Taming Asynchrony for Attractor Detection in Large Boolean Networks

Andrzej Mizera, Jun Pang, Hongyang Qu, and Qixia Yuan

Abstract—Boolean networks is a well-established formalism for modelling biological systems. A vital challenge for analysing a Boolean network is to identify all the attractors. This becomes more challenging for large asynchronous Boolean networks, due to the asynchronous updating scheme. Existing methods are prohibited due to the well-known state-space explosion problem in large Boolean networks. In this paper, we tackle this challenge by proposing a SCC-based decomposition method. We prove the correctness of our proposed method and demonstrate its efficiency with two real-life biological networks.

Index Terms—Boolean networks, gene regulatory networks, strongly connected components, binary decision diagram, attractor detecttion.

1 Introduction

Boolean networks (BNs) is a well-established framework used for modelling biological systems such as gene regulatory networks (GRNs). It has the advantage of being simple yet able to capture the important dynamic properties of the modelled system [1], e.g., the system's *attractors*. An attractor of a biological system is a set of the system's states satisfying that any two states in this set can be reached from each other and the system remains in this set until some external stimulus pushes the system out of it. Attractors are hypothesised to characterise cellular phenotypes [2] or to correspond to functional cellular states such as proliferation, apoptosis, or differentiation [3]. For example, in the recent study of Sanchez-Corrales et al. [4], attractors of an *Arabidopsis thaliana* system correspond to stable gene expression levels during the different stages of flower development. These interpretations can cast new light on the understanding of cellular homeostasis and cancer progression. Identification of attractors is therefore of great importance for the analysis of biological systems modelled as BNs.

Attractor detection of a BN is non-trivial since attractors are determined based on the BN's states, the number of which is exponential in the number of nodes. A lot of efforts have been put in the development of attractor detection algorithms and tools. In the early 2000s, an enumeration and simulation method has been proposed. The idea is to enumerate all the possible states and to run simulation from each of them until an attractor is found [5]. This method is largely restricted by the network size since the time grows exponentially with the number of nodes. In 2006, Irons proposed a method to detect BNs with a special topology [6], making it possible to deal with BNs with maximum 50 nodes. Later on, the performance has been greatly improved with two techniques, i.e., binary decision diagrams (BDDs) and satisfiability (SAT) solvers. BDD-based methods [7], [8] encode Boolean functions of BNs with BDDs, use BDD operations to capture the dynamics of the network, and use BDD structure to represent the network's corresponding transition system. Using the BDD operations, the forward and backward reachable states can be often efficiently computed. Detecting attractors is then reduced to finding fix point set of states in the corresponding transition system. The other technique 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 process is repeated iteratively for larger bounded numbers of steps until all attractors are identified. The efficiency of the algorithm largely relies on the number of unfolding steps required and the number of nodes in the BN. Recently, a few decomposition methods [10], [11], [12] were proposed to deal with large BNs. The main idea is to decompose a large BN into small components based on its structure, detect attractors in the small components, and then recover the attractors of the original BN. In addition, there are some approximation methods [13], [14], reduction methods [15], [16], and methods designed for identifying specific types of attractors [17], [18]. These methods can deal with large networks; however, they cannot guarantee the identification of all attractors correctly.

The above mentioned methods are mainly designed for BNs with the *synchronous* updating scheme, i.e., BNs where the values of all the nodes are updated simultaneously. In biology, however, the update speed of each node is not necessarily the same. Updating nodes values *asynchronously* is considered more realistic [19]. In synchronous BNs, an attractor is either a single state selfloop or a cycle since there is exactly one outgoing transition for each state. Under the asynchronous updating scheme, each state may have multiple outgoing transitions. Therefore, an attractor in general

- *A. Mizera is with the Allergology - Immunology - Inflammation Research Unit, Department of Infection and Immunity, Luxembourg Institute of Health. J. Pang and Q. Yuan are with the Computer Science and Communications Research Unit, University of Luxembourg. H. Qu is with the Department of Automatic Control & Systems Engineering, University of Sheffield.*
E-mail: andrzej.mizera@lih.lu, jun.pang@uni.lu, h.qu@sheffield.ac.uk, qixia.yuan@uni.lu

is a bottom strongly connected component (BSCC)[1] in the corresponding state transition system. The potentially complex attractor structure renders SAT-based methods ineffective as the respective SAT formulas become prohibitively large. Besides, the decomposition methods [10], [11], [12] are also prohibited by the asynchronous updating requirement. Moreover, BDD-based methods face the state-space explosion problem even in the synchronous updating scheme. In the asynchronous updating scheme, the problem gets even worse as the number of edges in the state transition system increases multiple times.

In this paper, we tackle the challenge of attractor detection for asynchronous BNs, especially for large ones, and we propose a strongly connected component (SCC) based decomposition method: decompose a BN into subnetworks called *blocks* according to the SCCs in the BN and recover attractors of the original BN based on attractors of the blocks. Since the *decomposition is performed on the BN structure, not in the state space*, the decomposition time cost is linear in the number of nodes and the state space of each block is exponentially smaller in comparison to that of the original BN. Our method shares similar ideas on the way of decomposition as those used for synchronous BNs. However, due to the asynchronous updating scheme, the *bottom-up* decomposition methods [10], [11] for synchronous BNs are no longer valid, as they may produce spurious attractors. The asynchrony poses two main challenges for the decomposition methods: one is to take care of the dependency relations between different blocks; the other is to strictly comply with the asynchronous updating scheme when recovering attractors from different blocks. To overcome these difficulties, we order the blocks according to their dependency relations and detect attractors of each block with consideration of the block that it depends on. In this way, our method is *top-down*, starting with elementary blocks which do not depend on others. We prove that our proposed method can correctly detect all the attractors of a BN (Section 3), and we implement it using efficient BDD techniques (Section 4). Evaluation results show that our method can effectively detect attractors of two real-life biological networks (Section 5).

2 Preliminaries

2.1 Boolean networks

A BN describes elements of a biological system with binary-valued nodes and interactions between elements with Boolean functions. It was first introduced by Kauffman in 1969 as a class of simple models for analysing the dynamical properties of GRNs [21], in which each gene was assumed to be in only 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 $f_i : \{x_{i_1}, x_{i_2}, \ldots, x_{i_{k(i)}}\} \rightarrow \{0, 1\}$ is a predictor function associated with node v_i ($i = 1, 2, \ldots, n$) and $x_{i_j} \in \{0, 1\}$ for $j \in [1, k(i)]$ is a value assigned to node v_{i_j}. 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.

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 $1 \le i_1 < i_2 < \cdots < i_{k(i)} \le n$. 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 is updated at a time. The synchronous updating scheme is used mostly due to its simplicity; however, for the modelling of GRNs, the asynchronous scheme is more suitable as the expression of a gene is usually not an instantaneous process.

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

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

It states that node v_i is updated by its Boolean function and other nodes are kept unchanged. Each node has a chance to be updated by its Boolean function, therefore there are n outgoing transitions in maximum from any state.

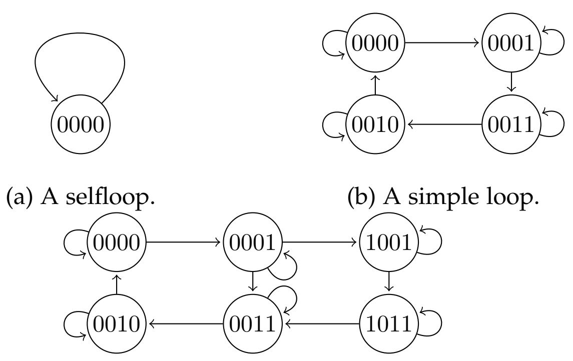
Many key characters of a BN, e.g., attractors, are often examined in the level of its state transition system (STS). [2] Formally, the state transition system and attractors of a BN are defined as follows.

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, specifying the evolvement of the system. When $S = S_0$, we write $\langle S, T\rangle$.

An asynchronous 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 states set S_0 is the same as S since usually all states are accessible in a biological system modelled as a BN; finally, the transition relation T is given by Equation 1.

1. It is also referred as *loose attractor* in the literature [20].

2. For presentation purpose, we draw a few state transition systems as graphs in the remaining part of the paper and also refer them as transition graphs.



(a) A selfloop. (b) A simple loop.

(c) A complex loop.

Fig. 1: Three types of attractors of an asynchronous BN.

Definition 3 (Attractor of a BN). An *attractor* of a BN is a set of states satisfying 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.

The attractors of a BN characterise its long-run behaviour [22] and are of particular interest due to their biological interpretation as, for instance, attractors are hypothesised to characterise cellular phenotypes [2]. When analysing an attractor, we often need to identify transition relations between the attractor states. We refer to an attractor together with its state transition relations as an *attractor system*. The states constituting an attractor are called *attractor states*. In the synchronous updating scheme, each state can only have one outgoing transition. Therefore, the attractor system in a synchronous BN is simply a loop. By detecting all the loops in a synchronous BN, one can identify all its attractors. However, it becomes much more complicated with the asynchronous updating scheme. The attractor system does not necessarily need to be a loop and may have a more intricate topology. In fact, it may include several loops. We list three general types of attractors of an asynchronous BN: a singleton attractor, i.e., a *selfloop*, as shown in Figure 1a; a *simple loop*, as shown in Figure 1b; and a *complex loop*, as shown in Figure 1c. The selfloops and simple loops also exist in the corresponding synchronous BN. Therefore, one can identify the selfloops and simple loop attractors of an asynchronous BN by detecting the attractors of its corresponding synchronous BN.[3] However, this is not the case for complex loop attractors. Special algorithms need to be designed to detect such attractors under asynchronous updating scheme.

