.1 – continued from previous page

Test	Models	Composition
10	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex3})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex3})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex5})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex1})$, $BN_{20}(\mathcal{M}_{Ex6})$ and $BN_{21}(\mathcal{M}_{Ex3})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}\}$
11	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex3})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex3})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex5})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex5})$, $BN_{21}(\mathcal{M}_{Ex1})$, $BN_{22}(\mathcal{M}_{Ex3})$ and $BN_{23}(\mathcal{M}_{Ex4})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}\}$
12	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex3})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex3})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex3})$, $BN_{17}(\mathcal{M}_{Ex5})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex1})$, $BN_{20}(\mathcal{M}_{Ex5})$, $BN_{21}(\mathcal{M}_{Ex3})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex4})$, $BN_{24}(\mathcal{M}_{Ex1})$ and $BN_{25}(\mathcal{M}_{Ex5})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}\}$

Continued on next page

Table A.1 – continued from previous page

Test	Models	Composition
13	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex5})$, $BN_9(\mathcal{M}_{Ex4})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex3})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex1})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex5})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex3})$, $BN_{21}(\mathcal{M}_{Ex3})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex4})$, $BN_{24}(\mathcal{M}_{Ex3})$, $BN_{25}(\mathcal{M}_{Ex1})$, $BN_{26}(\mathcal{M}_{Ex1})$ and $BN_{27}(\mathcal{M}_{Ex3})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}, g_1^{26}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}, g_1^{27}\}$
14	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex1})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex1})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex5})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex1})$, $BN_{21}(\mathcal{M}_{Ex3})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex6})$, $BN_{24}(\mathcal{M}_{Ex3})$, $BN_{25}(\mathcal{M}_{Ex5})$, $BN_{26}(\mathcal{M}_{Ex5})$, $BN_{27}(\mathcal{M}_{Ex3})$, $BN_{28}(\mathcal{M}_{Ex3})$ and $BN_{29}(\mathcal{M}_{Ex1})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}, g_1^{26}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}, g_1^{27}\}$ $g_{10}^c = \{g_3^{27}, g_1^{28}, g_1^{29}\}$
15	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex1})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex1})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex3})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex4})$, $BN_{21}(\mathcal{M}_{Ex3})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex3})$, $BN_{24}(\mathcal{M}_{Ex3})$, $BN_{25}(\mathcal{M}_{Ex5})$, $BN_{26}(\mathcal{M}_{Ex5})$, $BN_{27}(\mathcal{M}_{Ex3})$, $BN_{28}(\mathcal{M}_{Ex3})$, $BN_{29}(\mathcal{M}_{Ex3})$, $BN_{30}(\mathcal{M}_{Ex6})$ and $BN_{31}(\mathcal{M}_{Ex1})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}, g_1^{26}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}, g_1^{27}\}$ $g_{10}^c = \{g_3^{27}, g_1^{28}, g_1^{29}, g_1^{30}, g_1^{31}\}$

