A New Decomposition-based Method for Detecting Attractors in Synchronous Boolean Networks

Qixia Yuan[a], Andrzej Mizera[b], Jun Pang[c,*], Hongyang Qu[d]

[a]*Faculty of Science, Technology and Communication, University of Luxembourg*
[b]*Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland*
[c]*Faculty of Science, Technology and Communication & Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg*
[d]*Department of Automatic Control & Systems Engineering, University of Sheffield*

Abstract

Boolean networks are a well-established formalism for modelling biological systems. An important aspect of analysing a Boolean network is to identify all its attractors. This becomes challenging for large Boolean networks due to the infamous state-space explosion problem. In this paper, we propose a new strongly connected component (SCC) based decomposition method for attractor detection in large synchronous Boolean networks and prove its correctness. Experimental results show that our proposed method is significantly better in terms of performance when compared to existing methods in the literature.

Keywords: synchronous Boolean networks, attractor detection, decomposition, binary decision diagram (BDD)

1. Introduction

Boolean networks (BNs) are a well-established framework widely used for modelling biological systems, such as gene regulatory networks (GRNs). Originally proposed by Kauffman for modelling GRNs, BNs have quickly contributed to many other scientific fields, such as fault diagnosis (e.g. see [1]) and robotics applications (e.g. see [2]). Although they provide a simple system representation, BNs can still capture

*Corresponding author
Email addresses: `qixia.yuan@uni.lu` (Qixia Yuan), `andrzej.mizera@gmail.com` (Andrzej Mizera), `jun.pang@uni.lu` (Jun Pang), `h.qu@sheffield.ac.uk` (Hongyang Qu)

the important dynamic properties of the modelled system, e.g. the *attractors*. An attractor is a set of states that a system will stay forever once entered. In the literature, attractors are hypothesised to characterise cellular phenotypes [3] or to correspond to functional cellular states such as proliferation, apoptosis, or differentiation [4]. Hence, attractor identification is of vital importance to the study of biological systems modelled as BNs.

Attractors are defined based on the BN's state space (often represented as a transition system or graph), the size of which is exponential in the number of nodes in the network. Therefore, attractor detection becomes non-trivial when it comes to a large network. In the BN framework, algorithms for detecting attractors have been extensively studied in the literature. The first study dates back to the early 2000s when Somogyi and Greller proposed an enumeration and simulation method [5]. The idea is to enumerate all the possible states and to run simulations from each of them until an attractor is found. This method is largely limited by the network size as its running time grows exponentially with the number of nodes in the BN. In 2006, Irons proposed a method to detect BNs with a special topology [6], making it possible to deal with BNs with maximum 80 nodes. Later on, the performance of attractor detection has been greatly improved with the use of two techniques. The first technique utilizes binary decision diagrams (BDDs). These methods [7, 8] encode Boolean functions of BNs with BDDs and represents the network's corresponding transition system with BDD structures. Using the BDD operations, the forward and backward reachable states can be often efficiently computed. Detecting attractors is then reduced to finding fix point sets of states in the corresponding transition system. The other technique makes use of satisfiability (SAT) solvers. It transforms attractor detection in BNs into a SAT problem [9]. An unfolding of the transition relation of the BN for a bounded number of steps is represented as a propositional formula. The formula is then solved by a SAT solver to identify a valid path in the state transition system of the BN. The procedure is repeated iteratively for larger and larger upper bound on the number of steps until all attractors are identified. The efficiency of this type of algorithms largely relies on the number of unfolding steps required and the number of nodes in the BN. In addition to BDD and SAT based methods, modular and Gröebner bases computation in Boolean rings were also used for the computation of attractors in a BN [10]. The usage of this method, however, scales only to a few tens of nodes. Recently, a few decomposition methods [11, 12, 13, 14, 15] were proposed to deal with large BNs. The main idea is to decompose a large BN into small components based on its structure, to detect attractors of the small components, and then to restore the attractors of the original BN.

In the literature, two different updating schemes have been proposed for BNs. One is the *synchronous* scheme, where all the nodes are updated simultaneously at each time point. In [16], an HGF network has been analysed under the synchronous scheme. Moreover, in [17], a synchronous Boolean network model of the cell-cycle regulatory network of fission yeast (Schizosaccharomyces Pombe) was shown to faithfully reproduce the known activity sequence of regulatory proteins along the cell cycle of the living cell. The other scheme is the *asynchronous* one, where one randomly selected node is updated at each time point. A network of tumour has been analysed under the asynchronous scheme in [18]. In this paper, we propose a new decomposition method for attractor detection in BNs, in particular, in large synchronous BNs. We prove the correctness of our proposed method and demonstrate its performance with experiments on both randomly generated and real-life biological networks. Considering the fact that a few decomposition methods have already been introduced, we explain our new method by showing its main differences from the existing ones. First, our method carefully considers the semantics of synchronous BNs and thus it does not encounter a problem that the method proposed in [11] does, which we explain in more details in Section 3. Second, our previous method [12] does not consider the dependency relation among different sub-networks. Therefore, the state space of a sub-network is not restricted by the networks it depends on. This results in a lot of unnecessary states in the state space of a sub-network. In this newly proposed method, we take into consideration the dependency relation to reduce the state space of a sub-network. Therefore, the performance of our method can potentially be improved. We show with experimental results that this consideration can significantly improve the performance of attractor detection in large BNs. Third, the method in [13] also considers the dependency relation among different sub-networks. However, the dependency relation is not used to restrict the construction of the state space or the state transition graph in a sub-network, but the recovery of attractors. Therefore, the state space and state transition graph in their sub-networks are bigger than ours. Hence, we expect to have a better performance with our method than with theirs. Due to the fact that no tool is provided for their method, we do not perform experimental comparison with their method. Fourth, the method proposed in [14] proposed a different way of partition. Then they over-approximated the set of attractors and generated the exact sets of attractors using global fixed-point iterations. Comparing to the other methods mentioned above, this method is suitable for networks with large average in-degree (also known as number of parent nodes). However, the size of the network this method can handle is reduced to a few tens due to the increase of average in-degree. Further, the decomposition method in [15] is designed for asynchronous networks while here we extend it to synchronous networks. As a consequence, the key operation *realisation* for the synchronous BNs is completely re-designed with respect to the one for asynchronous BNs in [15].

Note that this is an extended and revised version of the conference paper [19]. This extended version has the following two main contributions. Firstly, we have included the proofs for all the theorems and lemmas in the conference paper. The proofs are added after each of the corresponding theorems or lemmas in Sections 3 and 4. With the proofs, our proposed method becomes mathematically sound. Secondly, we have performed further evaluation of our method on 9 real-life biological models (see Section 5.2). The evaluation results on real-life biological networks are consistent with that on randomly generated models. Moreover, by using the same model with different input settings (see the last three models in Table 2), we further demonstrate the fact that our proposed method can gain large speedups for models with a small size of attractors. This is crucial as the number of attractors for real-life biological networks should be small in order to be meaningful.

2. Preliminaries

2.1. Boolean networks

A Boolean network (BN) is composed of two elements: binary-valued nodes, which represent elements of a biological system, and Boolean functions, which represent interactions between the elements. The concept of BNs was first introduced in 1969 by S. Kauffman for analysing the dynamical properties of GRNs [20], where each gene was assumed to be in only one of two possible states: ON/OFF.

Definition 1 (Boolean network)**.** *A* Boolean network $G(V, \boldsymbol{f})$ *consists of a set of nodes* $V = \{v_1, v_2, \ldots, v_n\}$*, also referred to as genes, and a vector of Boolean functions* $\boldsymbol{f} = (f_1, f_2, \ldots, f_n)$*, where* f_i *is a predictor function associated with node* v_i *(*$i = 1, 2, \ldots, n$*). A state of the network is given by a vector* $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$*, where* $x_i \in \{0,1\}$ *is a value assigned to node* v_i*.*

Since the nodes are binary, the state space of a BN is exponential in the number of nodes. Each node $v_i \in V$ has an associated subset of nodes $\{v_{i_1}, v_{i_2}, \ldots, v_{i_{k(i)}}\}$, referred to as the set of *parent nodes* of v_i, where $k(i)$ is the number of parent nodes and $1 \leq i_1 < i_2 < \cdots < i_{k(i)} \leq n$. Starting from an initial state, the BN evolves in time by transiting from one state to another. The state of the network at a discrete time point t ($t = 0, 1, 2, \ldots$) is given by a vector $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))$, where $x_i(t)$ is a binary-valued variable that determines the value of node v_i at time point t. The value of node v_i at time point $t+1$ is given by the predictor function f_i applied to the values of the parent nodes of v_i at time t, i.e. $x_i(t+1) = f_i(x_{i_1}(t), x_{i_2}(t), \ldots, x_{i_{k(i)}}(t))$. For simplicity, with slight abuse of notation, we use $f_i(x_{i_1}, x_{i_2}, \ldots, x_{i_{k(i)}})$ to denote the value of node v_i at the next time step. For any $j \in [1, k(i)]$, node v_{i_j} is called a *parent node* of v_i and v_i is called a *child node* of v_{i_j}.

In general, the Boolean predictor functions can be formed by combinations of any logical operators, e.g. logical AND $\wedge$, OR $\vee$, and NEGATION $\neg$, applied to variables associated with the respective parent nodes. The BNs are divided into two types based on the time evolution of their states, i.e. *synchronous* and *asynchronous*. In synchronous BNs, values of all the variables are updated simultaneously; while in asynchronous BNs, one variable at a time is randomly selected for update.

In this paper, we focus on synchronous BNs. The transition relation of a synchronous BN is given by

$$T\left(\boldsymbol{x}(t), \boldsymbol{x}(t+1)\right) = \bigwedge_{i=1}^{n} \left(x_i(t+1) \leftrightarrow f_i(x_{i_1}(t), x_{i_2}(t), \ldots, x_{i_{k(i)}}(t))\right). \quad (1)$$

Equation 1 states that in every step all the nodes are updated simultaneously in accordance with their Boolean functions.

In many cases, a BN $G(V, \boldsymbol{f})$ is studied as a state transition system. Formally, the definition of a state transition system is given as follows.

Definition 2 (State transition system)**.** *A* state transition system $\mathcal{T}$ *is a 3-tuple* $\langle S, S_0, T\rangle$ *where* S *is a finite set of states,* $S_0 \subseteq S$ *is the initial set of states, and* $T \subseteq S \times S$ *is the transition relation. When* $S = S_0$*, we simply write* $\langle S, T\rangle$*.*

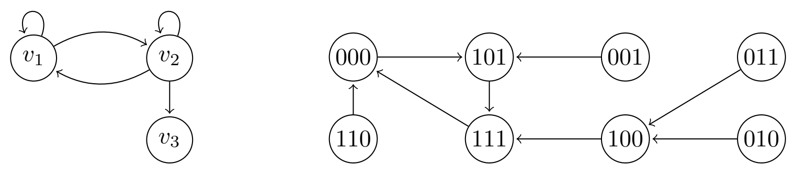
A BN can be easily modelled as a state transition system: the set S is just the state space of the BN, so there are 2^n states for a BN with n nodes; the initial set of states S_0 is the same as S; finally, the transition relation T is given by Equation 1.

In most part of the paper, we focus on the structure of a BN and demonstrate a BN with a graph showing its structure while ignoring the predictor function details. Similarly, we demonstrate a state transition system as a transition graph.

Example 1 (BN)**.** *A BN* G_1 *with 3 nodes is shown in Figure 1a. Its Boolean functions are given as:* $f_1 = \neg(x_1 \wedge x_2)$*,* $f_2 = x_1 \wedge \neg x_2$*, and* $f_3 = \neg x_2$*. In Figure 1a, the three circles* v_1, v_2*, and* v_3 *represent the three nodes of the BN. The edges between nodes represent the interactions between nodes. By applying the transition relation to each of the states, we obtain the corresponding state transition system. For better understanding, we demonstrate the state transition system as a state transition graph in this paper. The corresponding state transition graph of this example is shown in Figure 1b. Throughout this paper, the nodes in a state are always arranged lexicographically based on their subscripts. For example, the nodes in a state in Figure 1b are arranged as* $\{v_1, v_2, v_3\}$*.*

In the transition graph of Figure 1b, the three states $(000), (1*1)$[1] can be reached from each other but no other state can be reached from any of them. This forms an *attractor* of the BN. The formal definition of an *attractor* is given as follows.

[1]We use $*$ to denote that the corresponding bit can have value either 1 or 0, so $(1*1)$ actually denotes two states: 101 and 111.



(a) A BN with 3 nodes. (b) Transition graph of the BN in Example 1.

Figure 1: The Boolean network in Example 1 and its state transition graph.

Definition 3 (Attractors of a state transition system)**.** *An* attractor *of a state transition system is a set of states such that any state in this set can be reached from any other state in this set and no state in this set can reach any other state that is not in this set.*

Attractors of a BN are simply attractors of its corresponding state transition system. The states constituting an attractor are called *attractor states*. When analysing an attractor, we often need to identify transition relations between the attractor states. This leads to the following definition.