2.2 Encoding BNs in BDDs

Binary decision diagrams (BDDs) were introduced to represent Boolean functions [23], [24]. A BDD consists of three types of nodes: a root node, (intermediate) decision nodes, and two terminal nodes, i.e., 0-terminal and 1-terminal. It uses a decision node to represent a variable of a Boolean function. Each decision node contains two outgoing edges, representing the two possible values, i.e., 0 and 1, of the variable. A path from the root node to the 1-terminal represents an assignment of values to the variables that results in the true value of the Boolean function; while a path from the root node to the 0-terminal represents an assignment of values to the variables that results in the false value of the Boolean function. BDDs have the advantage of memory efficiency and have been applied in model checking algorithms to alleviate the state space explosion problem. A BN $G(V, \boldsymbol{f})$ can be easily encoded in a BDD by modelling a BN as an STS. Each variable in V can be represented by a binary BDD variable. By slight abuse of notation, we 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 T can be viewed as a Boolean function $T^f : 2^{|V|+|V'|} \rightarrow \{0,1\}$, where values 1 and 0 indicate a valid and an invalid transition, respectively. Our attractor detection algorithms also use two basis 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. To simplify the presentation, we also define $\mathit{Preimage}^i(X,T) = \underbrace{\mathit{Preimage}(...(\mathit{Preimage}(X,T)))}_{i}$ with $\mathit{Preimage}^0(X,T) = X$. Thus, the set of all states that can reach a state in X via transitions in T is defined as a fix point $\mathit{Predecessors}(X,T) = \bigcup_{i=0}^{n} \mathit{Preimage}^n(X,T)$ such that $\mathit{Preimage}^n(X,T) = \mathit{Preimage}^{n+1}(X,T)$. Given a set of states $X \subseteq S$, the projection $T|_X$ of T on X is defined as $T|_X = \{(s,s') \in T \mid s \in X \wedge s' \in X\}$.

The BDD b representing a state $s = (x_1, x_2, \dots, x_n)$ can be seen as a Boolean formula $g(b) = (v_1 = x_1) \wedge (v_2 = x_2) \wedge \ldots \wedge (v_n = x_n)$. Let $g(b)^{-i} = (v_1 = x_1) \wedge \ldots \wedge (v_{i-1} = x_{i-1}) \wedge (v_{i+1} = x_{i+1}) \ldots \wedge (v_n = x_n)$ $(1 \leq i \leq n)$. The *existential abstraction* of v_i from b produces a new BDD $b|_{\{v_i\}}$ equivalent to the Boolean formula $g(b)^{-i} \wedge (v_i = x_i \vee v_i = \neg x_i)$. For our convenience, we say that node v_i is set to value "-" by existential abstraction and the new BDD can be written as $g(b)^{-i} \wedge (v_i = \text{"-"})$. The existential abstraction can be applied to a set $V' \subseteq V$ of nodes on a set of states $S' \subseteq S$, written as $S'|_{V'}$. The intersection of two BDDs b_1 and b_2, written as $b_1 \cap b_2$, is equivalent to the Boolean formula $g(b_1) \wedge g(b_2)$.

3 Method

In this section, we describe in details our SCC-based decomposition method for detecting attractors of large asynchronous BNs and prove its correctness. The method consists of three main steps. First, we divide a BN into subnetworks called *blocks*. This step is performed based on the BN network structure and therefore it can be executed efficiently. Second, we detect attractors of each block. This step is performed on the constructed STSs of the blocks. Notice that for each block the size of its STS is exponentially reduced with respect to the size of the STS of the original

3. Note that some of the detected attractors in the form of loops in a synchronous BN can be absent in the corresponding asynchronous BN. See [8], [20] for detailed discussions.

BN. Finally, we recover attractors of the original BN by merging the detected attractors of the blocks.

3.1 Decomposition of a BN into blocks

We start a detailed presentation of our approach by giving the formal definition of a block.

Definition 4 (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 $V^B \subseteq V$ and $\boldsymbol{f^B}$ is a list of Boolean functions for nodes in V^B: for any node $v_i \in V^B$, if B contains all the parent nodes of v_i, its Boolean function in B remains the same as in G, i.e., f_i; otherwise, the Boolean function is undetermined, meaning that additional information is required to determine the value of v_i in B. We call the nodes with undetermined Boolean functions as *undetermined nodes*. We refer to a block as an *elementary block* if it contains no undetermined nodes.

We consider asynchronous networks and therefore a block is also under the asynchronous updating scheme, i.e., only one node in the block can be updated at any given 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 5 (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 in C, there is a directed path from u to v and vice versa.

For a given BN, we decompose its network structure into SCCs. Figure 2 shows the decomposition of a BN into four SCCs: Σ_1, Σ_2, Σ_3, and Σ_4. 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 2, node v_1 is a control node of Σ_2 and Σ_4; node v_2 is a control node of Σ_3; and node v_6 is a control node of Σ_4. The SCC Σ_1 does not have any control node.

Definition 6 (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. 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 Σ_j's ancestor. We use $\Omega(\Sigma_j)$ to denote the set of ancestor SCCs of Σ_j.

An SCC together with its control nodes forms a *block*. For example, in Figure 2, Σ_2 and its control node v_1 form one block B_2. Σ_1 itself is a block, denoted as B_1, since the SCC it contains does not have any control node. If a control node in a block B_i is a determined node in another block B_j, block B_j is called a *parent* of block B_i and B_i is a child of B_j. In this way, the concepts of parent and ancestor are naturally extended to blocks.

By adding directed edges from all parent blocks to all their child blocks, we form a directed acyclic graph (DAG) of the blocks as the blocks are formed from SCCs. We notice here that in our decomposition approach, as long as the block graph is guaranteed to be a DAG, other strategies to form blocks can be used. Two blocks can be merged into one larger block. For example, blocks B_1 and B_2 can be merged to form a larger block $B_{1,2}$.

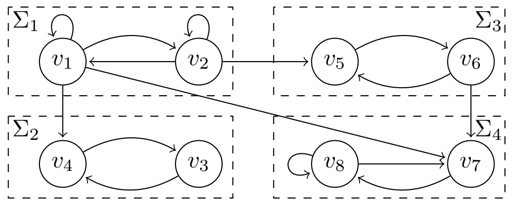
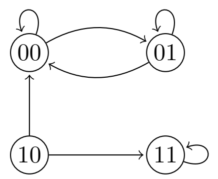
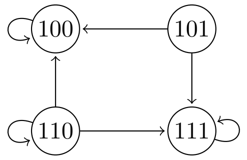


Fig. 2: SCC decomposition of a BN.



(a) Transition graph of B_1.



(b) Realisation 1 of Example 2.

Fig. 3: Two transition graphs.

A state of a block is a binary vector of length equal to the size of the block which 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 below.

Definition 7 (Projection map, Compressed state, Mirror states). For a BN G and its block B, where the set of nodes in B is $V^B = \{v_1, v_2, \ldots, v_m\}$ and the set of nodes in G is $V = \{v_1, v_2, \ldots, v_m, v_{m+1}, \ldots, v_n\}$, the *projection map* $\delta_B : X \to X^B$ is given by $\boldsymbol{x} = (x_1, x_2, \ldots, x_m, x_{m+1}, \ldots, x_n) \mapsto \delta_B(\boldsymbol{x}) = (x_1, x_2, \ldots, x_m)$. For any set of states $S \subseteq X$, we define $\delta_B(S) = \{\delta_B(\boldsymbol{x}) : \boldsymbol{x} \in S\}$. The projected state $\delta_B(\boldsymbol{x})$ is called a *compressed state* of $\boldsymbol{x}$. For any state $\boldsymbol{x}^B \in X^B$, we define its set of *mirror states* in G as $\mathcal{M}_G(\boldsymbol{x}^B) = \{\boldsymbol{x} \mid \delta_B(\boldsymbol{x}) = \boldsymbol{x}^B\}$. For any set of states $S^B \subseteq X^B$, its set of mirror states is $\mathcal{M}_G(S^B) = \{\boldsymbol{x} \mid \delta_B(\boldsymbol{x}) \in S^B\}$.

The concept of the projection map can be extended to blocks. Given a block with nodes $V^B = \{v_1, v_2, \ldots, v_m\}$, let $V^{B'} = \{v_1, v_2, \ldots, v_j\} \subseteq V^B$. We can define $\delta_{B'} : X^B \to X^{B'}$ as $\boldsymbol{x} = (x_1, x_2, \ldots, x_m) \mapsto \delta_{B'}(\boldsymbol{x}) = (x_1, x_2, \ldots, x_j)$ and for a set of states $S^B \subseteq X^B$, we define $\delta_{B'}(S^B) = \{\delta_{B'}(\boldsymbol{x}) : \boldsymbol{x} \in S^B\}$.

3.2 Detection of attractors in blocks

An elementary block does not depend on any other block while a non-elementary block does. Therefore, they should be treated separately. We first consider the case of elementary blocks. An elementary block is in fact a BN; therefore, the notion of attractors of an elementary block is given by the definition of attractors of a BN. Next, we introduce the following concept.

Definition 8 (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 $A^B_{k'} \in \mathcal{A}^B$ such that $\delta_B(A_k) \subseteq A^B_{k'}$.