Continued on next page

Table A.1 – continued from previous page

Test	Models	Composition
16	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex1})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex1})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex3})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex4})$, $BN_{21}(\mathcal{M}_{Ex3})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex3})$, $BN_{24}(\mathcal{M}_{Ex3})$, $BN_{25}(\mathcal{M}_{Ex5})$, $BN_{26}(\mathcal{M}_{Ex5})$, $BN_{27}(\mathcal{M}_{Ex3})$, $BN_{28}(\mathcal{M}_{Ex3})$, $BN_{29}(\mathcal{M}_{Ex3})$, $BN_{30}(\mathcal{M}_{Ex6})$, $BN_{31}(\mathcal{M}_{Ex1})$, $BN_{32}(\mathcal{M}_{Ex3})$, $BN_{33}(\mathcal{M}_{Ex5})$ and $BN_{34}(\mathcal{M}_{Ex1})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}, g_1^{32}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}, g_1^{33}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8, g_1^{34}\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}, g_1^{26}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}, g_1^{27}\}$ $g_{10}^c = \{g_3^{27}, g_1^{28}, g_1^{29}, g_1^{30}, g_1^{31}\}$
17	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex1})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex1})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex3})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex4})$, $BN_{21}(\mathcal{M}_{Ex3})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex3})$, $BN_{24}(\mathcal{M}_{Ex3})$, $BN_{25}(\mathcal{M}_{Ex5})$, $BN_{26}(\mathcal{M}_{Ex5})$, $BN_{27}(\mathcal{M}_{Ex3})$, $BN_{28}(\mathcal{M}_{Ex3})$, $BN_{29}(\mathcal{M}_{Ex3})$, $BN_{30}(\mathcal{M}_{Ex6})$, $BN_{31}(\mathcal{M}_{Ex1})$, $BN_{32}(\mathcal{M}_{Ex3})$, $BN_{33}(\mathcal{M}_{Ex5})$, $BN_{34}(\mathcal{M}_{Ex1})$,$BN_{35}(\mathcal{M}_{Ex1})$, $BN_{36}(\mathcal{M}_{Ex5})$ and $BN_{37}(\mathcal{M}_{Ex3})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}, g_1^{32}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}, g_1^{33}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8, g_1^{34}\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}, g_1^{35}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}, g_1^{36}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}, g_1^{26}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}, g_1^{27}, g_1^{39}\}$ $g_{10}^c = \{g_3^{27}, g_1^{28}, g_1^{29}, g_1^{30}, g_1^{31}\}$

Continued on next page

Table A.1 – continued from previous page

Test	Models	Composition
18	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex1})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex1})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex3})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex4})$, $BN_{21}(\mathcal{M}_{Ex3})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex3})$, $BN_{24}(\mathcal{M}_{Ex3})$, $BN_{25}(\mathcal{M}_{Ex5})$, $BN_{26}(\mathcal{M}_{Ex5})$, $BN_{27}(\mathcal{M}_{Ex3})$, $BN_{28}(\mathcal{M}_{Ex3})$, $BN_{29}(\mathcal{M}_{Ex3})$, $BN_{30}(\mathcal{M}_{Ex6})$, $BN_{31}(\mathcal{M}_{Ex1})$, $BN_{32}(\mathcal{M}_{Ex3})$, $BN_{33}(\mathcal{M}_{Ex5})$, $BN_{34}(\mathcal{M}_{Ex1})$,$BN_{35}(\mathcal{M}_{Ex1})$, $BN_{36}(\mathcal{M}_{Ex1})$, $BN_{37}(\mathcal{M}_{Ex3})$,$BN_{38}(\mathcal{M}_{Ex3})$, $BN_{39}(\mathcal{M}_{Ex3})$ and $BN_{40}(\mathcal{M}_{Ex5})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}, g_1^{32}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}, g_1^{33}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8, g_1^{34}\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}, g_1^{35}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}, g_1^{36}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}, g_1^{37}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}, g_1^{38}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}, g_1^{26}, g_1^{39}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}, g_1^{27}, g_1^{40}\}$ $g_{10}^c = \{g_3^{27}, g_1^{28}, g_1^{29}, g_1^{30}, g_1^{31}\}$
19	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex1})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex4})$, $BN_8(\mathcal{M}_{Ex3})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex1})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex3})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex5})$, $BN_{21}(\mathcal{M}_{Ex3})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex3})$, $BN_{24}(\mathcal{M}_{Ex3})$, $BN_{25}(\mathcal{M}_{Ex5})$, $BN_{26}(\mathcal{M}_{Ex5})$, $BN_{27}(\mathcal{M}_{Ex3})$, $BN_{28}(\mathcal{M}_{Ex3})$, $BN_{29}(\mathcal{M}_{Ex3})$, $BN_{30}(\mathcal{M}_{Ex4})$, $BN_{31}(\mathcal{M}_{Ex1})$, $BN_{32}(\mathcal{M}_{Ex3})$, $BN_{33}(\mathcal{M}_{Ex5})$, $BN_{34}(\mathcal{M}_{Ex3})$,$BN_{35}(\mathcal{M}_{Ex1})$, $BN_{36}(\mathcal{M}_{Ex1})$, $BN_{37}(\mathcal{M}_{Ex3})$,$BN_{38}(\mathcal{M}_{Ex3})$, $BN_{39}(\mathcal{M}_{Ex3})$, $BN_{40}(\mathcal{M}_{Ex5})$, $BN_{41}(\mathcal{M}_{Ex1})$, $BN_{42}(\mathcal{M}_{Ex5})$, and $BN_{43}(\mathcal{M}_{Ex3})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}, g_1^{32}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}, g_1^{33}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8, g_1^{34}\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}, g_1^{35}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}, g_1^{36}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}, g_1^{37}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}, g_1^{38}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}, g_1^{26}, g_1^{39}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}, g_1^{27}, g_1^{40}\}$ $g_{10}^c = \{g_3^{27}, g_1^{28}, g_1^{29}, g_1^{30}, g_1^{31}, g_1^{41}\}$ $g_{11}^c = \{g_1^{41}, g_1^{42}, g_1^{43}\}$