Definition 4 (Attractor system)**.** *We call an attractor together with its state transition relation an* attractor system *(AS).*

The attractors of a BN characterise its long-run behaviour [21] and are of particular interest due to their biological interpretation. For synchronous BNs, each state of the network has exactly one outgoing transition. Therefore, the transition graph of an attractor in a synchronous BN is simply a loop. By detecting all the loops in a synchronous BN, one can identify all its attractors.

Example 2 (Attractor)**.** *The BN given in Example 1 has one attractor with three states, i.e.* $\{(000), (1 * 1)\}$.

2.2. Encoding BNs in BDDs

Binary decision diagrams (BDDs) were introduced by Lee in [22] and Akers in [23] to represent Boolean functions [22, 23]. BDDs have the advantage of memory efficiency and have been applied in many model checking algorithms to alleviate the state space explosion problem. A BN $G(V, \boldsymbol{f})$ can be modelled as a state transition system, which can further be encoded into a BDD.

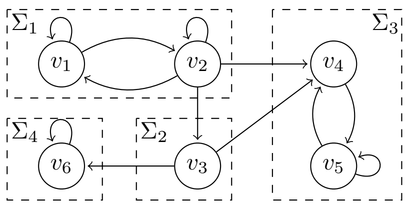
Each variable in V can be represented by a binary BDD variable. By slight abuse of notation, we also use V to denote the set of BDD variables. In order to encode the transition relation, another set V' of BDD variables, which is a copy of V, is introduced: V encodes the possible current states, i.e. $\boldsymbol{x}(t)$, and V' encodes the possible next states, i.e. $\boldsymbol{x}(t+1)$. Hence, the transition relation can be viewed as a Boolean function $T : 2^{|V|+|V'|} \to \{0,1\}$, where values 1 and 0 indicate a valid and an invalid transition, respectively. Our attractor detection algorithm, which will be discussed in the next section, also utilises two basic functions: $\mathit{Image}(X,T) = \{s' \in S \mid \exists s \in X \text{ such that } (s,s') \in T\}$, which returns the set of target states that can be reached from any state in $X \subseteq S$ with a single transition in T; $\mathit{Preimage}(X,T) = \{s' \in S \mid \exists s \in X \text{ such that } (s',s) \in T\}$, which returns the set of predecessor states that can reach a state in X with a single transition. We define the pre-image $\mathit{Predecessors}(X,T)$, which can reach the set of states X in an arbitrary number of steps, by the least fix-point equation $\mu Z.(X \bigcup Z \bigcup \mathit{Preimage}(Z,T))$.

3. The New Method

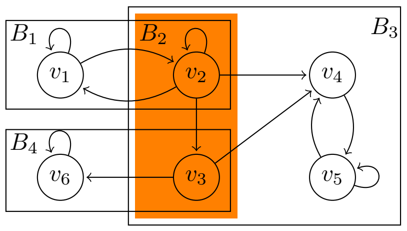
In this section, we describe in details the new SCC-based decomposition method for detecting attractors of large synchronous BNs and we prove its correctness. The method comes from the idea of divide-and-concur and it consists of three steps. First, we divide a BN into sub-networks called *blocks*. This step is performed on the *network structure* and **not** on the state transition system of the network. We want to divide a BN in such a way that the attractors of the original BN can be partially "preserved" in a block. In another word, the concept of attractors still exist in a block and the attractors in a block is related with the attractors of the BN such that we can use the attractors in the blocks to recover attractors of the BN. Second, we detect attractors in individual blocks. To make the attractor detection speed as fast as possible, we consider the dependency relationships between blocks in this process because by taking into consideration the dependency relationship, we can reduce the state-space of a block. Last, we restore attractors of the original BN from the attractors of the blocks.

Before we immerse in explaining the details of our method, we consider an example to demonstrate the method and to provide an intuitive overview in order to ease the understanding of the technicalities that follow.

Example 3 (Attractor detection with the new method)**.** *As the BN G_1 used in Examples 1 is too simple, we extend it to a more complexed BN G_2 in order to better demonstrate our method. The structure of the extended BN is shown in Figure 2a. We highlight the SCCs with rectangles of dashed lines in this figure as they will be used for dividing the network. Compared to G_1, G_2 has three more nodes and the Boolean functions of the three nodes are given as: $f_4 = (x_2 \wedge x_3) \vee x_5$, $f_5 = x_4 \vee x_5$, and $f_6 = \neg x_3 \wedge x_6$. The remaining nodes have the same Boolean functions as the corresponding nodes in G_1.*
Step 1. *We divide the BN into blocks based on the SCCs of the BN's structure. There are four SCCs labelled with Σ_i ($i \in [1, 4]$). For each SCC, we form one block with the nodes in the SCC and the nodes that can directly affect the SCC. Figure 2b shows the division. In this example, the BN is divided into four blocks, namely, block B_1 containing nodes v_1 and v_2; block B_2 containing nodes v_2 and v_3; block B_3 containing nodes v_2, v_3, v_4, and v_5; and block B_4 containing nodes v_3 and v_6. Notice that contrary to SCCs, the blocks can be overlapping. For detailed explanation, we refer to Section 3.1.*
Step 2. *We detect attractors of the four blocks one by one. Since the nodes in block B_1 do not depend on nodes in other blocks, B_1 can be viewed as a small BN. The small BN's attractors can then be easily obtained from its state transition system, which is exponentially smaller than the transition system of the whole BN G_2. The nodes in any of the other blocks, however, depend not only on the nodes inside the considered block, but also on nodes outside the block. When computing attractors for any of these blocks, the relevant outside nodes need to be considered in a specific way. We explain in details the attractor detection of such blocks in Section 3.2.2, but the general idea is that the blocks can be topologically ordered and the attractors of a block can be effectively obtained by considering the attractors of its predecessor blocks. Therefore, the attractors of the blocks are computed in an iterative manner in accordance with the topological ordering of the blocks.*
Step 3. *Once the attractors for the four blocks are computed, the attractors of G_2*



(a) BN G_2 with 6 nodes and its SCCs.



(b) The decomposition of the BN.

Figure 2: BN G_2 with 6 nodes and its decomposition.

can be obtained, or ***restored*** *as we call this operation, by* ***crossing*** *(see Definition 15) the attractors of "terminal" blocks, i.e. blocks that no other blocks depend on. In this example, we need to cross the attractors of blocks B_3 and B_4 since these are the only terminal blocks. G_2 has only one attractor, which is the restored attractor $\mathcal{A} = \{\{(000000), (101000), (111000)\}, \{(000110), (111110), (101010)\}\}$. Details of this step are explained in Section 3.3.*

3.1. Decomposition of a BN

The first step is to divide a BN into sub-networks. The nodes in a sub-network is a subset of the nodes in the original BN and so do the Boolean functions. Due to this, we distinguish two types of sub-networks. First, the nodes in a sub-network do not rely on nodes in other sub-networks. Secondly, at least one node in a sub-network relies on nodes in other sub-networks. In the first case, the sub-network in fact is a small BN and the attractors can be identified with existing methods as mentioned in Section 1. In the second case, the attractors rely on other sub-networks as at least one node relies on nodes in other sub-networks. The problem will become complexed if two sub-networks rely on each other or the dependency relationships form a loop. Therefore, we will decompose a BN in such a way that the dependency relationships in sub-networks do not form a loop. Formally, we name a sub-network as a block and give its definition as follows.

Definition 5 (Block)**.** *Given a BN $G(V, \boldsymbol{f})$ with $V = \{v_1, v_2, \ldots, v_n\}$ and $\boldsymbol{f} = \{f_1, f_2, \ldots, f_n\}$, a* block *$B(V^B, \boldsymbol{f^B})$ is a subset of the network, where $V^B \subseteq V$. For any* *node $v_i \in V^B$, its Boolean function is exactly the same as that in G if B contains all the parent nodes of v_i based on the function f_i. We refer to this kind of nodes as* determined *nodes. Otherwise, the node is referred as an* undermined *node and its Boolean function is undetermined, meaning that additional information is required to determine the value of v_i in B. We refer to a block as an* elementary block *if it contains no undetermined nodes.*

Based on the above definition, a BN is in fact an elementary block. We consider synchronous networks in this paper and therefore a block is also under the synchronous updating scheme, i.e. all the nodes in the block will be updated simultaneously at each time point no matter this node is undetermined or not.

We now introduce a method to construct blocks using SCC-based decomposition. Formally, the standard graph-theoretical definition of an SCC is as follows.

Definition 6 (SCC)**.** *Let $\mathcal{G}$ be a directed graph and $\mathcal{V}$ be its vertices. A strongly connected component (*SCC*) of $\mathcal{G}$ is a maximal set of vertices $C \subseteq \mathcal{V}$ such that for every pair of vertices u and v, there is a directed path from u to v and vice versa.*

We first decompose a given BN into SCCs. We use the BN G_2 in Example 3 to demonstrate this decomposition. Figure 2a shows the decomposition of this BN into four SCCs: Σ_i for $i \in [1, 4]$. The SCCs are shown with rectangles of dashed lines. A node outside an SCC that is a parent to a node in the SCC is referred to as a *control node* of this SCC. In Figure 2a, node v_2 is a control node of Σ_2. The SCC Σ_1 does not have any control node. An SCC together with its control nodes forms a *block*. For example, in Figure 2a, Σ_2 and its control node v_2 form one block B_2. Recall the blocks in Figure 2b.

Definition 7 (Parent SCC, Ancestor SCC)**.** *An SCC Σ_i is called a* parent SCC *(or* parent *for short) of another SCC Σ_j if Σ_i contains at least one control node of Σ_j. Σ_i is called a child of Σ_j. Denote $P(\Sigma_i)$ the set of parent SCCs of Σ_i. An SCC Σ_k is called an* ancestor SCC *(or* ancestor *for short) of an SCC Σ_j if and only if either (1) Σ_k is a parent of Σ_j or (2) Σ_k is a parent of $\Sigma_{k'}$, where $\Sigma_{k'}$ is an ancestor of Σ_j. Denote $\Omega(\Sigma_j)$ the set of ancestor SCCs of Σ_i.*

The definition of parent and ancestor can be naturally extended to blocks. For an SCC Σ_j, if it has no parent SCC, then this SCC forms an elementary block; if it has at least one parent, then it must have an ancestor that has no parent, and all its ancestors $\Omega(\Sigma_j)$ together can form an elementary block, which is also a BN. The SCC-based decomposition will usually result in one or more non-elementary blocks.

By adding directed edges from all parent blocks to all their child blocks, we form a directed acyclic graph (DAG) of the blocks. Notice that our proposed attractor detection method is general in the sense that as long as the block graph is guaranteed to be a DAG, other strategies to form blocks can be applied. Two blocks can be *merged* into one larger block. Formally, the merge of two blocks is given below.

Definition 8 (Merge of blocks)**.** *Given two blocks $B_1(V^{B_1}, \boldsymbol{f}^{\boldsymbol{B_1}})$ and $B_2(V^{B_2}, \boldsymbol{f}^{\boldsymbol{B_2}})$ of a BN G, merge of B_1 and B_2 forms a new block $B_{1,2}(V^{B_{1,2}}, \boldsymbol{f}^{\boldsymbol{B_{1,2}}})$, where $V^{B_{1,2}} = V^{B_1} \cup V^{B_2}$ and $\boldsymbol{f}^{\boldsymbol{B_{1,2}}}$ is the set of Boolean functions for the nodes in $V^{B_{1,2}}$. The set $\boldsymbol{f}^{\boldsymbol{B_{1,2}}}$ is defined based on the definition of a block (Definition 5).*

The merging of blocks allows us to construct a single elementary parent block for any non-elementary block. For example, blocks B_1 and B_2 can be merged in a natural way to form a larger block $B_{1,2}$, which is a single elementary parent block of B_3.

Definition 9 (The least single elementary parent block)**.** *Given a block B_i and its elementary parent block B_j in G, B_j is called a single elementary parent block of B_i if B_j contains all the control nodes of B_i. Moreover, if B_j is the minimal elementary block in terms of inclusion which contains all the control nodes of B_i, then B_j is called the least single elementary parent block of B_i.*

Example 4 (The least single elementary parent block)**.** *Recall the blocks in Figure 2b. By merging blocks B_1 and B_4, we obtain a new elementary block $B_{1,4}$, which contains all the control nodes of B_3. Therefore, $B_{1,4}$ is a single elementary parent block of B_3. However, it is not the least single elementary parent block of B_3: by merging B_2 containing all the control nodes of B_3 with block B_1, we obtain another single elementary parent block $B_{1,2}$, which nodes form a proper subset of the nodes in $B_{1,4}$. In fact, $B_{1,2}$ is the least single elementary parent block of B_3.*

A state of a block is a binary vector of length equal to the number of nodes in the block and it determines the values of all the nodes in the block. In this paper, we use a number of operations on the states of a BN and its blocks. Their definitions are given in Definition 10. In addition, we also extend the definition of a path in a graph into the concept of a path in a BN as given by Definition 11.