Example 1. Consider the Boolean network shown in Figure 2. The Boolean functions of this network are given as follows:

$$\begin{cases} f_1 = x_1 \wedge x_2, & f_2 = x_1 \vee \neg x_2, \\ f_3 = \neg x_4, & f_4 = x_1 \wedge \neg x_3, \\ f_5 = x_2 \wedge x_6, & f_6 = x_5, \\ f_7 = (x_1 \vee x_6) \wedge x_8, & f_8 = x_7 \vee x_8. \end{cases}$$

It has 10 attractors, i.e., $\mathcal{A} = \{\{(0*100000)\}, \{(0*100001)\}, \{(11010000)\}, \{(11010011)\}, \{(11011100)\}, \{(11011111)\}, \{(11100000)\}, \{(11100011)\}, \{(11101100)\}, \{(11101111)\}\}$ ($*$ means either 0 or 1). Nodes v_1 and v_2 form an elementary block B_1. Since B_1 is an elementary block, it can be viewed as a BN. The transition graph of this block is shown in Figure 3a. Its set of attractors is $\mathcal{A}^{B_1} = \{\{(0*)\}, \{(11)\}\}$ (nodes are arranged as v_1, v_2). We have $\delta_{B_1}(\{(0*100000)\}) = \{(0*)\} \in \mathcal{A}^{B_1}$ and $\delta_{B_1}(\{(0*100001)\}) = \{(0*)\} \in \mathcal{A}^{B_1}$. For the remaining attractors, their compressed set of state is always $\{(11)\}$, which belongs to $\mathcal{A}$. Hence, block B_1 preserves the attractors of the original BN G.

With Definition 8, we have the following lemma and theorem. We refer to [25] for the proofs of all the lemmas, theorems and corollaries in this paper.

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 $\Phi \subseteq \mathcal{M}_G(\Phi^B)$.

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

For an elementary block B in a BN G, the mirror states of its attractor states cover all G's attractor states according to Lemma 1 and Theorem 1. Therefore, by searching from the mirror states only instead of the whole state space, we can detect all the attractor states of G.

We now proceed to consider the case of non-elementary 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 result in at least one elementary block and usually one or more non-elementary blocks. 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 child block with respect to the attractors of its parent elementary block. In some cases, a realisation of a 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 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 block may have more than one realisation.

By the following two definitions, we explain in details what realisations are. We first introduce the concept of crossability and cross operations in Definition 9. The concept of crossability specifies a special relation between states of a non-elementary block and of its parent blocks, while the cross operations are used for merging attractors of two blocks when recovering the attractors of the original BN.

Definition 9 (Crossability, Cross operations). Let G be a BN and let B_i be a non-elementary block in G with the set of nodes $V^{B_i} = \{v_{p_1}, v_{p_2}, \ldots, v_{p_s}, v_{q_1}, v_{q_2}, \ldots, v_{q_t}\}$, where q_k ($k \in [1, t]$) are the indices of the control nodes also contained in B_i's parent block B_j and p_k ($k \in [1, s]$) are the indices of the remaining nodes. We denote the set of nodes in B_j as $V^{B_j} = \{v_{q_1}, v_{q_2}, \ldots, v_{q_t}, v_{r_1}, v_{r_2}, \ldots, v_{r_u}\}$, where r_k ($k \in [1, u]$) are the indices of the non-control nodes in B_j. Let further $\boldsymbol{x}^{B_i} = (x_1, x_2, \ldots, x_s, y^i_1, y^i_2, \ldots, y^i_t)$ 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. 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 $\Pi(\boldsymbol{x}^{B_i}, \boldsymbol{x}^{B_j}) = (x_1, x_2, \ldots, x_s, y^i_1, y^i_2, \ldots, y^i_t, z_1, z_2, \ldots, z_u)$. The notion of crossability naturally extends to two elementary blocks; any two states of any two elementary blocks are always crossable.
We say a set of states $S^{B_i} \subseteq X^{B_i}$ and a set of states $S^{B_j} \subseteq X^{B_j}$ are crossable, denoted as $S^{B_i} \; \mathcal{C} \; S^{B_j}$, if at least one of the set is empty or the following two conditions hold: 1) for any state $\boldsymbol{x}^{B_i} \in S^{B_i}$, there always exists a state $\boldsymbol{x}^{B_j} \in S^{B_j}$ such that $\boldsymbol{x}^{B_i}$ and $\boldsymbol{x}^{B_j}$ are crossable; 2) vice versa. The cross operation of two crossable non-empty sets of states S^{B_i} and S^{B_j} are defined as $\Pi(S^{B_i}, S^{B_j}) = \{\Pi(\boldsymbol{x}^{B_i}, \boldsymbol{x}^{B_j}) \mid \boldsymbol{x}^{B_i} \in S^{B_i}, \boldsymbol{x}^{B_j} \in S^{B_j}$ 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., $\Pi(S^{B_i}, S^{B_j}) = S^{B_i}$ if $S^{B_j} = \emptyset$ and $\Pi(S^{B_i}, S^{B_j}) = S^{B_j}$ if $S^{B_i} = \emptyset$.
Let $\mathcal{S}^{B_i} = \{S^{B_i} \mid S^{B_i} \subseteq X^{B_i}\}$ be a set of states set in B_i and $\mathcal{S}^{B_j} = \{S^{B_j} \mid S^{B_j} \subseteq X^{B_j}\}$ be a set of states set 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 for any states set $S^{B_i} \in \mathcal{S}^{B_i}$, there always exists a states set $S^{B_j} \in \mathcal{S}^{B_j}$ such that S^{B_i} and S^{B_j} are crossable; 2) vice versa. The cross operation of two crossable sets of states sets $\mathcal{S}^{B_i}$ and $\mathcal{S}^{B_j}$ are 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 $S_i \; \mathcal{C} \; S_j\}$.

With the crossability defined, the definition of a realisation is now given as follows.

Definition 10 (Realisation of a block). Let B_i be a non-elementary block formed by merging an SCC with its control nodes. Let nodes $u_1, u_2, \ldots, u_r$ be all the control nodes of B_i which are also contained by its single and elementary parent block B_j (we can always merge all B_i's ancestor blocks to form B_j if B_i has more than one parent block or has a non-elementary parent block). Let

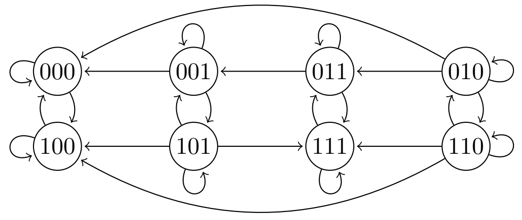


Fig. 4: Realisation 2 of Example 2.

$A_1^{B_j}, A_2^{B_j}, \ldots, A_t^{B_j}$ 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;
2) the state space of this realisation is crossable with $A_k^{B_j}$;
3) for any transition $\boldsymbol{x}^{B_i} \to \tilde{\boldsymbol{x}}^{B_i}$ in this realisation, if this transition is caused by a non-control node, the transition should be regulated by the Boolean function of this node; if this transition is caused by the updating of a control node, one can always find two states $\boldsymbol{x}^{B_j}$ and $\tilde{\boldsymbol{x}}^{B_j}$ in $A_k^{B_j}$ such that there is a transition from $\boldsymbol{x}^{B_j}$ to $\tilde{\boldsymbol{x}}^{B_j}$ in $A_k^{B_j}$, $\boldsymbol{x}^{B_i} \,\mathcal{C}\, \boldsymbol{x}^{B_j}$ and $\tilde{\boldsymbol{x}}^{B_i} \,\mathcal{C}\, \tilde{\boldsymbol{x}}^{B_j}$;
4) for any transition $\boldsymbol{x}^{B_j} \to \tilde{\boldsymbol{x}}^{B_j}$ in $A_k^{B_j}$, one can always find a transition $\boldsymbol{x}^{B_i} \to \tilde{\boldsymbol{x}}^{B_i}$ in this realisation such that $\boldsymbol{x}^{B_i} \,\mathcal{C}\, \boldsymbol{x}^{B_j}$ and $\tilde{\boldsymbol{x}}^{B_i} \,\mathcal{C}\, \tilde{\boldsymbol{x}}^{B_j}$.

We consider asynchronous networks and hence the realisation of a block is also in accordance with the asynchronous updating scheme, i.e., at each time, only one node is updated. Moreover, the asynchronous updating scheme also enables us to form realisations with the control nodes only. If the updating scheme is synchronous, the realisation will need to be formed with all the nodes in the parent block [26] and all the following theorems and proofs need to be adjusted to synchronous updating schemes accordingly.

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

Example 2. Consider the BN shown in Figure 2. The network contains four SCCs $\Sigma_1, \Sigma_2, \Sigma_3$ and Σ_4. For any Σ_i ($i \in [1, 4]$), we form a block B_i by merging Σ_i with its control nodes. Block B_1 is an elementary block and its transition graph is shown in Figure 3a. Block B_1 has two attractors, i.e., $\{(11)\}$ and $\{(0*)\}$. Regarding the first attractor, block B_3 has a realisation by setting node v_2 to contain only the transition $\{(1) \to (1)\}$. Its transition graph is shown in Figure 3b. Regarding the second attractor, block B_3 has a realisation by setting node v_2 to contain the following transitions $\{(0) \to (*), (1) \to (*)\}$. The transition graph of this realisation is shown in Figure 4.