Continued on next page

Table A.1 – continued from previous page

Test	Models	Composition
20	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex1})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex4})$, $BN_8(\mathcal{M}_{Ex3})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex1})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex3})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex5})$, $BN_{21}(\mathcal{M}_{Ex3})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex3})$, $BN_{24}(\mathcal{M}_{Ex3})$, $BN_{25}(\mathcal{M}_{Ex5})$, $BN_{26}(\mathcal{M}_{Ex5})$, $BN_{27}(\mathcal{M}_{Ex3})$, $BN_{28}(\mathcal{M}_{Ex3})$, $BN_{29}(\mathcal{M}_{Ex3})$, $BN_{30}(\mathcal{M}_{Ex5})$, $BN_{31}(\mathcal{M}_{Ex1})$, $BN_{32}(\mathcal{M}_{Ex3})$, $BN_{33}(\mathcal{M}_{Ex5})$, $BN_{34}(\mathcal{M}_{Ex3})$,$BN_{35}(\mathcal{M}_{Ex1})$, $BN_{36}(\mathcal{M}_{Ex1})$, $BN_{37}(\mathcal{M}_{Ex3})$,$BN_{38}(\mathcal{M}_{Ex3})$, $BN_{39}(\mathcal{M}_{Ex3})$, $BN_{40}(\mathcal{M}_{Ex5})$, $BN_{41}(\mathcal{M}_{Ex1})$, $BN_{42}(\mathcal{M}_{Ex5})$, $BN_{43}(\mathcal{M}_{Ex3})$,$BN_{44}(\mathcal{M}_{Ex5})$, $BN_{45}(\mathcal{M}_{Ex1})$, and $BN_{46}(\mathcal{M}_{Ex3})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}, g_1^{32}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}, g_1^{33}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8, g_1^{34}\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}, g_1^{35}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}, g_1^{36}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}, g_1^{37}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}, g_1^{38}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}, g_1^{26}, g_1^{39}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}, g_1^{27}, g_1^{40}\}$ $g_{10}^c = \{g_3^{27}, g_1^{28}, g_1^{29}, g_1^{30}, g_1^{31}, g_1^{41}\}$ $g_{11}^c = \{g_1^{41}, g_1^{42}, g_1^{43}, g_2^{45}\}$ $g_{12}^c = \{g_1^{44}, g_1^{45}, g_1^{46}\}$

Table A.2 Test models used in the second experiment.