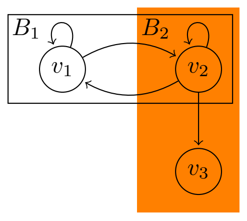
Definition 10 (Projection map, Projected state, Lifting states)**.** *Given a block B, its set of nodes is $V^B = \{v_1, v_2, \dots, v_m, v_{m+1}, \dots, v_n\}$. Let B' be a sub-block of B and its set of nodes is $V^{B'} = \{v_1, v_2, \dots, v_m\}$. Denote the state space of block B and B' as S^B and $S^{B'}$ respectively. The* projection map $\pi_{B'} : S^B \to S^{B'}$ *is given by $\boldsymbol{x} = (x_1, x_2, \dots, x_m, x_{m+1}, \dots, x_n) \mapsto \pi_{B'}(\boldsymbol{x}) = (x_1, x_2, \dots, x_m)$. For any set of states $S \subseteq S^B$, we define $\pi_{B'}(S) = \{\pi_{B'}(\boldsymbol{x}) \mid \boldsymbol{x} \in S\}$. The projected state $\pi_{B'}(\boldsymbol{x})$ is called a* projected state *of $\boldsymbol{x}$. For any state $\boldsymbol{x}^{B'} \in S^{B'}$, we define its set of* lifting states *in B as $\mathcal{M}_B(\boldsymbol{x}^{B'}) = \{\boldsymbol{x} \mid \pi_{B'}(\boldsymbol{x}) = \boldsymbol{x}^{B'}\}$. For any set of states $S^{B'} \subseteq S^{B'}$, its set of lifting states is $\mathcal{M}_B(S^{B'}) = \{\boldsymbol{x} \mid \pi_{B'}(\boldsymbol{x}) \in S^{B'}\}$.*

Definition 11 (Path)**.** *Given a block B of n nodes and its state space S, a* path *of length k ($k \geqslant 2$) in B is a series $\boldsymbol{x}_1 \to \boldsymbol{x}_2 \to \cdots \to \boldsymbol{x}_k$ of states in S such that there exists a transition between any consecutive two states $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$, where $i \in [1, k-1]$.*

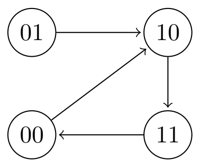
When B is an elementary block, for any two consecutive states $\boldsymbol{x}_i$, $\boldsymbol{x}_{i+1}$ in a path $\boldsymbol{x}_1 \to \boldsymbol{x}_2 \to \cdots \to \boldsymbol{x}_k$ in a BN G, $k \geqslant 2$ and $i \in [1, k-1]$, there is a transition from $\pi_B(\boldsymbol{x}_i)$ to $\pi_B(\boldsymbol{x}_{i+1})$ since the Boolean functions of the nodes in the block B are the same as the ones in the BN G. Therefore, the projection of all the states in the path $\boldsymbol{x}_1 \to \boldsymbol{x}_2 \to \cdots \to \boldsymbol{x}_k$ on block B actually forms a path $\pi_B(\boldsymbol{x}_1) \to \pi_B(\boldsymbol{x}_2) \to \cdots \to \pi_B(\boldsymbol{x}_k)$ in block B. When B is a non-elementary block, the projection will also hold given that the control nodes in the block follow the same transition rules as in the BN.

3.2. Detection of attractors in a block

As mentioned in the beginning of this section, we want to the attractors in each block can partially "preserve" the attractors of the original BN in order to recover the attractors of the original BN from the attractors in the blocks. Therefore, in this step,



(a) Blocks of G_1.



(b) Transition graph of block B_1.

Figure 3: SCC decomposition and a transition graph of a block.

we will introduce how we define attractors in a block to "preserve" the attractor information of the original BN and how we can detect these attractors. An elementary block does not depend on any other block while a non-elementary block does. Therefore, they are treated separately.

3.2.1. Detection of attractors in elementary blocks.

We first consider the case of elementary blocks. Recall that the attractors of a BN or an elementary block are attractors of its corresponding transition state system. As discussed in Section 1, a few existing methods for detecting attractors of a transition state system can be applied to detect attractors of an elementary block. A key point is how one can use the attractors in an elementary block of a BN to detect or recover attractors of the BN. In the following discussion, we show that one can use the lifting states of the attractors of an elementary block to improve the performance for detecting attractors in a BN. We first give the following definition to show the relationship between attractors in an elementary block and attractors in a BN.

Definition 12 (Preservation of attractors)**.** *Given a BN G and an elementary block B in G, let $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ be the set of attractors of G and $\mathcal{A}^B = \{A_1^B, A_2^B, \ldots,$ $A_{m'}^B\}$ be the set of attractors of B. We say that B* preserves the attractors *of G if for any $k \in [1, m]$, there is an attractor $A_{k'}^B \in \mathcal{A}^B$ such that $\pi_B(A_k) = A_{k'}^B$.*

Example 5 (Preservation of attractors)**.** *For simplification, we will use G_1 to demonstrate the preservation of attractors. Its set of attractors is $\mathcal{A} = \{\{(000), (1*1)\}\}$.*

We show the blocks of G_1 in Figure 3a. Nodes v_1 and v_2 form an elementary block B_1. B_1 can be viewed as a BN and its transition graph is shown in Figure 3b. Its set of attractors is $\mathcal{A}^{B_1} = \{\{(00), (1*)\}\}$ (nodes are arranged as v_1, v_2). We have $\pi_{B_1}(\{(000), (1*1)\}) = \{(00), (1*)\} = \mathcal{A}^{B_1}$, i.e. block B_1 preserves the attractors of G_1.

The last equal in Definition 12 can also be replaced with inclusion without affecting other definitions and theorems in this paper. With Definition 12, we have the following theorem and lemma.

Theorem 1. *Given a BN G, let B be an elementary block in G. B preserves the attractors of G.*

Proof. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ be the set of attractors of G. For any $i \in [1, m]$, let $L = \boldsymbol{x}_1 \to \boldsymbol{x}_2 \to \cdots \to \boldsymbol{x}_k$ be a path containing all the states in A_i and let $\boldsymbol{x}_1 = \boldsymbol{x}_k$. In fact, L is an attractor system of A_i. Therefore, $\pi_B(\boldsymbol{x}_1) \to \pi_B(\boldsymbol{x}_2) \to \cdots \to \pi_B(\boldsymbol{x}_k)$ is a path in B. We denote this path as L^B. Given that the choice of the attractor A_i is arbitrary, the claim holds if we can prove that states in the path L^B form an attractor of B. Since $\boldsymbol{x}_1 = \boldsymbol{x}_k$, we have $\pi_B(\boldsymbol{x}_1) = \pi_B(\boldsymbol{x}_k)$. The path L^B is in fact a loop. As B is a synchronous BN, the transitions in B are determined and thus starting from any state in B, no state not in B is reachable. Therefore, the states in the path L^B form an attractor. □

Lemma 1. *Given a BN G and an elementary block B in G, let Φ be the set of attractor states of G and Φ^B be the set of attractor states of B. If B preserves the attractors of G, then $\Phi \subseteq \mathcal{M}_G(\Phi^B)$.*

Proof. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ be the set of attractors of G and $\mathcal{A}^B = \{A_1^B, A_2^B, \ldots, A_{m'}^B\}$ be the set of attractors of B. Since B preserves the attractors of G, for any $k \in [1, m]$, there exists a $k' \in [1, m']$ such that $\pi_B(A_k) \subseteq A_{k'}^B$. Therefore, $\pi_B(\Phi) = \cup_{i=1}^{m} \pi_B(A_i) \subseteq \cup_{i=1}^{m'} A_i^B = \Phi^B$. By Definition 10, we have that $\Phi \subseteq \mathcal{M}_G(\pi_B(\Phi))$. Hence, $\Phi \subseteq \mathcal{M}_G(\Phi^B)$. □

For an elementary block B, the lifting states of its attractor states cover all G's attractor states according to Lemma 1 and Theorem 1. Therefore, by *searching from* *the lifting states only instead of the whole state space*, we can detect all the attractor states of G. We now consider the case of non-elementary blocks.

3.2.2. Detection of attractors in non-elementary blocks.

As mentioned in Section 1, our method considers the dependency relationships among different blocks. This consideration actually applies to attractor detection in non-elementary blocks. The key idea is that if a non-elementary block B_i depends on another block B_j, the attractors of B_j will be used to detect the attractors of B_i to improve the detection speed and to recover the attractors of the original BN later on. We call this process as *realisation* as described later in Definition 16. We first give four definitions to better describe the dependency relationships between blocks.

Definition 13 (Crossability, Cross operations of two states)**.** *Let G be a BN, B_i and B_j be two blocks in G. The two blocks share common nodes $\{v_1, v_2, \ldots, v_t\}$. Let $\boldsymbol{x}^{B_i} = (x_1, x_2, \ldots, x_s, y^i_1, y^i_2, \ldots, y^i_t)$ be a state of B_i and $\boldsymbol{x}^{B_j} = (y^j_1, y^j_2, \ldots, y^j_t, z_1, z_2, \ldots, z_u)$ be a state of B_j, where y^i_k and y^j_k for $k \in [1, t]$ represent the values of v_k in B_i and B_j respectively. States $\boldsymbol{x}^{B_i}$ and $\boldsymbol{x}^{B_j}$ are said to be* crossable*, denoted as $\boldsymbol{x}^{B_i} \ \mathcal{C} \ \boldsymbol{x}^{B_j}$, if the values of their common nodes are the same, i.e. $y^i_k = y^j_k$ for all $k \in [1, t]$. The cross operation of two crossable states $\boldsymbol{x}^{B_i}$ and $\boldsymbol{x}^{B_j}$ is defined as $\Pi(\boldsymbol{x}^{B_i}, \boldsymbol{x}^{B_j}) = (x_1, x_2, \ldots, x_s, y^i_1, y^i_2, \ldots, y^i_t, z_1, z_2, \ldots, z_u)$.*

Example 6 (Crossability of two states)**.** *We continue the demonstration based on Example 5. Consider the state $(000) \in \mathcal{A}$ and the state $(00) \in \mathcal{A}^{B_1}$. The values of their common nodes (v_1 and v_2) are the same. Therefore, $(000) \ \mathcal{C} \ (00)$.*

Note that if two blocks do not have common nodes, any two states of the two blocks are crossable. We continue to give the definition of crossability and cross operations in the level of two sets of states and two families of sets.

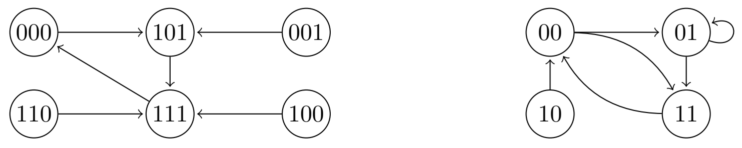
Definition 14 (Crossability, Cross operations of two sets of states)**.** *We say $S^{B_i}_1 \subseteq S^{B_i}$ and $S^{B_j}_1 \subseteq S^{B_j}$ are crossable, denoted as $S^{B_i}_1 \ \mathcal{C} \ S^{B_j}_1$, if at least one of the sets is empty or the following two conditions hold: 1) for any state $\boldsymbol{x}^{B_i} \in S^{B_i}_1$, there exists a state $\boldsymbol{x}^{B_j} \in S^{B_j}_1$ such that $\boldsymbol{x}^{B_i}$ and $\boldsymbol{x}^{B_j}$ are crossable; 2) vice versa. The cross operation* *on two crossable non-empty sets of states* $S_1^{B_i}$ *and* $S_1^{B_j}$ *is defined as* $\Pi(S_1^{B_i}, S_1^{B_j}) = \{\Pi(\boldsymbol{x}^{B_i}, \boldsymbol{x}^{B_j}) \mid \boldsymbol{x}^{B_i} \in S_1^{B_i}, \boldsymbol{x}^{B_j} \in S_1^{B_j}$ *and* $\boldsymbol{x}^{B_i}\ \mathcal{C}\ \boldsymbol{x}^{B_j}\}$*. When one of the two sets is empty, the cross operation simply returns the other set, i.e.* $\Pi(S_1^{B_i}, S_1^{B_j}) = S_1^{B_i}$ *if* $S_1^{B_j} = \emptyset$ *and* $\Pi(S_1^{B_i}, S_1^{B_j}) = S_1^{B_j}$ *if* $S_1^{B_i} = \emptyset$*.*