A realisation of a block takes care of the dynamics of the undetermined nodes and instantiates a transition system of the block. Therefore, we can extend the attractor definition to realisations and to non-elementary blocks as follows.

Definition 11 (Attractors of a non-elementary block). An attractor of a realisation of a non-elementary block is a set of states satisfying 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. The attractors of a non-elementary block is the set of the attractors of all realisations of the block.

With the definition of attractors of non-elementary blocks, we can relax Definition 10 by allowing B_j to be a single and either elementary or non-elementary parent block with known attractors. This is due to the fact that when forming the realisations of a non-elementary block, we only need the attractors of its parent block that contains all its control nodes, no matter whether this parent block is elementary or not. In other words, computing attractors for non-elementary blocks requires the knowledge of the attractors of its parent block that contains all its control nodes. Therefore, we need to consider blocks in a specific order which guarantees that when computing attractors for block B_i, the attractors of its parent block that contains all B_i's control nodes are already available. To facilitate this, we introduce the concept of a credit as follows.

Definition 12 (Credit). Given a BN G, an elementary block B_i of G has a credit of 0, denoted as $\mathcal{P}(B_i) = 0$. Let B_j be a non-elementary block and $B_{j_1}, \ldots, B_{j_{p(j)}}$ be all its parent blocks. The credit of B_j is $\mathcal{P}(B_j) = max_{k=1}^{p(j)}(\mathcal{P}(B_{j_k})) + 1$.

3.3 Recover attractors of the original BN

After computing attractors for all the blocks, we need to recover attractors for the original BN. This is achievable by the following theorem for recovering the attractors of two blocks in the original BN.

Theorem 2. Given a BN G with B_i and B_j being 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. If B_i and B_j are both elementary blocks or B_i is an elementary and single parent block of B_j, then $\mathcal{A}^{B_i} \,\mathcal{C}\, \mathcal{A}^{B_j}$ and $\Pi(\mathcal{A}^{B_i}, \mathcal{A}^{B_j})$ is the set of attractors of $B_{i,j}$.

Finally, from Theorem 2 we obtain the following corollary which states that for specific configurations of blocks, certain orderings according to which the blocks are merged are equivalent in terms of the resulting attractor set for the merged block.

Corollary 1. Given a BN G with B_i, B_j, and B_k being its three blocks, let $\mathcal{A}^{B_i}$, $\mathcal{A}^{B_j}$, and $\mathcal{A}^{B_k}$ be the sets of attractors for blocks B_i, B_j, and B_k, respectively. If the three blocks are all elementary blocks or B_i is an elementary block and it is the only parent block of B_j and B_k, it holds that $\Pi(\Pi(\mathcal{A}^{B_i}, \mathcal{A}^{B_j}), \mathcal{A}^{B_k}) = \Pi(\Pi(\mathcal{A}^{B_i}, \mathcal{A}^{B_k}), \mathcal{A}^{B_j})$.

The above developed theoretical background with Theorem 2 being its core result, allows us to design a new decomposition-based approach towards detection of attractors in large asynchronous BNs. The idea is as follows. We divide a BN into blocks according to the detected SCCs. We order the blocks in the ascending order based on their credits and detect attractors of the ordered blocks one by one in an iterative way. According to Theorem 2, we can perform a cross operation for any two elementary blocks (credits 0) or an elementary block (credit 0) with one of its child blocks (credit 1) which has no other parent blocks to recover the attractors of the two merged blocks. The resulting merged block will form a new elementary block, i.e., one with credit 0. By iteratively performing the cross operation until a single elementary block containing all the nodes of the BN is obtained, we can recover the attractors of the original BN. The details of this new approach are discussed in the next section.

4 Implementation

We first introduce a BDD-based algorithm to detect attractors for relatively small BNs in Section 4.1. Then we describe how our SCC-based decomposition method can be implemented using the BDD-based algorithm in Section 4.2.

4.1 BDD-based attractor detection algorithm

Attractors of an asynchronous BN are in fact bottom strongly connected components (BSCCs) in the transition system of the BN. Thus, detecting attractors is the same as detecting the BSCCs. Formally, the definition of BSCCs is given as follows.

Definition 13. A bottom strongly connected component (BSCC) is an SCC Σ such that no state outside Σ is reachable from Σ.

We encode a BN with BDDs, and adapt the hybrid Tarjan algorithm described in Algorithm 7 of [27] to detect BSCCs in the corresponding transition system of the BN. Given a transition system $\mathcal{T} = \langle S, S_0, T\rangle$, our attractor detection algorithm $\textsc{Detect}(\mathcal{T})$ in Algorithm 1 computes the set of BSCCs in $\mathcal{T}$. If $\mathcal{T}$ is converted from a BN G, then $\textsc{Detect}(\mathcal{T})$ computes all the attractors of G. The correctness of Algorithm 1 is guaranteed by the following two propositions.

Proposition 1. The first SCC returned by the Tarjan's algorithm is a BSCC.

Proposition 2. If a state that reaches a BSCC is located outside the BSCC, then this state is not contained by any BSCC.

The first proposition can be deduced from the fact that the Tarjan's algorithm is a depth-first search. The second one comes from the definition of BSCCs, as no states inside a BSCC can lead to a state in any other BSCC. In Algorithm 1, the hybrid Tarjan algorithm $HybridTarjan(s, T)$ takes as input a starting state s and the transition relation T. When it finds the first SCC Σ (also a BSCC), which is reached from s, it terminates immediately and returns Σ.

With the use of BDD representation, $\textsc{Detect}(\mathcal{T})$ can deal with relatively small BNs (e.g., a BN with tens of nodes) with small memory usage. Moreover, the computation of SCCs can also benefit from the efficient BDD operations. However, real life biological BNs usually contain hundreds of nodes and the state space is exponential in the number of nodes. Therefore, $\textsc{Detect}(\mathcal{T})$ would still suffer from

Algorithm 1 Attractor detection using the hybrid Tarjan's algorithm

```
1:  procedure DETECT($\mathcal{T}$)
2:      $\mathcal{A} := \emptyset$; $X := S$;                    //$S$ is from $\mathcal{T}$
3:      while $X \neq \emptyset$ do
4:          Randomly pick a state $s \in X$;
5:          $\Sigma := HybridTarjan(s, T)$;
6:          $\mathcal{A} := \mathcal{A} \cup \Sigma$;
7:          $X := X \setminus Predecessors(\Sigma, T)$;
8:      end while
9:      return $\mathcal{A}$.
10: end procedure
```

the state space explosion problem when dealing with large BNs. Thus for large BNs, we propose to use the SCC-based decomposition method as described in Section 3. We now give the algorithm for implementing this method in the following section.

4.2 SCC-based decomposition algorithm

We describe the detection process in Algorithm 2. This algorithm takes a BN G and its corresponding transition system $\mathcal{T}$ as inputs and outputs the set of attractors of G. Lines 21-24 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 sets of 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 credits by the main **for** loop in Line 4. Otherwise, all the ancestor blocks are considered in the **for** loop in Lines 13-19. By iteratively applying the cross operation in Line 16 to the attractor sets of the ancestor blocks in the ascending order, the attractors of a new block formed by merging all the ancestor blocks are computed as assured by Theorem 2. The new block is in fact an elementary block which is a single parent of the considered non-elementary block. By considering blocks in the ascending order, the order in which blocks with the same credit are considered does not influence the final result due to Corollary 1. The correctness of the algorithm is stated as Theorem 3.

Theorem 3. Algorithm 2 correctly identifies the set of attractors of a given BN G.

The algorithm stores all computed attractors for the original SCC blocks and all auxiliary merged blocks in the dictionary structure $\mathcal{A}^{\ell}$. We use BDDs to encode transitions and the realisations are performed via BDD operations directly. As stated in Section 2.2, our implementation, which is based on the CUDD library [28], uses $2n$ BDD variables to encode a BN $G(V, \boldsymbol{f})$ with n nodes. Each state in G is encoded by n BDD variables, and a projection of a state on a subset $V' \subseteq V$ of nodes is performed by applying existential abstraction to BDD variables for nodes in $V \setminus V'$. As a state for a block B is encoded by $|V^B|$ BDD variables, the variables in $V \setminus V^B$ are set to "-" in the BDD representation, and therefore, can be ignored. This way, after we verify that S^{B_i} and S^{B_j} are crossable, i.e., $S^{B_i}\ \mathcal{C}\ S^{B_j}$, the cross operation