Test	Models	Composition
1	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$ and $BN_3(\mathcal{M}_{Ex7})$	$g_1^c = \{g_1^1, g_1^3\}$ $g_2^c = \{g_1^2, g_4^3\}$
2	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$ and $\mathcal{BN}_5(\mathcal{M}_{Ex7})$	$g_1^c = \{g_1^1, g_1^3\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_4^5, g_1^4\}$
3	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex7})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$ and $BN_7(\mathcal{M}_{Ex3})$	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_4^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4\}$ $g_4^c = \{g_3^5, g_1^6\}$
4	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex7})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$ and $BN_9(\mathcal{M}_{Ex3})$	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_4^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$

Continued on next page

Table A.2 – continued from previous page

Test	Models	Composition
5	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex7})$ and $BN_{11}(\mathcal{M}_{Ex3})$	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}\}$ $g_6^c = \{g_4^{10}, g_1^{11}\}$
6	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex1})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex7})$, $BN_{11}(\mathcal{M}_{Ex5})$, $BN_{12}(\mathcal{M}_{Ex1})$ and $BN_{13}(\mathcal{M}_{Ex6})$	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_4^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}\}$ $g_6^c = \{g_4^{10}, g_1^{11}, g_1^{12}\}$
7	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex1})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex7})$, $BN_{11}(\mathcal{M}_{Ex2})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex3})$ and $BN_{15}(\mathcal{M}_{Ex6})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}\}$ $g_6^c = \{g_4^{10}, g_1^{11}, g_1^{12}\}$ $g_7^c = \{g_2^{12}, g_1^{14}\}$ $g_8^c = \{g_2^{14}, g_1^{15}\}$
8	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex7})$, $BN_{11}(\mathcal{M}_{Ex3})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex6})$, $BN_{16}(\mathcal{M}_{Ex1})$ and $BN_{17}(\mathcal{M}_{Ex5})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}\}$ $g_6^c = \{g_4^{10}, g_1^{11}, g_1^{12}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}\}$

Continued on next page

Table A.2 – continued from previous page

Test	Models	Composition
9	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex7})$, $BN_{11}(\mathcal{M}_{Ex3})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex6})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex3})$, $BN_{18}(\mathcal{M}_{Ex1})$ and $BN_{19}(\mathcal{M}_{Ex1})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_4^{10}, g_1^{11}, g_1^{12}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}\}$ $g_9^c = \{g_5^{13}, g_1^{19}\}$
10	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex7})$, $BN_{11}(\mathcal{M}_{Ex3})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex3})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex5})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex6})$ and $BN_{21}(\mathcal{M}_{Ex1})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_4^{10}, g_1^{11}, g_1^{12}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}\}$
11	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex7})$, $BN_{11}(\mathcal{M}_{Ex3})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex6})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex5})$, $BN_{18}(\mathcal{M}_{Ex3})$ $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex5})$, $BN_{21}(\mathcal{M}_{Ex1})$, $BN_{22}(\mathcal{M}_{Ex3})$ and $BN_{23}(\mathcal{M}_{Ex4})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_4^{10}, g_1^{11}, g_1^{12}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}\}$

Continued on next page

Table A.2 – continued from previous page

Test	Models	Composition
12	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex1})$, $BN_{11}(\mathcal{M}_{Ex3})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex7})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex3})$, $BN_{17}(\mathcal{M}_{Ex5})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex1})$, $BN_{20}(\mathcal{M}_{Ex5})$, $BN_{21}(\mathcal{M}_{Ex1})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex6})$, $BN_{24}(\mathcal{M}_{Ex3})$ and $BN_{25}(\mathcal{M}_{Ex1})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}\}$ $g_8^c = \{g_4^{14}, g_1^{15}, g_1^{17}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}\}$
13	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex2})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex5})$, $BN_9(\mathcal{M}_{Ex4})$, $BN_{10}(\mathcal{M}_{Ex3})$, $BN_{11}(\mathcal{M}_{Ex1})$, $BN_{12}(\mathcal{M}_{Ex7})$, $BN_{13}(\mathcal{M}_{Ex6})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex3})$, $BN_{18}(\mathcal{M}_{Ex1})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex5})$, $BN_{21}(\mathcal{M}_{Ex1})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex5})$, $BN_{24}(\mathcal{M}_{Ex3})$, $BN_{25}(\mathcal{M}_{Ex3})$, $BN_{26}(\mathcal{M}_{Ex5})$ and $BN_{27}(\mathcal{M}_{Ex1})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}\}$ $g_7^c = \{g_4^{12}, g_1^{14}, g_1^{16}, g_1^{25}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}, g_1^{26}\}$ $g_9^c = \{g_5^{13}, g_1^{19}, g_1^{21}, g_1^{27}\}$
14	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex1})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex1})$, $BN_{10}(\mathcal{M}_{Ex7})$, $BN_{11}(\mathcal{M}_{Ex1})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex5})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex5})$, $BN_{21}(\mathcal{M}_{Ex1})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex6})$, $BN_{24}(\mathcal{M}_{Ex3})$, $BN_{25}(\mathcal{M}_{Ex5})$, $BN_{26}(\mathcal{M}_{Ex5})$, $BN_{27}(\mathcal{M}_{Ex3})$, $BN_{28}(\mathcal{M}_{Ex3})$ and $BN_{29}(\mathcal{M}_{Ex1})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_4^{10}, g_1^{11}, g_1^{12}, g_1^{24}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}, g_1^{26}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}, g_1^{27}\}$ $g_{10}^c = \{g_3^{27}, g_1^{28}, g_1^{29}\}$