Example 7 (Crossability of two sets of states)**.** *We continue the demonstration based on Example 6. Denote the state space of* G_1 *as* S *and the state space of* B_1 *as* S^{B_1}*. Then* $\mathcal{A} \subset S$ *and* $\mathcal{A}^{B_1} \subset S^{B_1}$*. For* $(000) \in \mathcal{A}$ *and* $(00) \in \mathcal{A}^{B_1}$*, we have* $(000)\ \mathcal{C}\ (00)$*. For* $(1 * 1) \in \mathcal{A}$ *and* $(1*) \in \mathcal{A}^{B_1}$*, we have* $(1 * 1)\ \mathcal{C}\ (1*)$*. Therefore,* $\mathcal{A}\ \mathcal{C}\ \mathcal{A}^{B_1}$*.*

Definition 15 (Crossability, Cross operations of two families of sets)**.** *Let* $\mathcal{S}^{B_i} = \{S_1^{B_i} \mid S_1^{B_i} \subseteq S^{B_i}\}$ *be a family of sets of states in* B_i *and* $\mathcal{S}^{B_j} = \{S_1^{B_j} \mid S_1^{B_j} \subseteq S^{B_j}\}$ *be a family of sets of states in* B_j*. We say* $\mathcal{S}^{B_i}$ *and* $\mathcal{S}^{B_j}$ *are crossable, denoted as* $\mathcal{S}^{B_i}\ \mathcal{C}\ \mathcal{S}^{B_j}$ *if 1) for any set* $S_1^{B_i} \in \mathcal{S}^{B_i}$*, there exists a set* $S_1^{B_j} \in \mathcal{S}^{B_j}$ *such that* $S_1^{B_i}$ *and* $S_1^{B_j}$ *are crossable; 2) vice versa. The cross operation on two crossable families of sets* $\mathcal{S}^{B_i}$ *and* $\mathcal{S}^{B_j}$ *is defined as* $\Pi(\mathcal{S}^{B_i}, \mathcal{S}^{B_j}) = \{\Pi(S_i, S_j) \mid S_i \in \mathcal{S}^{B_i}, S_j \in \mathcal{S}^{B_j}$ *and* $S_i\ \mathcal{C}\ S_j\}$*.*

Example 8 (Crossability of two families of sets)**.** *We continue the demonstration based on Example 7. We define a new set of states* $\{(000)\} \subset S$ *and a new set of states* $\{(00)\} \subset S^{B_1}$*. Therefore,* $\{(000)\}\ \mathcal{C}\ \{(00)\}$*. We denote the family of set* $\{\mathcal{A}, \{(000)\}\}$ *as* $\mathcal{S}$ *and the family of set* $\{\mathcal{A}^{B_1}, \{(00)\}\}$ *as* $\mathcal{S}^{B_1}$*. Since* $\mathcal{A}\ \mathcal{C}\ \mathcal{A}^{B_1}$*, we have* $\mathcal{S}\ \mathcal{C}\ \mathcal{S}^{B_1}$*.*

After decomposing a BN into SCCs, there is at least one SCC with no control nodes. Hence, there is at least one elementary block in every BN. Moreover, for each non-elementary block we can construct, by merging all its predecessor blocks, a single parent elementary block. We detect the attractors of the elementary blocks and use the detected attractors to guide the values of the control nodes of their child blocks. The guidance is achieved by considering *realisations of the dynamics of a non-elementary block with respect to the attractors of its parent elementary block*, shortly referred to as *realisations of a non-elementary block*. In some cases, a realisation of a non-elementary block is simply obtained by assigning new Boolean functions to the control nodes of the block. However, in many cases, it is not this simple and a realisation of



(a) For the realisation in Example 9. (b) For the incorrect "realisation" in Example 10.

Figure 4: Two transition graphs used in Example 9 and Example 10.

a non-elementary block is obtained by explicitly constructing a transition system of this block corresponding to the considered attractor of the elementary parent block. Since the parent block of a non-elementary block may have more than one attractor, a non-elementary block may have more than one realisation.

Definition 16 (Realisation of a non-elementary block)**.** *Let B_i be a non-elementary block formed by merging a single SCC with its control nodes. Let nodes $u_1, u_2, \ldots, u_r$ be all the control nodes of B_i and let them all be contained in block B_j being the least single elementary parent block of B_i. Denote as $B_{i,j}$ the block obtained by merging B_i and B_j. Let $A_1^{B_j}, A_2^{B_j}, \ldots, A_t^{B_j}$ be the attractor systems of B_j. For any $k \in [1, t]$, a* realisation *of block B_i with respect to $A_k^{B_j}$ is a state transition system such that: 1) a state of the system is a vector of the values of all the nodes in the block B_i and its ancestor blocks; 2) the state space of this realisation is crossable with $A_k^{B_j}$; 3) for any transition $\boldsymbol{x}^{B_j} \to \tilde{\boldsymbol{x}}^{B_j}$ in the attractor system of $A_k^{B_j}$, there is a transition $\boldsymbol{x}^{B_{i,j}} \to \tilde{\boldsymbol{x}}^{B_{i,j}}$ in the realisation such that $\boldsymbol{x}^{B_{i,j}} \mathcal{C}\, \boldsymbol{x}^{B_j}$ and $\tilde{\boldsymbol{x}}^{B_{i,j}} \mathcal{C}\, \tilde{\boldsymbol{x}}^{B_j}$; 4) each transition in the realisation is caused by the update of all nodes synchronously: the update of non-control nodes of B_i is regulated by the Boolean functions of the nodes and the update of nodes in its parent block B_j is regulated by the transitions of the attractor system of $A_k^{B_j}$.*

Example 9 (Realisation)**.** *We continue to use the BN G_1 in Example 1 to show how to construct realisations. In this BN, the parent block of the non-elementary block Σ_2 only contains one attractor. We form one realisation with respect to this attractor by assigning transitions $\{(00) \to (10), (10) \to (11), (11) \to (00)\}$ to nodes v_1 and v_2 (nodes from its parent block B_1). The transitions of node v_3 will be in accordance with* *its Boolean function $f_3 = \neg x_2$. Combining these transitions, we have the transitions for the three nodes as follows.* $\{(00*) \rightarrow (101), (10*) \rightarrow (111), (11*) \rightarrow (000)\}$. *With the transitions, we can easily get the realisation (state transition system). We demonstrate its transition graph in Figure 4a.*

In the realisation of a non-elementary block all the nodes of its least single elementary parent block are considered and not only the control nodes of the parent block. This allows to distinguish the potentially different states in which the values of control nodes are the same. Without this, a state in the state transition graph of the realisation may have more than one outgoing transition, which is contrary to the fact that the outgoing transition for a state in a synchronous network is always determined. Although the definition of attractors can still be applied to such a transition graph, the attractor detection algorithms for synchronous networks, e.g. SAT-based algorithms, may not work any more. Moreover, the meaning of attractors in such a graph is not consistent with the synchronous semantics and therefore the detected "attractors" may not be attractors of the synchronous BN. Note that the decomposition method introduced in [11] does not take this issue into account and therefore produces incorrect results in certain cases. We now give an example in Example 10 to illustrate one of such cases.

Example 10 (Counter example)**.** *Consider the BN in Example 1, which can be divided into two blocks: block B_1 with nodes v_1, v_2 and block B_2 with nodes v_2, v_3. The transition graph of B_1 is shown in Figure 3b and its attractor is* $(00) \rightarrow (10) \rightarrow (11)$. *If we do not include the node v_1 when forming the realisation of B_2, we will get a transition graph as shown in Figure 4b, which contains two states with two outgoing transitions. This is contrary to the synchronous semantics.*

For asynchronous networks, however, such a distinction is not necessary since the situation of multiple outgoing transitions is inconsistent with the asynchronous updating semantics. Definition 16 forms the basis for a key difference between this decomposition method for synchronous BNS and the one for asynchronous BNs proposed in [15].

Constructing realisations for a non-elementary block is a key step in obtaining its attractors. For each realisation, the construction process requires the knowledge of all the transitions in the corresponding attractor of its elementary parent block. In Section 4, we explain in details how to implement it with BDDs. We now give an example to show how we construct realisations.

A realisation of a non-elementary block takes care of the dynamics of the undetermined nodes, providing a state transition system of the block. Therefore, attractors of a realisation is an attractors of its corresponding state transition system. Moreover, we extend the attractors of a non-elementary blocks as follows.

Definition 17 (Attractors of a non-elementary block)**.** *The* attractors *of a non-elementary block is the set of the attractors of all realisations of the block.*

With this definition, we can extend Definition 12 of preservation of attractors in a straightforward way to also include non-elementary blocks. Therefore, the notion of preservation of attractors is valid for any block. Moreover, we can extend Definition 16 by allowing B_j to be a non-elementary block. As long as B_i's parent block B_j contains all the control nodes of block B_i, the attractors of B_j can be used to form the realisations of B_i, no matter B_j is elementary or not. Observe that using a non-elementary block as a parent block does not change the fact that the attractor states of the parent block contain the values of all the nodes in the current block and all its ancestor blocks.

Computing attractors of non-elementary blocks requires the knowledge of the attractors of their parent blocks. Therefore, we need to order the blocks so that for any block B_i, the attractors of its parent blocks are always detected prior to the attractors of block B_i being considered. This can be achieved by considering a topological ordering of a directed graph of blocks, where the edges represent the relation of being a parent: there is an edge from block B_j to block B_i if and only if B_j is parent of B_i. However, instead of constructing such a graph, we introduce the concept of *depth* as follows.

Definition 18 (Depth)**.** *Given a BN G, an elementary block B_i of G has a* depth *of 0, denoted as $\mathcal{P}(B_i) = 0$. Let B_j be a non-elementary block and $B_{j_1}, \dots, B_{j_{p(j)}}$ be all its parent blocks. The* depth *of B_j is defined as $\mathcal{P}(B_j) = max_{k=1}^{p(j)}(\mathcal{P}(B_{j_k})) + 1$, where $p(j)$ is the number of parent blocks of B_j.*

By this definition, the depth of a parent block is always lower than of its child block and a topological ordering is obtained by considering the blocks in the ascending order of their depths. With this ordering, the attractors of a block with a lower depth value are always detected first and are already available when determining the attractors of descendant blocks.

3.3. Restoring attractors of the original BN

After computing attractors for all the blocks, we need to restore the attractors of the original BN. Our SCC-based decomposition may result in two cases. In the first case, there is only one leaf block (a block without child block), and the ancestor blocks together with this leaf block form the original BN. In this case, we assume that the attractors of the leaf block is the attractors of the original BN based on Definitions 16 and 17. In the second case, there are more than one leaf blocks. We assume that the attractors of the original BN can be recovered by merging the attractors of the leaf blocks. We now give the following two theorems to show that our assumption is correct.

Theorem 2. *Let G be a BN and let B_i be one of its blocks. Denote as $\mathcal{X}(B_i)$ the block formed by merging B_i with its ancestor blocks. The attractors of block B_i are the attractors of $\mathcal{X}(B_i)$.*

Proof. $\mathcal{X}(B_i)$ is an elementary block, which is also a BN. If B_i is an elementary block, B_i is the same as $\mathcal{X}(B_i)$ and the claim holds. We now prove the case where B_i is a non-elementary block. This is equivalent to proving the following two statements: 1) any attractor of B_i is an attractor in $\mathcal{X}(B_i)$; 2) any attractor in $\mathcal{X}(B_i)$ is an attractor of B_i.
Statement 1: Let A^{B_i} be an attractor of B_i and let $\boldsymbol{x}_1^{B_i} \to \boldsymbol{x}_2^{B_i} \to \cdots \to \boldsymbol{x}_k^{B_i}$ be a path L^{B_i} containing all the states in this attractor with $\boldsymbol{x}_1^{B_i} = \boldsymbol{x}_k^{B_i}$. For any state $\boldsymbol{x}_\ell^{B_i}$ in this path, $\boldsymbol{x}_\ell^{B_i}$ is also a state in the block $\mathcal{X}(B_i)$ as $\boldsymbol{x}_\ell^{B_i}$ is a vector formed by the values of nodes in B_i and all its ancestors. In the transition $\boldsymbol{x}_\ell^{B_i} \to \boldsymbol{x}_{\ell+1}^{B_i}$, the nodes in block B_i are updated by their Boolean functions and the nodes that are not in B_i are updated in accordance with the attractor that forms the corresponding realisation. Therefore, all the nodes are updated in accordance with their Boolean functions. Hence, such a transition $\boldsymbol{x}_\ell^{B_i} \to \boldsymbol{x}_{\ell+1}^{B_i}$ also exists in the block $\mathcal{X}(B_i)$. Path L^{B_i} is therefore a path

in $\mathcal{X}(B_i)$. Since $\boldsymbol{x}_1^{B_i} = \boldsymbol{x}_k^{B_i}$, states in the path L^{B_i}, i.e., states in the attractor A^{B_i} form an attractor in $\mathcal{X}(B_i)$.