Algorithm 2 SCC-based decomposition algorithm

```
procedure SCC_DETECT($G$, $\mathcal{T}$)
    $B :=$ FORM_BLOCK($G$); $\mathcal{A} := \emptyset$; $B_a := \emptyset$; $k :=$ size of $B$;
    initialise dictionary $\mathcal{A}^\ell$;        //$\mathcal{A}^\ell$ is a dictionary storing the set of attractors for each block
    for $i := 1; i <= k; i++$ do
        if $B_i$ is an elementary block then
            $\mathcal{T}^{B_i} :=$ transition system converted from $B_i$;        //see Section 2.2 for more details
            $\mathcal{A}_i :=$ DETECT($\mathcal{T}^{B_i}$);
        else $\mathcal{A}_i := \emptyset$;
            if $B_i^p$ is the only parent block of $B_i$ then
                $\mathcal{A}_i^p := \mathcal{A}^\ell.getAtt(B_i^p)$;        //obtain attractors of $B_i^p$
            else $B^p := \{B_1^p, B_2^p, \ldots, B_m^p\}$ be the ancestor blocks of $B_i$ (ascending ordered);
                $B_c := B_1^p$;        //$B^p$ is ordered based on credit
                for $j := 2; j <= m; j++$ do
                    $B_{c,j} :=$ a new block containing nodes in $B_c$ and $B_j^p$;
                    if $(\mathcal{A}_i^p := \mathcal{A}^\ell.getAtt(B_{c,j})) == \emptyset$ then
                        $\mathcal{A}_i^p := \Pi(\mathcal{A}^\ell.getAtt(B_c), \mathcal{A}^\ell.getAtt(B_j))$; $\mathcal{A}^\ell.add(B_{c,j}, \mathcal{A}_i^p)$
                    end if
                    $B_c := B_{c,j}$;
                end for
            end if
            for $A \in \mathcal{A}_i^p$ do
                $\mathcal{T}^{B_i}(A) := \langle S^{B_i}(A), T^{B_i}(A)\rangle$;        //obtain the realisation of $B_i$ with $A$
                $\mathcal{A}_i := \mathcal{A}_i \cup$ DETECT($\mathcal{T}^{B_i}(A)$);
            end for
        end if
        $\mathcal{A}^\ell.add(B_i, \mathcal{A}_i)$;        //the add operation will not add duplicated elements
        if $B_a! = \emptyset$ then $\mathcal{A} = \Pi(\mathcal{A}_i, \mathcal{A})$; $B_a := B_{a,i}$; $\mathcal{A}^\ell.add(B_a, \mathcal{A})$;
        else $B_a := B_i$
        end if
    end for
    return $\mathcal{A}$.
end procedure

procedure FORM_BLOCK($G$)
    decompose $G$ into SCCs and form blocks with SCCs and their control nodes;
    order the blocks in an ascending order according to their credits; $B := (B_1, \ldots, B_k)$;
    return $B$.        //B is the list of blocks after ordering
end procedure
```

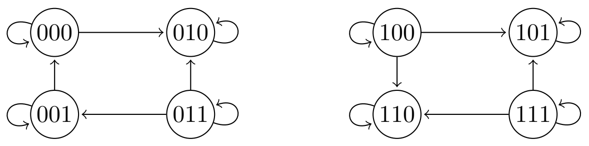


Fig. 5: Transition graphs of the two realisations for block B_2.

$\Pi(S^{B_i}, S^{B_j})$ is equivalent to the intersection of two BDDs, i.e., $bdd_{S^{B_i}}$ and $bdd_{S^{B_j}}$ encoding S^{B_i} and S^{B_j}, respectively. Formally, we have that $\Pi(S^{B_i}, S^{B_j}) = bdd_{S^{B_i}} \cap bdd_{S^{B_j}}$. Let $\mathcal{T}^B = \langle S^B, T^B\rangle$ be the transition system converted from block B, and let V^C be the set of control nodes in B. The set of states $S^B(A)$ of the realisation of block B with respect to attractor A is $\mathcal{M}_B(\delta_C(A))$ and the transition relation $T^B(A)$ of the realisation is $T^B|_{S^B(A)}$. We continue to illustrate in the following example how Algorithm 2 detects attractors.

Example 3. Consider the BN shown in Example 2 and its four blocks. Block B_1 is an elementary block and it has two attractors, i.e., $\mathcal{A}_1 = \{\{(0*)\}, \{(11)\}\}$. 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. B_1 has two attractors so there are two realisations for B_2. The transition graphs of the two realisations are shown in Figure 5. We get three attractors for block B_2, i.e., $\mathcal{A}_2 = \{\{(010)\}, \{(101)\}, \{(110)\}\}$. Performing a cross operation, we get the attractors of the merged block $B_{1,2}$, i.e., $\mathcal{A}_{1,2} = \Pi(\mathcal{A}_1, \mathcal{A}_2) = \{\{(0*10)\}, \{(1101)\}, \{(1110)\}\}$. In Example 2, we have shown the two realisations of B_3 with respect to the two attractors of B_1. Clearly, B_3 has three attractors, i.e., $\mathcal{A}_3 = \{\{(*00)\}, \{(100)\}, \{(111)\}\}$. Merging $B_{1,2}$ and B_3, we get the attractors of the merged block $B_{1,2,3}$, i.e., $\mathcal{A}_{1,2,3} = \Pi(\mathcal{A}_{1,2}, \mathcal{A}_3) = \{\{(0*10\,00)\}, \{(110100)\}, \{(110111)\}, \{(111000)\}, \{(111011)\}\}$. B_4 has two parent blocks. Therefore, we need to merge B_4's ancestors (B_1 and B_3) as its new parent block. After merging, we get the attractors of the merged block as $\mathcal{A}_{1,3} = \Pi(\mathcal{A}_1, \mathcal{A}_3) = \{\{(0*00)\}, \{(1100)\}, \{(1111)\}\}$. There are three attractors so there will be three

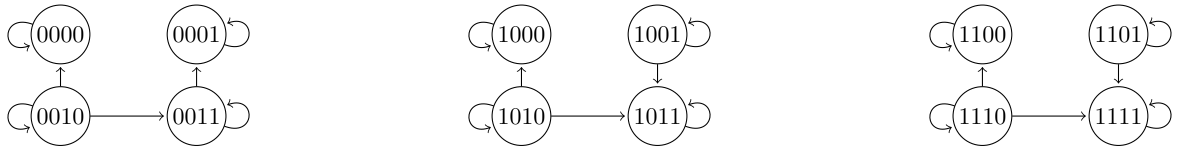


Fig. 6: Transition graphs of the three realisations for block B_4.

realisations for block B_4. The transition graphs of the three realisations are shown in Figure 6. From the transition graphs, we easily get the attractors of B_4, i.e., $\mathcal{A}_4 = \{\{(0000)\}, \{(0001)\}, \{(1000)\}, \{(1011)\}, \{(1100)\}, \{(1111)\}\}$. Now the attractors for all the blocks have been detected. We can then obtain the attractors of the BN by applying one more cross operation, i.e., $\mathcal{A} = \mathcal{A}_{1,2,3,4} = \Pi(\mathcal{A}_{1,2,3}, \mathcal{A}_4) = \{\{(0 * 100000)\}, \{(0 * 100001)\}, \{(11010000)\}, \{(11010011)\}, \{(11011100)\}, \{(11011111)\}, \{(11100000)\}, \{(11100011)\}, \{(11101100)\}, \{(11101111)\}\}$.

5 Evaluation

#	nodes	#	nodes	#	nodes
0	Apoptosis	6	p21	13	FGFR3_stimulus
1	BCL2	8	TAOK	5	Growth_Arrest
2	FOXO3	9	ATM	10	DNA_damage
3	Proliferation	15	TAK1	12	EGFR_stimulus
4	p70	11	EGFR	17	TGFR_stimulus
14	SMAD	16	TGFBR		
7	AKT AP1 ATF2 CREB DUSP1 FGFR3 ELK1 ERK FOS FRS2 GAB1 GADD45 GRB2 JNK JUN MAP3K1_3 MAX MDM2 MEK1_2 MSK MTK1 MYC PDK1 PI3K PKC PLCG PPP2CA PTEN RAF RAS RSK SOS SPRY p14 p38 p53				

TABLE 1: Nodes of the MAPK pathway (mutant r3) in SCCs as shown in Figure 8. The # refers to the SCC index.

We have implemented the decomposition algorithm presented in Section 4 in the model checker MCMAS [30]. In this section, we demonstrate the efficiency of our method using two real-life biological systems. One is a logical MAPK network model of [29] containing 53 nodes and the other is a Boolean network model of apoptosis, originally presented in [31], containing 97 nodes. All the experiments are conducted on a computer with an Intel Xeon W3520@2.67GHz CPU and 12GB memory. Notice that we tried to apply genYsis [7] to these two systems, but it failed in both cases to detect attractors within 5 hours.

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 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 crass-talk 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, effective therapeutic strategies. In [29] 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. For the wiring of the logical model, we refer to [29]. 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 [29], respectively, and therefore we refer to them as as MAPK_r3 and MAPK_r4. However, in contrast to the original variants r3 and r4, we do not set the values for the four stimuli nodes to 0 but perform the computations for all 2^4 possible fixed sets of values for these nodes. For the remaining nodes, all possible initial states are considered as in [29]. In consequence, our results for MAPK_r3 and MAPK_r4 include the attractors for variants r7, r13 and r8, r14 of [29], respectively. The corresponding SCC structure of the mutant MAPK_r3 is shown in Figure 8 and Table 1. We compute the attractors of the MAPK_r3 and MAPK_r4 BNs using both the BDD-based algorithm, i.e., Algorithm 1 and our decomposition algorithm, i.e., Algorithm 2. The two algorithms compute the same attractors for the same network. We show in the left part of Table 2 the number of attractors and the computational time costs (in seconds) for both mutants. Besides, we show the speedups of Algorithm 2 with respect to Algorithm 1.