Continued on next page

Table A.2 – continued from previous page

Test	Models	Composition
15	$BN_1(\mathcal{M}_{Ex1})$, $BN_2(\mathcal{M}_{Ex1})$,$BN_3(\mathcal{M}_{Ex3})$, $BN_4(\mathcal{M}_{Ex4})$, $BN_5(\mathcal{M}_{Ex5})$, $BN_6(\mathcal{M}_{Ex6})$, $BN_7(\mathcal{M}_{Ex3})$, $BN_8(\mathcal{M}_{Ex2})$, $BN_9(\mathcal{M}_{Ex3})$, $BN_{10}(\mathcal{M}_{Ex7})$, $BN_{11}(\mathcal{M}_{Ex1})$, $BN_{12}(\mathcal{M}_{Ex1})$, $BN_{13}(\mathcal{M}_{Ex5})$, $BN_{14}(\mathcal{M}_{Ex1})$, $BN_{15}(\mathcal{M}_{Ex4})$, $BN_{16}(\mathcal{M}_{Ex1})$, $BN_{17}(\mathcal{M}_{Ex5})$, $BN_{18}(\mathcal{M}_{Ex3})$, $BN_{19}(\mathcal{M}_{Ex3})$, $BN_{20}(\mathcal{M}_{Ex5})$, $BN_{21}(\mathcal{M}_{Ex3})$, $BN_{22}(\mathcal{M}_{Ex3})$, $BN_{23}(\mathcal{M}_{Ex6})$, $BN_{24}(\mathcal{M}_{Ex3})$, $BN_{25}(\mathcal{M}_{Ex5})$, $BN_{26}(\mathcal{M}_{Ex1})$, $BN_{27}(\mathcal{M}_{Ex3})$, $BN_{28}(\mathcal{M}_{Ex3})$, $BN_{29}(\mathcal{M}_{Ex1})$, $BN_{30}(\mathcal{M}_{Ex5})$ and $BN_{31}(\mathcal{M}_{Ex1})$.	$g_1^c = \{g_1^1, g_1^3, g_1^7, g_1^{22}\}$ $g_2^c = \{g_1^2, g_3^3, g_1^5, g_1^{20}\}$ $g_3^c = \{g_2^5, g_1^4, g_1^8\}$ $g_4^c = \{g_3^5, g_1^6, g_1^9, g_1^{23}\}$ $g_5^c = \{g_2^4, g_1^{10}, g_1^{13}, g_1^{18}\}$ $g_6^c = \{g_3^{10}, g_1^{11}, g_1^{12}, g_1^{24}\}$ $g_7^c = \{g_2^{12}, g_1^{14}, g_1^{16}, g_1^{25}\}$ $g_8^c = \{g_2^{14}, g_1^{15}, g_1^{17}, g_1^{26}\}$ $g_9^c = \{g_2^{13}, g_1^{19}, g_1^{21}, g_1^{27}\}$ $g_{10}^c = \{g_3^{27}, g_1^{28}, g_1^{29}, g_1^{30}, g_1^{31}\}$