Statement 2: We prove this by induction. Our assumption is that if the statement holds for all blocks with depth k, then it holds for those with depth $k+1$, where $k \geq 0$. We first consider the base case, that is the statement holds for blocks with depth 0. In this case, B_i equals to $\mathcal{X}(B_i)$. Therefore, the statement holds in this case. We now prove the general case. When B_i has a depth of $k+1$, we denote its least single elementary parent block as B_j. The attractors of B_j are the attractors of $\mathcal{X}(B_j)$. Therefore, the realisations of B_i are formed with respect to the attractors of $\mathcal{X}(B_j)$. Let $A^{\mathcal{X}(B_i)}$ be an attractor of $\mathcal{X}(B_i)$ and $\boldsymbol{x}_1 \to \boldsymbol{x}_2 \to \cdots \to \boldsymbol{x}_k$ be a path $L^{\mathcal{X}(B_i)}$ containing all the states in this attractor and $\boldsymbol{x}_1 = \boldsymbol{x}_k$. Since $\mathcal{X}(B_j)$ is an elementary block, we have that $\pi_{\mathcal{X}(B_j)}(\boldsymbol{x}_1) \to \pi_{\mathcal{X}(B_j)}(\boldsymbol{x}_2) \to \cdots \to \pi_{\mathcal{X}(B_j)}(\boldsymbol{x}_k)$ is a path in $\mathcal{X}(B_j)$ and $\pi_{\mathcal{X}(B_j)}(\boldsymbol{x}_1) = \pi_{\mathcal{X}(B_j)}(\boldsymbol{x}_k)$. Therefore, states in this path form an attractor of $\mathcal{X}(B_j)$, which is also an attractor of B_j, denoted as A^{B_j}. Regarding A^{B_j}, block B_i has a realisation. For any $\ell \in [1,k]$, state $\boldsymbol{x}_\ell$ in the path $L^{\mathcal{X}(B_i)}$ is crossable with $\pi_{B_j}(\boldsymbol{x}_\ell)$ (which is also $\pi_{\mathcal{X}(B_j)}(\boldsymbol{x}_\ell)$). Therefore, $\boldsymbol{x}_\ell$ is also a state in the realisation. Hence, $A^{\mathcal{X}(B_i)}$ is also an attractor in the realisation. Hence, the claim holds. □

Theorem 3. *Given a BN G, where B_i and B_j are its two blocks, let $\mathcal{A}^{B_i}$ and $\mathcal{A}^{B_j}$ be the set of attractors for B_i and B_j, respectively. Let $B_{i,j}$ be the block got by merging the nodes in B_i and B_j. Denote the set of all attractor states of $B_{i,j}$ as $\mathcal{A}^{B_{i,j}}$. If both B_i and B_j are elementary blocks, $\mathcal{A}^{B_i} \,\mathcal{C}\, \mathcal{A}^{B_j}$ and $\cup_{A \in \Pi(\mathcal{A}^{B_i}, \mathcal{A}^{B_j})} A = \mathcal{A}^{B_{i,j}}$.*

Proof. Let B_i and B_j be two elementary blocks of G. If B_i and B_j do not have common nodes, then it holds by definition that $\mathcal{A}^{B_i} \,\mathcal{C}\, \mathcal{A}^{B_j}$. If they have common nodes, their common nodes must form an elementary block. Denote this block as B_c. For any attractor $A^{B_i} \in \mathcal{A}^{B_i}$, $\pi_{B_c}(A^{B_i})$ is an attractor in B_c and $A^{B_i} \,\mathcal{C}\, \pi_{B_c}(A^{B_i})$. Denote the nodes in B_i but not in B_c as N. The nodes in N and their control nodes in B_c (if they have) form a block B^N. We have the following two claims. Claim I: For any attractor A^{B_c} of B_c, there exists an attractor A^{B_N} in B^N such that $A^{B_c} \,\mathcal{C}\, A^{B_N}$. Claim II: For any attractor A^{B_c} of B_c, there exists an attractor $A^{B_i} \in \mathcal{A}^{B_i}$ such that $A^{B_i} \,\mathcal{C}\, A^{B_c}$. We first prove Claim I. If block B^N does not share nodes with B_c, Claim

I holds according to the definition of crossability. If block B^N shares nodes with B_c, the realisations of B^N is constructed based on the attractors of B_c. According to Definition 16, for any attractor A^{B_c} of B_c, a realisation will be constructed and the attractor of this realisation is crossable with A^{B_c}, thus Claim I holds in this case as well. We continue to prove Claim II. Denote the length of attractor A^{B_c} as ℓ^{B_c} and the length of attractor A^{B_N} as ℓ^{B_N}. Let $\boldsymbol{x}^{B_c}$ be a state in A^{B_c} and $\boldsymbol{x}^{B_N}$ be a state in A^{B_N}. Let $\boldsymbol{x}_1 = \Pi(\boldsymbol{x}^{B_c}, \boldsymbol{x}^{B_N})$. Let L be a path starting from state $\boldsymbol{x}_1$ and of length $k = lcm(\ell^{B_c}, \ell^{B_N}) + 1$, where lcm means the lowest common multiple. Since $\boldsymbol{x}^{B_c}$ is an attractor state, $\pi_{B_c}(\boldsymbol{x}_k) = \boldsymbol{x}^{B_c}$. Similarly, $\pi_{B_N}(\boldsymbol{x}_k) = \boldsymbol{x}^{B_N}$. Hence, $\boldsymbol{x}_k = \Pi(\pi_{B_c}(\boldsymbol{x}_k), \pi_{B_N}(\boldsymbol{x}_k)) = \boldsymbol{x}_1$. Therefore, states in L form an attractor of B_i. Hence the claim holds. Similarly, it holds that for any attractor A^{B_c} of B_c, there exists an attractor $A^{B_j} \in \mathcal{A}^{B_j}$ such that $A^{B_j}\ \mathcal{C}\ A^{B_c}$. Now, let A^{B_i} be an attractor in $\mathcal{A}^{B_i}$. Then, $\pi_{B_c}(A^{B_i})$ is an attractor in B_c and by the above, there exists an attractor $A^{B_j} \in \mathcal{A}^{B_j}$ such that $\pi_{B_c}(A^{B_i})\ \mathcal{C}\ A^{B_j}$. Thus, $A^{B_i}\ \mathcal{C}\ A^{B_j}$. By similar argument, for any $A^{B_j} \in \mathcal{A}^{B_j}$ there exists $A^{B_i} \in \mathcal{A}^{B_i}$ such that $A^{B_j}\ \mathcal{C}\ A^{B_i}$. In consequence, $\mathcal{A}^{B_i}\ \mathcal{C}\ \mathcal{A}^{B_j}$.

We now prove that $\cup_{A\in\Pi(\mathcal{A}^{B_i},\mathcal{A}^{B_j})}A = \mathcal{A}^{B_{i,j}}$. Denote $S = \cup_{A\in\Pi(\mathcal{A}^{B_i},\mathcal{A}^{B_j})}A$. This is equivalent to showing the following two statements: 1) for any state $\boldsymbol{s} \in S$, $\boldsymbol{s}$ is in $\mathcal{A}^{B_{i,j}}$; 2) any state in $\mathcal{A}^{B_{i,j}}$ is contained in S. We prove them one by one.

Statement 1: Let A be any set of states in $\Pi(\mathcal{A}^{B_i}, \mathcal{A}^{B_j})$. Then there exists $A^{B_i} \in \mathcal{A}^{B_i}$ and $A^{B_j} \in \mathcal{A}^{B_j}$ such that $A = \Pi(A^{B_i}, A^{B_j})$ and $A^{B_i}\ \mathcal{C}\ A^{B_j}$. Given the choice of A is arbitrary, it is enough to show that any $\boldsymbol{s} \in A$ is an attractor state of $B_{i,j}$. It holds that $\boldsymbol{s} = \Pi(\pi_{B_i}(\boldsymbol{s}), \pi_{B_j}(\boldsymbol{s}))$, where $\pi_{B_i}(\boldsymbol{s}) \in A^{B_i}$ and $\pi_{B_j}(\boldsymbol{s}) \in A^{B_j}$. Let l^{B_i} be the attractor length of A^{B_i} and l^{B_j} be the attractor length of A^{B_j}. Further, let $L = \boldsymbol{s}_1 \rightarrow \boldsymbol{s}_2 \rightarrow \cdots \rightarrow \boldsymbol{s}_k$ be a path starting from state $\boldsymbol{s}$, i.e., $\boldsymbol{s}_1 = \boldsymbol{s}$, with $k = lcm(l^{B_i}, l^{B_j})+1$. Since both B_i and B_j are elementary blocks, it holds that $\pi_{B_i}(\boldsymbol{s}_1) \rightarrow \pi_{B_i}(\boldsymbol{s}_2) \rightarrow \cdots \rightarrow \pi_{B_i}(\boldsymbol{s}_k)$ is a path in B_i and $\pi_{B_j}(\boldsymbol{s}_1) \rightarrow \pi_{B_j}(\boldsymbol{s}_2) \rightarrow \cdots \rightarrow \pi_{B_j}(\boldsymbol{s}_k)$ is a path in B_j. Since $\pi_{B_i}(\boldsymbol{s}_1) = \pi_{B_i}(\boldsymbol{s})$, we have $\pi_{B_i}(\boldsymbol{s}_k) = \pi_{B_i}(\boldsymbol{s})$. Similarly, we have $\pi_{B_j}(\boldsymbol{s}_k) = \pi_{B_j}(\boldsymbol{s})$. Then, $\boldsymbol{s}_k = \Pi(\pi_{B_i}(\boldsymbol{s}_k), \pi_{B_j}(\boldsymbol{s}_k)) = \Pi(\pi_{B_i}(\boldsymbol{s}), \pi_{B_j}(\boldsymbol{s})) = \boldsymbol{s}$. In consequence, $\boldsymbol{s}_1 = \boldsymbol{s} = \boldsymbol{s}_k$ and the states in L form an attractor of $B_{i,j}$ with $\boldsymbol{s}$ being one of its states.

Statement 2: Let $\boldsymbol{s}$ be a state in $\mathcal{A}^{B_{i,j}}$ and let A be the attractor of $B_{i,j}$ containing state $\boldsymbol{s}$. Let $L = \boldsymbol{s} \rightarrow \boldsymbol{s}_1 \rightarrow \boldsymbol{s}_2 \rightarrow \cdots \rightarrow \boldsymbol{s}_k \rightarrow \boldsymbol{s}$ be a path starting end ending with $\boldsymbol{s}$. It holds that $\pi_{B_i}(\boldsymbol{s}) \rightarrow \pi_{B_i}(\boldsymbol{s}_1) \rightarrow \pi_{B_i}(\boldsymbol{s}_2) \rightarrow \cdots \rightarrow \pi_{B_i}(\boldsymbol{s}_k) \rightarrow \pi_{B_i}(\boldsymbol{s})$ is an attractor system in the elementary block B_i. Let us denote the attractor's set of states as A^{B_i}. We have that $\pi_{B_i}(\boldsymbol{s}) \in A^{B_i}$. Similarly, $\pi_{B_j}(\boldsymbol{s})$ belongs to an attractor of B_j, denoted as A^{B_j}. Therefore, $\boldsymbol{s} = \Pi(\pi_{B_i}(\boldsymbol{s}), \pi_{B_j}(\boldsymbol{s})) \in \Pi(A^{B_i}, A^{B_j}) \subseteq S$. Given the arbitrary choice of $\boldsymbol{s}$, the claim holds. $\square$

The above developed theoretical background with Theorem 2 and Theorem 3 being its core result, allows us to design a new decomposition-based approach towards detection of attractors in large synchronous BNs. We describe the idea as follows and also in Algorithm 1. We divide a BN into blocks according to the detected SCCs. We order the blocks in the ascending order based on their depths and detect attractors of the ordered blocks one by one in an iterative way. We start with detecting attractors of elementary blocks (depth 0), and continue to detect attractors of blocks with higher depths after constructing their realisations. According to Theorem 2, by detecting the attractors of a block, we in fact obtain the attractors of the block formed by the current block and all its ancestor blocks. Hence, after the attractors of all the blocks have been detected, either we have obtained the attractors of the original BN or we have obtained the attractors of several elementary blocks of this BN. According to Theorem 3, we can perform a cross operation on any two elementary blocks (depths 0) to restore the attractor states of the two merged blocks. The resulting merged block will form a new elementary block, i.e. one with depth 0. Then, the set of attractor states can be straightforwardly split into the actual attractors by simply performing simulation of the BN dynamics staring from any of the attractor states. Once an attractor is detected, the procedure is repeated for the remaining attractor states. This continues till no attractor states are left. Since the initial set contains attractor states only, no additional states will ever be visited. By iteratively performing the cross operation until a single elementary block containing all the nodes of the BN is obtained, we can restore the attractor states of the original BN. The details of this new algorithm are discussed in the next section.