Notice that the analysis for the MAPK network in [29] was performed based on reduced versions of the network due to the computation limitations of the tool GINsim. The reduced versions contain 16 to 18 nodes only, while our computations are performed for the full model containing 53 nodes. The advanced computation capability of our method provides the possibility for more accurate analysis which may lead to new insights in the analysis of biological networks. The total number of attractors reported in [29] for the mutants r3, r7, and r13, is only 4 while we identify 20. This inconsistency is due to that much information is missing in the reduced versions and hence the attractors of the reduced versions cannot reflect the states of the missing nodes. Based on the reduced version analysis, the authors reported in [29] that when DNA damage stimulus is

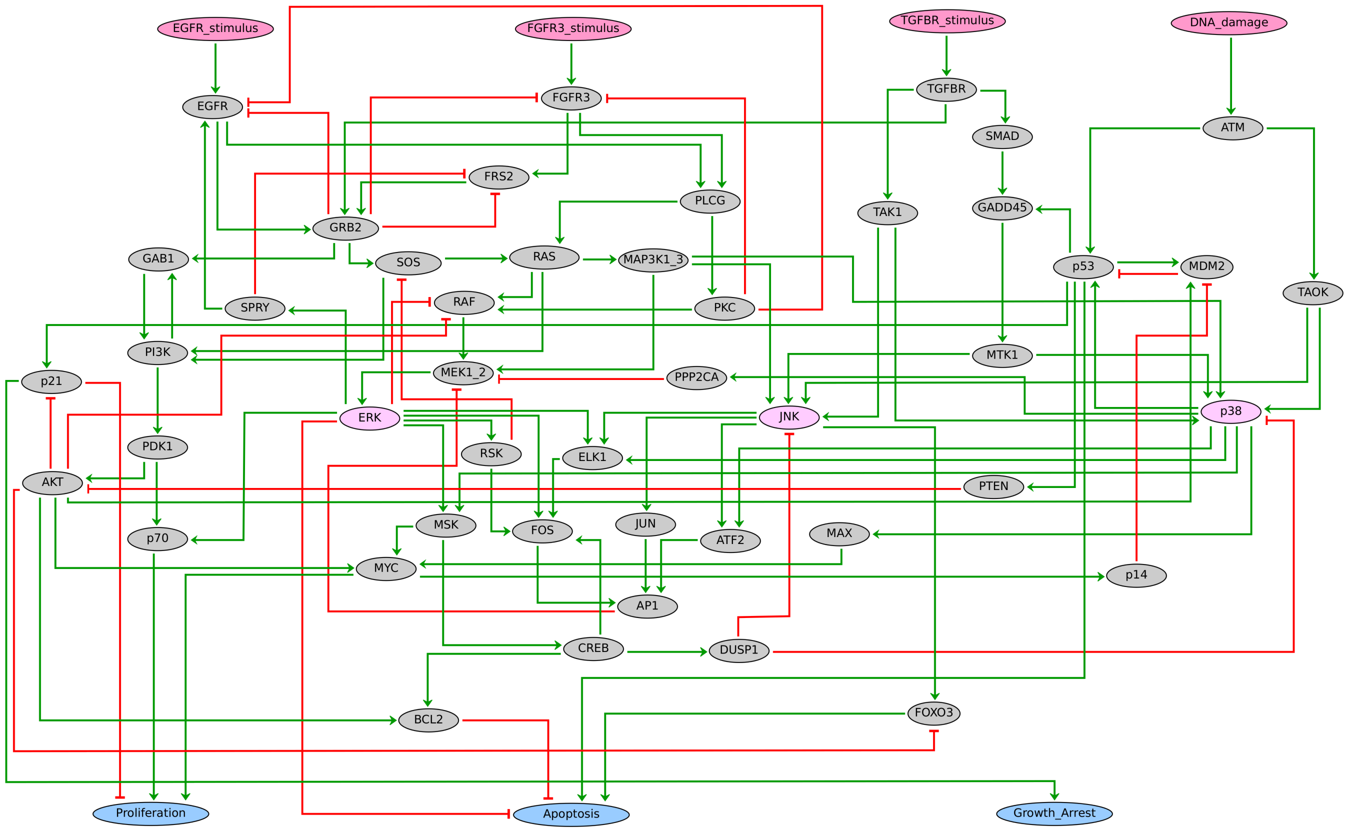


Fig. 7: Wiring of the MAPK logical model of [29]. The diagram contains three types of nodes: stimuli nodes (pink), signalling component nodes (gray) with highlighted MAPK protein nodes (light pink), and cell fate nodes (blue). Green arrows and red blunt arrows represent positive and negative regulations, respectively. For detailed information on the Boolean model of the MAPK network containing all modelling assumptions and specification of the logical rules refer to [29] and the supplementary material thereof.

Networks	# attractors	Time(second)		Speedup	Networks	# attractors	Time(second)		Speedup
		Alg. 1	Alg. 2				Alg. 1	Alg. 2	
MAPK_r3	20	6.070	2.614	**2.32**	apoptosis	1024	1633.970	103.856	**15.73**
MAPK_r4	24	11.674	1.949	**5.99**	apoptosis*	2048	8564.680	218.230	**39.25**

TABLE 2: Evaluation results on two real-life biological systems.

absent, the cell is still able to escape apoptotic cell death by down-regulating p53 signalling. This observation, however, neglected an important component, i.e., p38. It is true that p53 can directly induce apoptosis; p53 itself, however, is affected by p38. Without touching the signal of p53, by up-regulating p38 signalling, the cell can reach the apoptotic cell death. This observation is consistent with [34] and [35].

Apoptosis network. Apoptosis is a process of programmed cell death and has been linked to many diseases. It is often regulated by several signaling pathways extensively linked by crosstalks. 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. One of them is a housekeeping node which value is fixed to 1 and which is used to model constitutive activation of certain nodes in the network. For the wiring of the BN model, we refer to [31]. The corresponding SCC structure of this network is shown in Figure 10 and Table 3. Similar to the MAPK network, we compute the attractors of the apoptosis network with both Algorithm 1 and Algorithm 2. The results are shown in the right part of Table 2. Moreover, we also consider the network where the value of housekeeping is not fixed and show the result in the row apoptosis*. When the housekeeping node is not fixed, the state-space of the network is doubled. In both networks, the results of the two algorithms are identical. But our algorithm is much faster than Algorithm 1. The results clearly indicate that our proposed decomposition method provides better speedups with respect to Algorithm 1 for larger models. Moreover, the ability to identify all the attractors of the large apoptosis network enables biologists to validate the network with *in silico* experiments.

6 Discussions and Future Work

We have presented an SCC-based decomposition method for detecting attractors in large asynchronous BNs, which often arise and are important in the holistic study of biological systems. This problem is very challenging as the

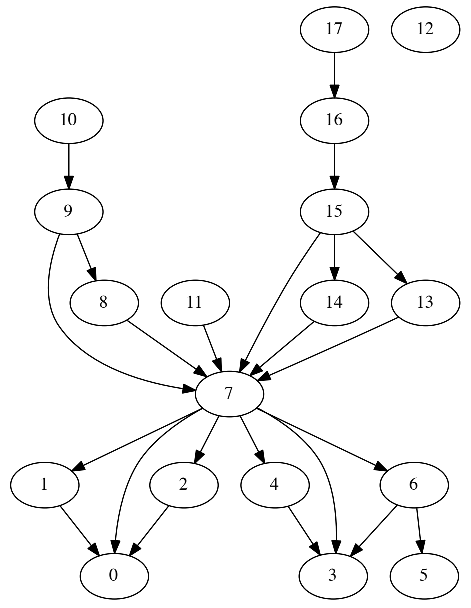


Fig. 8: The SCC structure of the MAPK network (mutant MAPK_r3). Each node represents an SCC. Model components contained in each SCC are listed in Table 1. For each pair of a parent SCC and one of its child SCCs, a directed edge is drawn pointing from the parent SCC to the child SCC. Node 12 is not connected to any other node as EGFR is set to be always true and hence the influence from EGFR_stimulus (node 12) is cut. The SCC structure of mutant MAPK_r4 is virtually the same; the only difference is that model components contained in certain SCCs are slightly different: EGFR is switched with FGFR3 and EGFR_stimulus is switched with FGFR3_stimulus.

state space of such networks is exponential in the number of nodes in the networks and therefore huge. Meanwhile, asynchrony greatly increases the difficulty of attractor detection as the density of the transition graph is inflated dramatically and the structure of attractors may be complex. Our method performs SCC-based decomposition of the network structure of a give BN to manage the cyclic dependencies among network nodes, computes the attractors of each SCC, and finally recovers the attractors of the original BN by merging the detected (partial) attractors. To the best of our knowledge, our method is the first scalable one able to deal with large biological systems modelled as asynchronous BNs, thanks to its divide and conquer strategy. We have prototyped our method and performed experiments with two real biological networks. The obtained results are very promising.

We have observed that the network structure of BNs can vary quite a lot, which potentially has impact on the performance of our proposed method. In principle, our method works well on large networks which contain several relatively small SCCs. Each of the two mutants of the MAPK network, however, contains one large SCC with 36 nodes and 17 SCCs each with one node only. Moreover, the large

#	nodes	#	nodes	#	nodes
0	apoptosis	21	Ras	42	FAS
1	gelsolin	22	Grb2_SOS	43	FASL_2
2	C3a_c_IAP	23	Shc	44	IL_1
3	I_kBb	24	Raf	45	TNFR_1
4	CAD	25	MEK	46	TNF
5	PARP	26	Pak1	47	UV
6	ICAD	27	Rac	48	UV_2
7	JNK	28	GSK_3	49	FASL
8	C8a_FLIP	29	Bad	50	PKA
10	XIAP	31	IR	51	cAMP
11	TRADD	32	IRS_P2	52	AdCy
12	RIP	33	IRS	53	GR
13	Bad_14_3_3	34	IKKdeact	54	Glucagon
14	P14_3_3	35	FLIP	55	Insulin
15	C8a_2	36	DISCa_2	56	smac_mimetics
16	C8a_DISCa_2	37	DISCa	57	P
17	C8a_DISCa	38	FADD	58	T2R
18	proC8	39	Bid	59	T2RL
19	p38	40	housekeeping		
20	ERK1o2	41	FAS_2		
9	Apaf_1 apopto A20 Bax Bcl_xl BIR1_2 c_IAP c_IAP_2 complex1 comp1_IKKa cyt_c C3ap20 C3ap20_2 C3a_XIAP C8a_comp2 C9a FLIP_2 NIK RIP_deubi smac smac_XIAP tBid TRAF2 XIAP_2 IKKa I_kBa I_kBe complex2 NF_kB C8a C3ap17 C3ap17_2				
30	IRS_P PDK1 PKB PKC PIP3 PI3K C6				

TABLE 3: Nodes of the apoptosis network in SCCs as shown in Figure 10. The # refers to the SCC index.

SCC is in the middle of the SCC network structure (see Figure 8). This network structure in fact does not fit well with our method. This explains why the speedups achieved for this network are less than 10. Both the MAPK network and the apoptosis network contain many small SCCs with only one node (see Figure 8 and Figure 10). One way to improve our method is to merge these small SCCs into larger blocks so that there will be fewer iterations in the main loop of Algorithm 1. Moreover, the single-node SCCs which do not have child SCCs are in fact leaves and they can be removed to reduce the network size. When the attractors in the reduced network are detected, we can then recover the attractors in the whole network.[4] Such optimisations will be part of our future work. We will also apply our method to other realistic large biological networks and we will develop optimisations fitted towards different SCC network structures.

ACKNOWLEDGMENTS

Qixia Yuan is supported by the National Research Fund, Luxembourg (grant 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). Andrzej Mizera contributed to this work while holding a postdoctoral researcher position at the Computer Science and Communications Research Unit, University of Luxembourg.

REFERENCES

[1] P. Dhaeseleer, S. Liang, and R. Somogyi, "Genetic network inference: from co-expression clustering to reverse engineering," *Bioinformatics*, vol. 16, no. 8, pp. 707–726, 2000.

4. This is in general related to network reduction techniques (e.g., see [36]) which aim to simplify the networks prior to dynamic analysis.

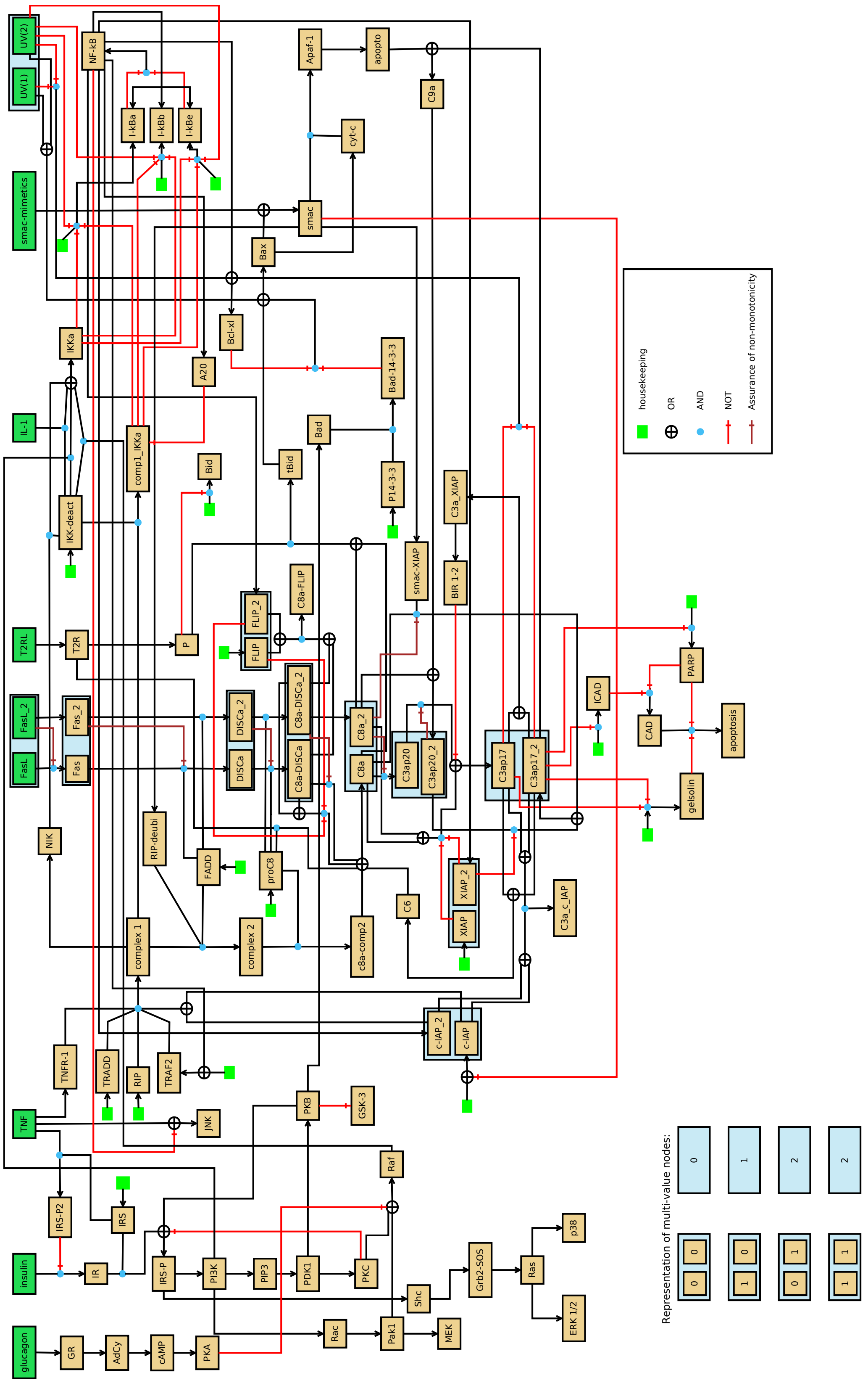


Fig. 9: The wiring of the multi-value logic model of apoptosis by Schlatter et al. [31] recast into a binary Boolean network. For clarity of the diagram the nodes I-kBa, I-kBb, and I-kBe have two positive inputs. The inputs are interpreted as connected via ⊕ (logical OR).

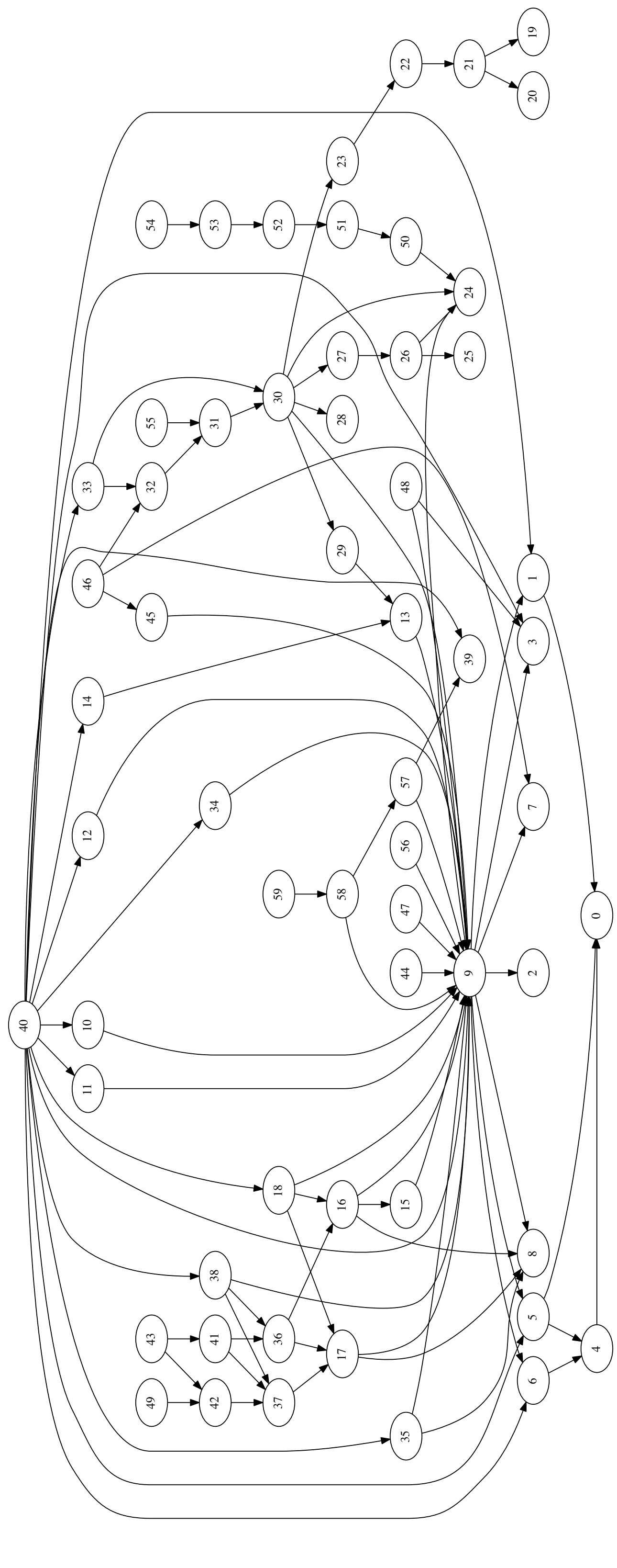


Fig. 10: The SCC structure of the apoptosis model. Each node represents an SCC in the apoptosis model. The nodes contained in each SCC are listed in Table 3. For each pair of a parent SCC and one of its child SCCs, a directed edge is added pointing from the parent SCC to the child SCC.