Algorithm 1 Attractor detection with decomposition-based approach

```
1: procedure AttractorDetect($G$)
2:     Divide a BN $G$ into $k$ block;
3:     Order the blocks ascendingly based on their depths as $\mathcal{B} := [B_1, B_2, \cdots, B_k]$;
4:     $\mathcal{A} := \emptyset$; initialise dictionary $\mathcal{A}^\ell$ to store attractors of the blocks later;
5:     for $B_i \in \mathcal{B}$ do
6:         Detect attractors of $B_i$ and store it in $\mathcal{A}^\ell$;
7:     end for
8:     for $B_i \in \mathcal{B}$ and $B_i$ has no child block do
9:         $S = \Pi(\mathcal{A}^\ell.get(B_i), \mathcal{A})$;        $\mathcal{A}^\ell.get(B_i)$ is the set of attractors of $B_i$
10:        $\mathcal{A} = D(S)$;        //$D(S)$ splits the set of attractor states $S$ into attractors
11:    end for
12:    return $\mathcal{A}$.
13: end procedure
```

Recall the counter example we show in Example 10. If we restored the attractors of the BN based on the attractors of the incorrect "realisation" shown in Figure 4b, we would get a non-attractor state of the original BN, i.e. (001). The attractor for this example computed with the tool from [11] has four states, i.e. $\{(000), (001), (101), (111)\}$. Hence, the state (001) would be included incorrectly.

In addition, we have the following corollary that extends Theorem 3 by allowing B_i and B_j to be non-elementary blocks. This corollary will be used in the next section.

Corollary 1. *Given a BN G, where B_i and B_j are its two blocks, let $\mathcal{A}^{B_i}$ and $\mathcal{A}^{B_j}$ be the set of attractors for B_i and B_j, respectively. Let $B_{i,j}$ be the block got by merging the nodes in B_i and B_j. Denote the set of attractor states of $B_{i,j}$ as $S^{B_{i,j}}$. It holds that $\mathcal{A}^{B_i} \, \mathcal{C} \, \mathcal{A}^{B_j}$ and $\cup_{S \in \Pi(\mathcal{A}^{B_i}, \mathcal{A}^{B_j})} S = S^{B_{i,j}}$.*

Proof. If B_i and B_j are both elementary blocks, the claim holds according to Theorem 3. We now prove the general cases. Denote $\Omega(B_i)$ the block formed by all B_i's ancestor blocks and denote $\mathcal{X}(B_i)$ the block formed with B_i and $\Omega(B_i)$. Denote $\Omega(B_j)$ the block formed by all B_j's ancestor blocks and denote $\mathcal{X}(B_j)$ the block formed with B_j and $\Omega(B_j)$. According to Theorem 2, the attractors of block B_i are in fact the attractors of the elementary block $\mathcal{X}(B_i)$ and the attractors of block B_j are in fact the attractors of the elementary block $\mathcal{X}(B_j)$. Since both $\mathcal{X}(B_i)$ and $\mathcal{X}(B_j)$ are elementary blocks, the claim holds by Theorem 3. $\square$

4. A BDD-based Implementation

We describe the SCC-based attractor detection method in Algorithm 2. As mentioned in Section 2.2, we encode BNs into BDDs; hence most of the operations in this algorithm is performed with BDDs. For example, the DETECT function in line 8, the cross operation in line 18, and the realisation operation in line 26. The realisation of a block B_i with respect to an attractor A is implemented by first projecting the transition relation of the whole set of nodes in a BN onto the subset of nodes in block B_i, and then by restricting the projected transitions with that from the attractor system of A. The projecting and restricting operations are performed with the existence abstraction in BDDs.

Algorithm 2 SCC-based decomposition algorithm

```
1:  procedure SCC_DETECT($G$)
2:      $\mathcal{B}$ := FORM_BLOCK($G$); $\mathcal{A} := \emptyset$; $k$ := size of $\mathcal{B}$;
3:      initialise dictionary $\mathcal{A}^\ell$;                    //$\mathcal{A}^\ell$ is a dictionary storing the set
4:      for $i := 1; i \leq k; i{+}{+}$ do                           // of attractors for various blocks
5:          if $B_i$ is an elementary block then
6:              $\mathcal{T}^{B_i}$ := transition system converted from $B_i$;
7:                                                               //see Section 2.2 for more details
8:              $\mathcal{A}_i$ := DETECT($\mathcal{T}^{B_i}$); $\mathcal{A}^\ell.add((B_i, \mathcal{A}_i))$;
9:          else
10:             $\mathcal{A}_i := \emptyset$;
11:             if $B_i^p$ is the only parent block of $B_i$ then
12:                 $\mathcal{A}_i^p := \mathcal{A}^\ell.getAtt(B_i^p)$;            //obtain attractors of $B_i^p$
13:             else $B^p := (B_1^p, B_2^p, \ldots, B_m^p)$ be the list of parent blocks of $B_i$;
14:                 $B_c := B_1^p$;                               //$B^p$ is ordered based on depths
```

```
15:             for $j := 2; j \leq m; j{+}{+}$ do
16:                 $B_{c,j}$ := a merged block comprised of nodes in $B_c$ and $B^p_j$;
17:                 if $(\mathcal{A}^p_i := \mathcal{A}^\ell.getAtt(B_{c,j})) == \emptyset$ then
18:                     $S := \Pi(\mathcal{A}^\ell.getAtt(B_c), \mathcal{A}^\ell.getAtt(B_j))$; $\mathcal{A}^p_i := D(S)$;
19:                         //$D(S)$ splits the set of attractor states $S$ into attractors
20:                     $\mathcal{A}^\ell.add(B_{c,j}, \mathcal{A}^p_i)$;
21:                 end if
22:                 $B_c := B_{c,j}$;
23:             end for
24:         end if
25:         for $A \in \mathcal{A}^p_i$ do
26:             $\mathcal{T}^{B_i}(A)$ := the realisation of $B_i$ with respect to $A$;
27:             $\mathcal{A}_i := \mathcal{A}_i \cup$ DETECT($\mathcal{T}^{B_i}(A)$);
28:         end for
29:         $\mathcal{A}^\ell.add((B_i, \mathcal{A}_i))$;    //the add operation will not add duplicated items
30:         $\mathcal{A}^\ell.add((B_{i,ancestors}, \mathcal{A}_i))$;
31:                 //$B_{i,ancestors}$ stands for $B_i$ merged with all its ancestor blocks
32:         for any $B^p \in \{B^p_1, B^p_2, \ldots, B^p_m\}$ do          //$B^p_1, B^p_2, \ldots, B^p_m$ are
33:             $\mathcal{A}^\ell.add((B_{i,p}, \mathcal{A}_i))$;                     // parent blocks of $B_i$
34:         end for
35:     end if
36: end for
37: for $B_i \in \mathcal{B}$ and $B_i$ has no child block do
38:     $\mathcal{A} = D(\Pi(\mathcal{A}^\ell.get(B_i), \mathcal{A}))$;
39: end for
40: return $\mathcal{A}$.
41: end procedure
42: procedure FORM_BLOCK($G$)
43:     decompose $G$ into SCCs and form blocks with SCCs and their control nodes;
44:     order the blocks in an ascending order according to their depths;
```

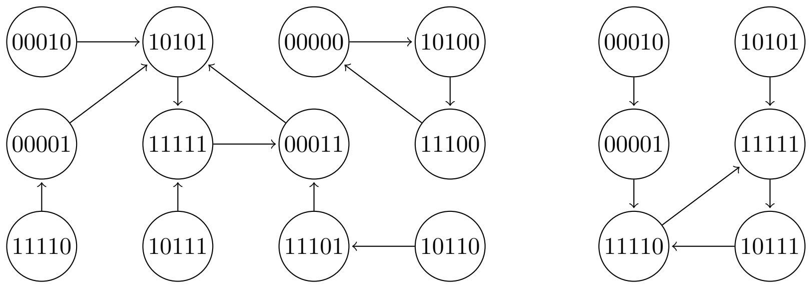
```
45:     return the list of blocks after ordering.
46: end procedure
```

Algorithm 2 takes a BN G as its input, and outputs the set of attractors of G. In this algorithm, we denote by $\textsc{Detect}(\mathcal{T})$ a basic function for detecting attractors of a given transition system $\mathcal{T}$. As mentioned in Section 1, there exist many methods for detecting attractors in a BN. In our implementation, we use the monolithic attractor detection algorithm mentioned in [12]. This is a BDD-based method relying on efficient BDD operations for computing of forward and backward images. Lines 25-28 of this algorithm describe the process for detecting attractors of a non-elementary block. The algorithm detects the attractors of all the realisations of the non-elementary block and performs the union operation on the detected attractors. For this, if the non-elementary block has only one parent block, its attractors are already computed as the blocks are considered in the ascending order with respect to their depths by the main **for** loop in line 4. Otherwise, all the parent blocks are considered in the **for** loop in lines 15-23. By iteratively applying the cross operation in line 18 to the attractor sets of the $ancestor$ blocks in the ascending order, the attractor states of a new block formed by merging all the parent blocks are computed according to Corollary 1. The attractors are then identified from the attractor states with one more operation. The correctness of the algorithm is stated as Theorem 4.

Theorem 4. *Algorithm 2 correctly identifies the set of attractors of a given BN G.*

Proof. Algorithm 2 divides a BN into SCC blocks and detects attractors of each block. Lines 4 to 36 describe the process for detecting attractors of each block. Assuming this process is correct, based on Theorems 2 and 3, the merging operations of all the terminal blocks (lines 37 to 39) can actually restore the attractors of the BN. Now we only need to prove the above assumption is correct.

The algorithm distinguishes between two different types of blocks. The first type is an elementary block and the algorithm takes care of this type with lines 5 to 8. Since it is in fact a BN, the attractors of this type of block are directly detected via the basic attractor detection function $\textsc{Detect}(\mathcal{T})$. Therefore, lines 5 to 8 correctly compute the attractors of an elementary block. The second type is a non-elementary block and



(a) The realisation of B_3. (b) The realisation of B_4.

Figure 5: Two realisations used in Example 11.

the algorithm takes care of this type with lines 9 to 35. Based on Definition 17, the algorithm constructs the realisations of this type of block (lines 10 to 28), detects attractors of each realisation (line 27), and adds them to the set of attractors of the block (line 27). The correctness of line 27 is guaranteed by the general attractor detection approach and Definition 17. We now focus on lines 10 to 24. The algorithm takes special care of blocks with more than one parent block. It merges all the parent blocks of such a block to form a single parent block. Since the parent blocks are considered in an ascending order with respect to their depths (comment in line 14), the two operations in line 18 will iteratively restore the attractors of the parent block in accordance with Corollary 1. □

We have given an example to demonstrate an overview of the attractor detection approach in Example 3 (the beginning of Section 3). We can now explain this example in more details following Algorithm 2.

Example 11 (Attractor detection)**.** *Recall the BN G_2 we have demonstrated in Example 3. It is divided into four SCCs Σ_i, where $i \in [1,4]$. The four SCCs form four blocks B_i, where $i \in [1,4]$. Since the blocks B_1 and B_2 in G_2 are the same as that in G_1, we do not distinguish them in the rest of the paper. Block B_1 is an elementary block and it has one attractor as can be seen from its transition graph shown in Figure 3b. Obtaining the attractors of B_1 corresponds to Lines 5-8 in Algorithm 2. To detect the attractors of block B_2, we first form realisations of B_2 with respect to the attractors of its parent block B_1. This has been shown in Example 9. The transition graph of* *this realisation is shown in Figure 4a. It is clear from the graph that this realisation has one attractor, i.e.* $\mathcal{A}_2 = \{(000), (101), (111)\}$*. Since block* B_2 *has only one realisation, the attractor* $\mathcal{A}_2$ *is also the attractor of block* B_2*. This part corresponds to Lines 9-12, and 25-27 in Algorithm 2.* B_3 *has two parent blocks. Therefore, we need to merge the two parent blocks to form a single parent block. Since the attractors of the merged block* $B_{1,2}$ *are the same as that of* B_2*, we directly obtain the attractors of* $B_{1,2}$*, i.e.* $\mathcal{A}_{1,2} = \mathcal{A}_2$*. We form one realisation of block* B_3 *with respect to* $\mathcal{A}_{1,2}$ *and its transition graph is shown in Figure 5a. Clearly* B_3 *contains two attractors, i.e.* $\mathcal{A}_3 = \{\{(10101), (11111), (00011)\}, \{(00000), (10100), (11100)\}\}$*. Detecting the attractors of* B_3 *corresponds to Lines 13-27 in Algorithm 2. Similarly to block* B_2*, we can obtain the realisation of block* B_4 *(transition graph shown in Figure 5b), and attractors of block* B_4*, i.e.,* $\mathcal{A}_4 = \{(0000), (1010), (1110)\}$*. Lastly, we follow Lines 37-39 in Algorithm 2 and recover the attractors of the original BN as* $\mathcal{A} = D(\Pi(\mathcal{A}_3, \mathcal{A}_4)) = \{\{(000000), (101000), (111000)\}, \{(000110), (111110), (101010)\}\}$.

4.1. An optimisation

It often happens that a BN contains many *leaf* nodes that do not have any child node. Each of the leaf nodes will be treated as an SCC in our algorithm and it is not worth the effort to process an SCC consisting of only a single leaf node. Therefore, we treat leaf nodes in a special way. Formally, leaf nodes are recursively defined as follows.

Definition 19. *A node in a BN is a* leaf node *(or* leaf *for short) if and only if it is not the only node in the BN and either (1) it has no child nodes except for itself or (2) it has no other children after iteratively removing all its child nodes which are leaf nodes.*

Algorithm 3 outlines the leaf-based optimisation for attractor detection.

We now show that Algorithm 3 can identify all attractor states of a given BN.

Theorem 5. *Algorithm 3 correctly identifies all the attractor states of a given BN* G*.*

Proof. Block B formed in Line 2 is an elementary block. Algorithm 3 finds the attractor states of B, denoted $\mathcal{A}^B$ in Line 3. Since B is an elementary block, it preserves the

Algorithm 3 Leaf-based optimisation

```
1: procedure LEAF_DETECT($G$)
2:     form a BN $B$ by removing all the leaf nodes of $G$;
3:     $\mathcal{A}^B :=$ SCC_DETECT ($B$);
4:     $\mathcal{T} :=$ transition system of $G$ with state space restricted to $\mathcal{M}_G(\cup_{A^B \in \mathcal{A}^B} A^B)$;
5:     $\mathcal{A} :=$ DETECT ($\mathcal{T}$);
6:     return $\mathcal{A}$.
7: end procedure
```

attractors of G by Theorem 1 and thus, by Lemma 1, it holds that $\mathcal{M}_G(\Phi^B)$ contains all the attractor states of G. Therefore, the basic attractor detection function DETECT applied in Line 5 to the transition system of G restricted to the states $\mathcal{M}_G(\Phi^B)$ identifies all the attractor states of G. □

5. Evaluation

We use BDD-based techniques to implement our algorithm. Existing BDD-based techniques have already been widely applied to model checking. Therefore, we implement our method based on an existing model checker. In particular, we have implemented the decomposition algorithm presented in Section 4 in the model checker MCMAS [24] and we have adapted the SCC detection function in this model checker for serving as the function DETECT in the algorithm. In this section, we demonstrate the efficiency of our method by comparing our method with the state-of-the-art decomposition method presented in [12] which is also based on BDD implementation. The demonstration is performed on both randomly generated networks and real-life biological networks. All the experiments are conducted on a high-performance computing (HPC) node, which contains an Intel(R) Xeon(R) CPU L5640 @ 2.26GHz. [2]

[2]The experiment environment mentioned in the conference paper [19] is incorrect. We correct it here.

5.1. Evaluation on Randomly Generated Networks

We generate 33 random BN models with different number of nodes using the tool ASSA-PBN [25, 26] and compare the performance of the two methods on these 33 models. The 33 BNs are randomly generated with different size (number of nodes) and different connectivity (the average number of parents of a Boolean function) to reflect different networks. The size ranges from 100 to 500 due to that 500 is almost the limit of our method. In total, there are 11 different sizes. For each size, we generate 3 different networks with different connectivity. In the end, there are 33 networks generated.

We denote our proposed decomposition method as M_1 and the one in [12] as M_{ref}. There are two possible implementations of the DETECT function used in Algorithm 2 as mentioned in [12]: monolithic and enumerative. We use the monolithic one which is shown to be more suitable for small networks as the decomposed sub-networks are relatively small. Since the method in [12] uses similar leaf reduction technique to the one presented in this study, we make comparisons on both the original models and the models obtained from the original ones by removing leaves in order to eliminate the influence of leaf nodes. We set the expiration time to 3 hours. Before removing leaf nodes, there are 11 cases where both methods fail to return a result within the 3-hour time limit. Among the other 22 cases, our method is faster than M_{ref} in 16 cases, i.e. in approximately 73$\%$ of the cases. After removing leaf nodes, there are 5 cases in which both methods fail to accomplish the computations. Among the other 28 cases, our method is faster than M_{ref} in 25 cases, i.e. in approximately 89$\%$ of the cases. We demonstrate the results in Table 1. Since our method considers the dependency relation between different blocks, the attractors of all the blocks need to be computed; while method M_{ref} can ignore the blocks with only leaf nodes. Therefore, the performance of our method is affected by the presence of leaf nodes to a larger extent than M_{ref}. This is why the percentage of cases in which our method is faster than M_{ref} is increased from 73% to 89$\%$ when leaf nodes are removed. Notably, after eliminating the influence of leaf nodes, our method is significantly faster than M_{ref}. The "–" in Table 1 means the method failed to process the model within 3 hours. The speedup is therefore not available (N/A) for the respective case. The speedup is computed as $t_{M_{ref}}/t_{M_{our}}$, where $t_{M_{our}}$ is the time cost for M_{our} and $t_{M_{ref}}$ is the time cost for M_{ref}. All the times shown in Table 1 are in seconds.

Several reasons may affect the performance of the two methods. For example, the number of attractors, the base attractor detection method DETECT, the BDD ordering. In our implementation, we have fixed the BDD variable ordering with the first ordering strategy used in [24] for the attractor detection method. In general, choosing a good variable ordering is known to be a hard problem [27]. Our choice is based on our experimental evaluation. However, the fixed ordering may not be the best for every network. For example, in our experiment on the network with 400 nodes, our algorithm takes 13.64 seconds to compute the attractors with the monolithic implementation of the DETECT function. The time will be reduced to 2.52 if we change the DETECT function to enumerative implementation. On the contrary, the time cost of the method M_{ref} will be increased from 8.28 to 19.68 if the same change is made. Due to these considerations, it is very difficult to find clear criteria which allow to judge which method is definitely faster for a given network. However, it is justified to conclude that our method is faster than M_{ref} in most cases and where the number of attractors is relatively small, the chances that our method is faster are even higher. This is due to the fact that our method takes the attractors of the parent block into account when forming a realisation of a non-elementary block and the number of realisations increases with the number of attractors.

In addition, we plot the speedups of method M_{our} with respect to (w.r.t.) method M_{ref} in Figure 6. Note that we manually set up a maximum value, i.e., 3×10^4 as highlighted by the dashed line in the upper part of the figure, and a minimum value, i.e., 3×10^{-3} as highlighted by the dashed line in the lower part of the figure, for the cases that only one of the two methods finishes the computation within 3 hours. According to the plot, it is clear that after removing leaves, our new method has more chances to perform faster than the method M_{ref} as there are more red diamonds between the two upper dashed lines comparing to the blue circles. Summarising, for an important proportion of random models, our new method shows a significant improvement on the state-of-the-art decomposition method.

# nodes	# non-leaves	# attractors	original models			models with leaves removed		
			$t_{M_{ref}}[s]$	$t_{M_{our}}[s]$	S.	$t_{M_{ref}}[s]$	$t_{M_{our}}[s]$	speedup
100	17	12	3.74	1.34	2.8	0.72	0.14	5.1
100	7	32	4.56	0.86	5.3	0.58	0.02	29.0
100	34	20	8.64	6.37	1.4	1.24	0.16	7.8
120	17	28	16.32	12.48	1.3	0.94	0.04	23.5
120	9	1	18.13	0.95	19.1	1.10	0.04	27.5
120	11	128	12.61	13.09	1.0	1.16	0.16	7.3
150	17	1266	517.8	13618.00	0.0	18.17	7.09	2.6
150	12	208	151.79	960.81	0.2	2.32	0.27	8.6
150	19	2	201.22	1.66	121.2	0.74	0.02	37.0
180	15	16	11.9	4.40	2.7	0.85	0.03	28.3
180	29	N/A	–	–	N/A	–	–	N/A
180	42	128	609.41	482.72	1.3	4.2	4.54	0.9
200	6	16	268.69	7.04	38.2	0.97	0.02	48.5
200	10	120	407.12	655.80	0.6	1.02	0.08	12.8
200	15	640	1493.42	231.06	6.5	3.59	1.96	1.8
250	25	12	533.57	11.16	47.8	0.90	0.04	22.5
250	19	4320	10631.83	–	N/A	33.88	24.79	1.4
250	35	4976	–	28434.00	N/A	100.78	43.35	2.3
300	21	192	–	5113.15	N/A	1.56	0.70	2.2
300	88	1	–	–	N/A	238.96	65.33	3.7
300	32	20	2204.57	37.86	58.2	1	0.19	5.3
350	68	3440	–	–	N/A	-1	8572.85	0.0
350	46	3904	–	–	N/A	75.14	28.84	2.6
350	21	188	–	10520.40	N/A	3.09	0.41	7.5
400	65	N/A	–	–	N/A	–	–	N/A
400	52	3	342.54	20.68	16.6	8.28	13.64	0.6
400	64	N/A	–	–	N/A	–	–	N/A
450	20	18304	–	–	N/A	154.98	132.08	1.2
450	83	N/A	–	–	N/A	–	–	N/A
450	43	8	–	60.82	N/A	3704.33	0.17	21790.2
500	62	612	–	–	N/A	26.43	23.16	1.1
500	81	N/A	–	–	N/A	–	–	N/A
500	48	1728	–	–	N/A	–	38.14	N/A

Table 1: Results of the performance comparison of methods M_{our} and M_{ref}.

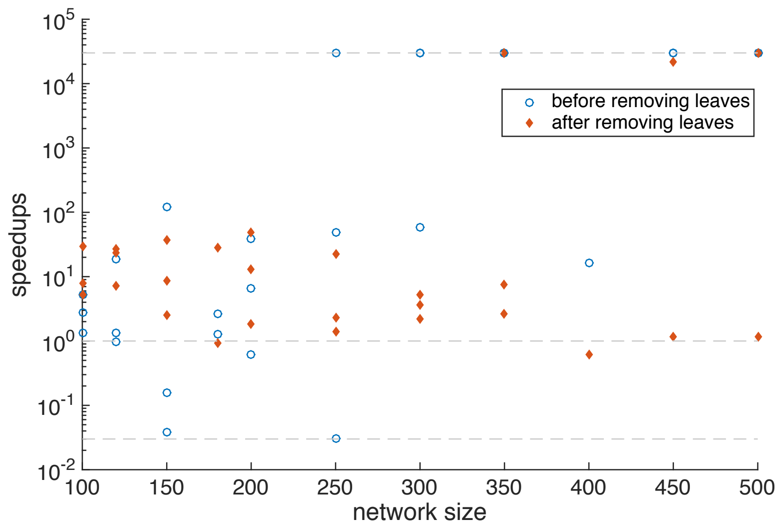


Figure 6: Speedups of Method M_{our} w.r.t. Method M_{ref}.

5.2. Evaluation on Real-life Networks

We continue to evaluate the performance of our proposed decomposition method M_{our} with respect to M_{ref} on 9 different models from 6 real-life biological networks which we obtain from the literature. We give a brief introduction of the 6 networks below.

- **Tumour.** Tumour is a model constructed in [18] for studying the role of individual mutations or their combinations affecting the metastatic development. Metastasis accounts for 90% of cancer patient mortality; therefore, understanding the etiology of metastasis is very important in clinical perspective. The early stages of metastasis are tightly controlled in normal cells and can be drastically affected by malignant mutations; therefore, they might constitute the principal determinants of the overall metastatic rate even if the later stages take long to occur.
- **MAPK network.** Mitogen-activated protein kinases (MAPKs) are a family of serine/threonine kinases that transduce biochemical signals from the cell membrane to the nucleus in response to a wide range of stimuli, such as growth

model name	reference	# nodes	# attractors	$t_{M_{ref}}[s]$	$t_{M_{our}}[s]$	speedup
tumour	[18]	32	15	1.48	0.28	5.34
MAPK_r3	[28]	53	60	2.85	0.98	2.91
MAPK_r4	[28]	53	132	6.37	2.68	2.37
T-LGL	[29]	60	1322	214.04	45.08	4.75
HGF	[16]	66	2	0.39	0.05	7.50
T-diff	[30]	68	18	1.15	0.60	1.91
apoptosis_1	[31]	97	152	13.86	11.42	1.21
apoptosis_2	[31]	97	16	7.91	1.17	6.77
apoptosis_3	[31]	97	1	7.93	0.33	24.24

Table 2: Evaluation on real-life models.

factors, hormones, inflammatory cytokines, and environmental stresses. Cascades of these kinases participate in multiple intracellular signalling pathways that control a wide range of cellular processes, e.g. cell cycle machinery, differentiation, survival, and apoptosis. MAPK pathways are highly evolutionary conserved among all eukaryotic cells and allow the cells to respond coordinately to multiple and diverse inputs. To date, three main pathways have been extensively studied: Extracellular Regulated Kinases (ERK), Jun NH_2 Terminal Kinases (JNK), and p38 Kinases (p38), named after their specific MAPK kinases involved. These pathways are characterised by enormous cross-talks with each other, which gives rise to a complex network of molecular interactions [32]. Malfunctioning of MAPK signalling mechanisms is often observed in cancer [33]. Therefore, a deeper comprehension of the MAPK pathways and their interactions is of utter importance to elucidate the roles of MAPKs in the development and progression of cancer. This in turn is crucial for the development of new and effective therapeutic strategies. In [28], a predictive dynamical Boolean model of the MAPK network is presented. It recapitulates observed responses of the MAPK network to characteristic stimuli in selected urinary bladder cancers together with its specific contribution to cell fate decision on proliferation, apoptosis, and growth arrest. In our study we consider two mutants of the model: one with EGFR over-expression and the other with FGFR3 activating mutation which correspond to the r3 and r4 variants of [28], respectively, and therefore we refer to them as as MAPK_r3 and MAPK_r4.