[2] S. Kauffman, "Homeostasis and differentiation in random genetic control networks," *Nature*, vol. 224, pp. 177–178, 1969.

[3] S. Huang, "Genomics, complexity and drug discovery: insights from Boolean network models of cellular regulation," *Pharmacogenomics*, vol. 2, no. 3, pp. 203–222, 2001.

[4] Y.-E. Sanchez-Corrales, E. R. Alvarez-Buylla, and L. Mendoza, "The arabidopsis thaliana flower organ specification gene regulatory network determines a robust differentiation process," *Journal of Theoretical Biology*, vol. 264, no. 3, pp. 971–983, 2010.

[5] R. Somogyi and L. D. Greller, "The dynamics of molecular networks: applications to therapeutic discovery," *Drug Discovery Today*, vol. 6, no. 24, pp. 1267–1277, 2001.

[6] D. J. Irons, "Improving the efficiency of attractor cycle identification in Boolean networks," *Physica D: Nonlinear Phenomena*, vol. 217, no. 1, pp. 7–21, 2006.

[7] A. Garg, L. Xenarios, L. Mendoza, and 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*, ser. LNCS, vol. 4453. Springer, 2007, pp. 62–76.

[8] A. Garg, A. Di Cara, I. Xenarios, L. Mendoza, and G. De Micheli, "Synchronous versus asynchronous modeling of gene regulatory networks," *Bioinformatics*, vol. 24, no. 17, pp. 1917–1925, 2008.

[9] E. Dubrova and M. Teslenko, "A SAT-based algorithm for finding attractors in synchronous Boolean networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 5, pp. 1393–1399, 2011.

[10] Y. Zhao, J. Kim, and M. Filippone, "Aggregation algorithm towards large-scale Boolean network analysis," *IEEE Transactions on Automatic Control*, vol. 58, no. 8, pp. 1976–1985, 2013.

[11] W. Guo, G. Yang, W. Wu, L. He, and M. Sun, "A parallel attractor finding algorithm based on Boolean satisfiability for genetic regulatory networks," *PLOS ONE*, vol. 9, no. 4, p. e94258, 2014.

[12] Q. Yuan, H. Qu, , J. Pang, and A. Mizera, "Improving BDD-based attractor detection for synchronous Boolean networks," *Science China Information Sciences*, vol. 59, no. 8, p. 080101, 2016.

[13] H. Klarner, A. Bockmayr, and H. Siebert, "Computing maximal and minimal trap spaces of Boolean networks," *Natural Computing*, vol. 14, no. 4, pp. 535–544, 2015.

[14] A. Naldi, E. Remy, D. Thieffry, and C. Chaouiya, "Dynamically consistent reduction of logical regulatory graphs," *Theoretical Computer Science*, vol. 412, no. 21, pp. 2207–2218, 2011.

[15] J. G. T. Zañudo and R. Albert, "An effective network reduction approach to find the dynamical repertoire of discrete dynamic networks," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 23, no. 2, p. 025111, 2013.

[16] A. Veliz-Cuba, B. Aguilar, F. Hinkelmann, and R. Laubenbacher, "Steady state analysis of boolean molecular network models via model reduction and computational algebra," *BMC bioinformatics*, vol. 15, no. 1, p. 221, 2014.

[17] S.-M. Choo and K.-H. Cho, "An efficient algorithm for identifying primary phenotype attractors of a large-scale Boolean network," *BMC systems biology*, vol. 10, no. 1, p. 95, 2016.

[18] T. Akutsu, S. Kosub, A. A. Melkman, and T. Tamura, "Finding a periodic attractor of a boolean network," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 9, no. 5, pp. 1410–1421, 2012.

[19] R. Thomas, "Regulatory networks seen as asynchronous automata: a logical description," *Journal of Theoretical Biology*, vol. 153, no. 1, pp. 1–23, 1991.

[20] R.-S. Wang, A. Saadatpour, and R. Albert, "Boolean modeling in systems biology: an overview of methodology and applications," *Physical Biology*, vol. 9, no. 5, p. 055001, 2012.

[21] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, no. 3, pp. 437–467, 1969.

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

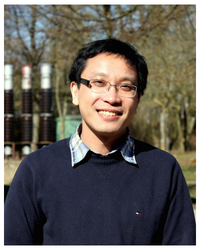
[23] C.-Y. Lee, "Representation of switching circuits by binary-decision programs," *Bell System Technical Journal*, vol. 38, no. 4, pp. 985–999, 1959.

[24] S. B. Akers, "Binary decision diagrams," *IEEE Transactions on Computers*, vol. 100, no. 6, pp. 509–516, 1978.

[25] A. Mizera, J. Pang, H. Qu, and Q. Yuan, "Taming asynchrony for attractor detection in large Boolean networks (technical report)," 2017, available online at http://arxiv.org/abs/1704.06530.

[26] ——, “A new decomposition method for attractor detection in large synchronous boolean networks,” to appear in Proc. 3nd International Symposium on Dependable Software Engineering: Theories, Tools, and Applications. IEEE, 2017.
[27] M. Kwiatkowska, D. Parker, and H. Qu, “Incremental quantitative verification for Markov decision processes,” in *Proc. 41st IEEE/IFIP International Conference on Dependable Systems & Networks*. IEEE, 2011, pp. 359–370.
[28] F. Somenzi, “CUDD: CU decision diagram package - release 2.5.1,” http://vlsi.colorado.edu/~fabio/CUDD/, 2015.
[29] L. Grieco, L. Calzone, I. Bernard-Pierrot, F. Radvanyi, B. Kahn-Perles, and D. Thieffry, “Integrative modelling of the influence of MAPK network on cancer cell fate decision,” *PLOS Computational Biology*, vol. 9, no. 10, p. e1003286, 2013.
[30] A. Lomuscio, H. Qu, and F. Raimondi, “MCMAS: An open-source model checker for the verification of multi-agent systems,” *International Journal on Software Tools for Technology Transfer*, vol. 19, no. 1, pp. 9–30, 2017.
[31] R. Schlatter, K. Schmich, I. A. Vizcarra, P. Scheurich, T. Sauter, C. Borner, M. Ederer, I. Merfort, and O. Sawodny, “ON/OFF and beyond - a boolean model of apoptosis,” *PLOS Computational Biology*, vol. 5, no. 12, p. e1000595, 2009.
[32] M. Krishna and H. Narang, “The complexity of mitogen-activated protein kinases (MAPKs) made simple,” *Cellular and Molecular Life Sciences*, vol. 65, no. 22, pp. 3525–3544, 2008.
[33] A. S. Dhillon, S. Hagan, O. Rath, and W. Kolch, “MAP kinase signalling pathways in cancer,” *Oncogene*, vol. 26, pp. 3279–3290, 2007.
[34] B. Cai, S. H. Chang, B. E. Becker, A. Bonni, and Z. Xia, “p38 MAP kinase mediates apoptosis through phosphorylation of BimEL at Ser-65,” *Journal of Biological Chemistry*, vol. 281, no. 35, pp. 25 215–25 222, 2006.
[35] X. Sui, N. Kong, L. Ye, W. Han, J. Zhou, Q. Zhang, C. He, and H. Pan, “p38 and JNK MAPk pathways control the balance of apoptosis and autophagy in response to chemotherapeutic agents,” *Cancer letters*, vol. 344, no. 2, pp. 174–179, 2014.
[36] A. Saadatpour, I. Albert, and R. Albert, “Attractor analysis of asynchronous Boolean models of signal transduction networks,” *Journal of Theoretical Biology*, vol. 266, pp. 641–656, 2010.

Dr. Andrzej Mizera received the MSc degree in Computer Science from the University of Warsaw, Poland in 2005. He then obtained his PhD in Computer Science with minor in mathematics in the area of Computational Systems Biology from the Åbo Akademi University, Turku, Finland in 2011. His research interests are related to computational and mathematical modelling of biological systems. He is currently holding a research associate position at the Luxembourg Institute of Health.



Dr. Jun Pang received his PhD in Computer Science from Vrije Universiteit Amsterdam, The Netherlands in 2004. Currently, he is a senior researcher in the Security and Trust of Software Systems research group at the University of Luxembourg. His research interests include formal methods, security and privacy, big data analytics, and computational systems biology.

Dr. Hongyang Qu is a Senior Research Fellow in the Department of Automatic Control & Systems Engineering, University of Sheffield. He is currently a member of Autonomous Systems and Robotics Research Group within the Department. He obtained his Ph.D. from the University of Warwick in 2006 and has previously held research positions at the Universite de Provence, Imperial College London and the University of Oxford. His research interests includes formal verification and model checking, in particular, efficient model checking techniques for discrete systems, real-time systems and probabilistic systems, as well as their applications.

Qixia Yuan received his MSc degrees in computer science from both the University of Luxembourg and Shandong University in 2012. He is currently a PhD student at the Computer Science and Communications Research Unit at the University of Luxembourg. His research interests focus on development and application of formal methods in systems biology.