- **T cell large granular lymphocyte leukemia network.** T cell large granular lymphocyte (T-LGL) leukemia is a clonal proliferation of cytotoxic T lymphocytes (CTL). In T-LGL leukemia, cytotoxic T cells avoid cell death and survive, which can cause diseases such as neutropenia, anemia, and autoimmune disorders. Zhang et al. constructed a T-LGL model in [29] in order to systematically understand signalling components that determine the survival of CTL in T-LGL leukemia. We take this model with 60 nodes and perform our evaluation.
- **HGF.** Hepatocyte growth factor (HGF) is one of the factors that are known to activate and regulate cell migration. Cell migration plays an important role in tissue homeostasis. In case the cell migration is in an abnormal condition, it can lead to scar formation and facilitate cancer metastasis formation. In [16], a large-scale dynamic network describing HGF-induced keratinocyte migration was developed based on prior knowledge. We take the fifth network named HGF from [16] for our evaluation.
- **T-diff.** Helper T cells is a type of T cell that plays a central role in the regulation of the immune response in mammals. They are essential in the activation of B cells and cytotoxic T cells to protect the body and kill infected cells. In [30], Naldi et al. propose an integrated, comprehensive model of the regulatory network and signalling pathways controlling T cell differentiation. The T-diff model is based on this work.
- **Apoptosis network.** Apoptosis is a process of programmed cell death and has been linked to many diseases. It is often regulated by several signalling pathways extensively linked by cross-talks. We take the apoptosis signalling network presented in [31] and recast it into the Boolean network framework: a BN model which compromise 97 nodes. In this network, there are 10 input nodes. We obtained three different variants of this network based on different sets of inputs and named the three variants as apoptosis_1, apoptosis_2, and apoptosis_3.

The evaluation results are shown in Table 2. In all the 9 networks, our proposed decomposition method M_{our} is faster than M_{ref}. Due to the fact that our decomposition method considers the attractors of the parent block when forming a realisation of a non-elementary block, our decomposition method takes more time when the number of attractors increases. Hence, our method results in larger speedups when the number of attractors is smaller. This can be clearly observed from the last three apoptosis models. This observation is important as most of the meaningful real-life biological systems should have a relatively small number of attractors [34, 35]. In addition, the model T-LGL contains a large number of attractors; however, our method still gains a relatively large speedup of 4.75 compared to the method M_{ref}. This is due to the fact that most of the attractors are contributed by only one block and this block has only three leaf children. Hence, it does not take too much time to take into consideration the dependency relationships between blocks in this case.

In addition to the above evaluation, we future applied the leaf-based optimisation to the network named "apoptosis_1". In this network, 17 out of 97 nodes are leaves. It takes 3.26 seconds to get the results of 152 attractors, which is 3.5 times faster than the original 11.42 seconds. We chose this network to apply our leaf-based optimisation due to the following three reasons. Firstly, the percentage of leaf nodes is high enough (bigger than 10%). Secondly, the time cost of the original method is relatively large (bigger than 5). Lastly, the number of attractors is relatively small (smaller than 500).

The evaluation results clearly show that our proposed decomposition method is faster than the one in [12], for a significant proportion of models and, in particular, all the real-life models that we have considered. Our method can lead to a large speedup when the number of attractors in a network is relatively small, which is often the case for real-life biological networks.

6. Conclusion and Future Work

We have introduced a new SCC-based decomposition method for attractor detection of large synchronous BNs. Although our decomposition method shares similar ideas on how to decompose a large network with existing decomposition methods, our method differs from them in the key process and has significant advantages.

First, our method is designed for synchronous BNs. As a consequence, the key process for constructing realisations in our method is substantially different from the one in [15], which is designed for asynchronous networks. Secondly, our method considers the dependency relation among the sub-networks. The method in [12] does not rely on this relation and only takes the attractors of sub-networks to restrict the initial states when detecting the attractors for the original network. In this way, the decomposition method in [12] potentially cannot scale up very well for large systems, as it still requires a BDD encoding of the transition relation of the whole network. This is our main motivation to extend our previous work [15] towards synchronous BNs. Experimental results show that our method is significantly faster than the one in [12]. Lastly, we have shown that the method proposed in [11] fails to compute correct results in certain cases.

The most time consuming part of our algorithm is the DETECT function. Our current implementation is based on BDDs. The performance is strongly related to the performance of BDD operations. We choose a BDD-based technique mainly due to the following two considerations. Firstly, it is fair to compare our method with the method in [12] as they are both using BDD-based techniques. Secondly, BDD-based technique is one of the leading techniques that are suitable for the computation of SCCs, which can be easily applied to the detection attractors. One future work is to use SAT-solvers to implement the DETECT function as SAT-based methods are normally more efficient for attractor detection of synchronous BNs [9].

Acknowledgments

Qixia Yuan is supported by the National Research Fund, Luxembourg (grant reference 7814267). This work is also partially supported by the research project SEC-PBN (funded by the University of Luxembourg) and the ANR-FNR project AlgoReCell (INTER/ANR/15/11191283).

References

[1] Z. Dong, Y. Pan, X. Huang, Parameter identifiability of Boolean networks with application to fault diagnosis of nuclear plants, Nuclear Engineering and Technology 50 (4) (2018) 599605.

[2] A. Roli, M. Villani, R. Serra, S. Benedettini, C. Pinciroli, M. Birattari, Dynamical properties of artificially evolved Boolean network robots, in: Proc. 14th International Conference of the Italian Association for Artificial Intelligence, Vol. 9336 of LNCS, Springer, 2015, pp. 45–57.

[3] S. Kauffman, Homeostasis and differentiation in random genetic control networks, Nature 224 (1969) 177–178.

[4] S. Huang, Genomics, complexity and drug discovery: insights from Boolean network models of cellular regulation, Pharmacogenomics 2 (3) (2001) 203–222.

[5] R. Somogyi, L. D. Greller, The dynamics of molecular networks: applications to therapeutic discovery, Drug Discovery Today 6 (24) (2001) 1267–1277.

[6] D. J. Irons, Improving the efficiency of attractor cycle identification in Boolean networks, Physica D: Nonlinear Phenomena 217 (1) (2006) 7–21.

[7] A. Garg, L. Xenarios, L. Mendoza, G. DeMicheli, An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments, in: Proc. 11th Annual Conference on Research in Computational Molecular Biology, Vol. 4453 of LNCS, Springer, 2007, pp. 62–76.

[8] A. Garg, A. Di Cara, I. Xenarios, L. Mendoza, G. De Micheli, Synchronous versus asynchronous modeling of gene regulatory networks, Bioinformatics 24 (17) (2008) 1917–1925.

[9] E. Dubrova, M. Teslenko, A SAT-based algorithm for finding attractors in synchronous Boolean networks, IEEE/ACM Transactions on Computational Biology and Bioinformatics 8 (5) (2011) 1393–1399.

[10] Q.-N. Tran, Algebraic model checking for Boolean gene regulatory networks, in: Software Tools and Algorithms for Biological Systems, Springer, 2011, pp. 113–122.

[11] W. Guo, G. Yang, W. Wu, L. He, M. Sun, A parallel attractor finding algorithm based on Boolean satisfiability for genetic regulatory networks, PLOS ONE 9 (4) (2014) e94258.

[12] Q. Yuan, H. Qu, , J. Pang, A. Mizera, Improving BDD-based attractor detection for synchronous Boolean networks, Science China Information Sciences 59 (8) (2016) 080101.

[13] Y. Zhao, J. Kim, M. Filippone, Aggregation algorithm towards large-scale Boolean network analysis, IEEE Transactions on Automatic Control 58 (8) (2013) 1976–1985.

[14] A. Le Coënt, L. Fribourg, R. Soulat, Compositional analysis of Boolean networks using local fixed-point iterations, in: International Workshop on Reachability Problems, Springer, 2016, pp. 134–147.

[15] A. Mizera, J. Pang, H. Qu, Q. Yuan, Taming asynchrony for attractor detection in large Boolean networks, IEEE/ACM Transactions on Computational Biology and Bioinformatics 15 (2).

[16] A. Singh, J. M. Nascimento, S. Kowar, H. Busch, M. Boerries, Boolean approach to signalling pathway modelling in HGF-induced keratinocyte migration, Bioinformatics 28 (18) (2012) i495–i501.

[17] M. I. Davidich, S. Bornholdt, Boolean network model predicts cell cycle sequence of fission yeast, PLOS One 3 (2) (2008) e1672.

[18] D. P. Cohen, L. Martignetti, S. Robine, E. Barillot, A. Zinovyev, L. Calzone, Mathematical modelling of molecular pathways enabling tumour cell invasion and migration, PLOS Computational Biology 11 (11) (2015) e1004571.

[19] A. Mizera, J. Pang, H. Qu, Q. Yuan, A new decomposition method for attractor detection in large synchronous Boolean networks, in: Proc. 3rd International Symposium on Dependable Software Engineering: Theories, Tools, and Applications, Vol. 10606 of LNCS, Springer, 2017, pp. 232–249.

[20] S. A. Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets, Journal of Theoretical Biology 22 (3) (1969) 437–467.

[21] I. Shmulevich, E. R. Dougherty, Probabilistic Boolean Networks: The Modeling and Control of Gene Regulatory Networks, SIAM Press, 2010.

[22] C.-Y. Lee, Representation of switching circuits by binary-decision programs, Bell System Technical Journal 38 (4) (1959) 985–999.

[23] S. B. Akers, Binary decision diagrams, IEEE Transactions on Computers 100 (6) (1978) 509–516.

[24] A. Lomuscio, H. Qu, F. Raimondi, MCMAS: An open-source model checker for the verification of multi-agent systems, International Journal on Software Tools for Technology Transfer 19 (1) (2017) 9–30.

[25] A. Mizera, J. Pang, Q. Yuan, ASSA-PBN 2.0: A software tool for probabilistic Boolean networks, in: Proc. 14th International Conference on Computational Methods in Systems Biology, Vol. 9859 of LNCS, Springer, 2016, pp. 309–315.

[26] A. Mizera, J. Pang, C. Su, Q. Yuan, ASSA-PBN: A toolbox for probabilistic Boolean networks, IEEE/ACM Transactions on Computational Biology and Bioinformatics 15 (4) (2018) 1203–1216.

[27] B. Bollig, I. Wegener, Improving the variable ordering of OBDDs is NP-complete, IEEE Transactions on Computers 45 (9) (1996) 993–1002.

[28] L. Grieco, L. Calzone, I. Bernard-Pierrot, F. Radvanyi, B. Kahn-Perles, D. Thieffry, Integrative modelling of the influence of MAPK network on cancer cell fate decision, PLOS Computational Biology 9 (10) (2013) e1003286.

[29] R. Zhang, M. V. Shah, J. Yang, S. B. Nyland, X. Liu, J. K. Yun, R. Albert, T. P. Loughran, Network model of survival signaling in large granular lymphocyte leukemia, Proceedings of the National Academy of Sciences 105 (42) (2008) 16308–16313.

[30] A. Naldi, J. Carneiro, C. Chaouiya, D. Thieffry, Diversity and plasticity of th cell types predicted from regulatory network modelling, PLoS Computational Biology 6 (9) (2010) e1000912.

[31] R. Schlatter, K. Schmich, I. A. Vizcarra, P. Scheurich, T. Sauter, C. Borner, M. Ederer, I. Merfort, O. Sawodny, ON/OFF and beyond - a Boolean model of apoptosis, PLOS Computational Biology 5 (12) (2009) e1000595.

[32] M. Krishna, H. Narang, The complexity of mitogen-activated protein kinases (MAPKs) made simple, Cellular and Molecular Life Sciences 65 (22) (2008) 3525–3544.

[33] A. S. Dhillon, S. Hagan, O. Rath, W. Kolch, MAP kinase signalling pathways in cancer, Oncogene 26 (2007) 3279–3290.

[34] B. Samuelsson, C. Troein, Superpolynomial growth in the number of attractors in Kauffman networks, Physical Review Letters 90 (9) (2003) 098701.

[35] B. Dorssel, Number of attractors in random Boolean networks, Physical Review Letters 72 (2005) 016110.