A New Decomposition Method for Attractor Detection in Large Synchronous Boolean Networks

Andrzej Mizera[1], Jun Pang[1,2], Hongyang Qu[3], and Qixia Yuan[1⋆]

[1] Faculty of Science, Technology and Communication, University of Luxembourg
[2] Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg
[3] Department of Automatic Control & Systems Engineering, University of Sheffield
firstname.lastname@uni.lu, h.qu@sheffield.ac.uk

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

1 Introduction

Boolean networks (BN) is a well-established framework widely used for modelling biological systems, such as gene regulatory networks (GRNs). Although it is simple by mainly focusing on the wiring of a system, BN can still capture the important dynamic property of the modelled system, e.g., the *attractors*. In the literature, attractors are hypothesised to characterise cellular phenotypes [1] or to correspond to functional cellular states such as proliferation, apoptosis, or differentiation [2]. 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 [3]. The idea is to enumerate all the possible states and to run simulation 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. Later on, the performance of attractor detection has been greatly improved with the use of two techniques. The first technique is binary decision diagrams (BDDs). This type of methods [4, 5] encodes 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

⋆ Supported by the National Research Fund, Luxembourg (grant 7814267).

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 [6]. 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 and 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 [7–9] 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 recover the attractors of the original BN.

In this paper, we propose a new decomposition method for attractor detection in BNs, in particular, in large synchronous BNs, where all the nodes are updated synchronously at each time point. 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 [7] does. We explain this in more details in Section 3. Second, our new method considers the dependency relation among different sub-networks when detecting attractors of them while our previous method [8] does not require this. We show with experimental results that this consideration can significantly improve the performance of attractor detection in large BNs. Further, the decomposition method in [9] is designed for asynchronous networks while here we extend it for 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 [9].

2 Preliminaries

2.1 Boolean networks

A Boolean network (BN) is composed of two elements: binary-valued nodes, which represents 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 [10], 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

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

It states that in every step, all the nodes are updated synchronously according to their Boolean functions.

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

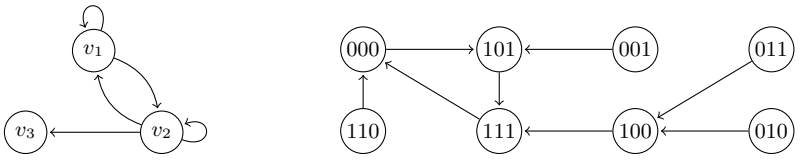
Definition 2 (State transition system). *A state transition system $\mathcal{T}$ is a 3-tuple $\langle S, S_0, T\rangle$ where S is a finite set of states, $S_0 \subseteq S$ is the initial set of states and $T \subseteq S \times S$ is the transition relation. When $S = S_0$, we write $\langle S, T\rangle$.*

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.

Example 1. A BN 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. Applying the transition relation to each of the states, we can get 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.

In the transition graph of Figure 1b, the three states $(000), (1*1)$[4] 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.

[4] We use $*$ to denote that the 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.

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

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.*

When analysing an attractor, we often need to identify transition relations between the attractor states. We call an attractor together with its state transition relation an *attractor system* (AS). The states constituting an attractor are called *attractor states*. The attractors of a BN characterise its long-run behaviour [11] and are of particular interest due to their biological interpretation.

For synchronous BNs, each state of the network can only have at most 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. The BN given in Example 1 has one attractor, i.e., $\{(000), (1*1)\}$.

2.2 Encoding BNs in BDDs

Binary decision diagrams (BDDs) were introduced by Lee in [12] and Akers in [13] to represent Boolean functions [12, 13]. 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 then be encoded in 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 $T : 2^{|V|+|V'|} \rightarrow \{0, 1\}$, 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 utilizes two basis functions: $Image(X, T) = \{s' \in S \mid \exists s \in X \text{ such that } (s, s') \in T\}$, which returns the set of target states that can be reached from any state in $X \subseteq S$ with a single transition in T; $Preimage(X, T) = \{s' \in S \mid \exists s \in X \text{ such that } (s', s) \in T\}$, which returns the set of predecessor states that can reach a state in X with a single transition. To simplify the presentation, we define $Preimage^i(X, T) = \underbrace{Preimage(...(Preimage(X, T)))}_{i}$ with $Preimage^0(X, T) = X$.

In this way, the set of all states that can reach a state in X via transitions in T is defined as an iterative procedure $Predecessors(X,T) = \bigcup_{i=0}^{n} Preimage^n(X,T)$ such that $Preimage^n(X,T) = Preimage^{n+1}(X,T)$. Given a set of states $X \subseteq S$, the projection $T|_X$ of T on X is defined as $T|_X = \{(s,s') \in T \mid s \in X \wedge s' \in X\}$.

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 consists of three steps. First, we divide a BN into sub-networks called *blocks* and this step is performed on the *network structure*, instead of the state transition system of the network. Second, we detect attractors in blocks. Last, we recover attractors of the original BN based on attractors of the blocks.

3.1 Decompose a BN

We start by giving the formal definition of a block.

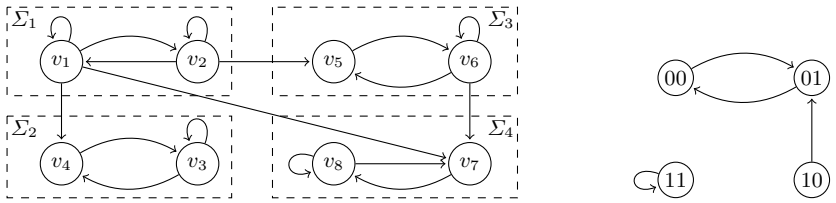
Definition 4 (Block). *Given a BN $G(V, \boldsymbol{f})$ with $V = \{v_1, v_2, \ldots, v_n\}$ and $\boldsymbol{f} = \{f_1, f_2, \ldots, f_n\}$, a* block *$B(V^B, \boldsymbol{f^B})$ is a subset of the network, where $V^B \subseteq V$. For any node $v_i \in V^B$, if B contains all the parent nodes of v_i, 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 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 at each 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, there is a directed path from u to v and vice versa.*

We first decompose a given BN into SCCs. Figure 2a 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 2a, 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. An SCC together with its control nodes forms a *block*. For example, in Figure 2a, 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. By adding directed



(a) SCC decomposition. (b) Transition graph of block B_1.

Fig. 2: SCC decomposition and the transition graph of block B_1.

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. 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 together to form a larger block $B_{1,2}$.

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 6 (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 $\pi_B : X \to X^B$ *is given by $\boldsymbol{x} = (x_1, x_2, \ldots, x_m, x_{m+1}, \ldots, x_n) \mapsto \pi_B(\boldsymbol{x}) = (x_1, x_2, \ldots, x_m)$. For any set of states $S \subseteq X$, we define $\pi_B(S) = \{\pi_B(\boldsymbol{x}) : \boldsymbol{x} \in S\}$. The projected state $\pi_B(\boldsymbol{x})$ 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 $\mathcal{M}_G(\boldsymbol{x}^B) = \{\boldsymbol{x} \mid \pi_B(\boldsymbol{x}) = \boldsymbol{x}^B\}$. For any set of states $S^B \subseteq X^B$, its set of mirror states is $\mathcal{M}_G(S^B) = \{\boldsymbol{x} \mid \pi_B(\boldsymbol{x}) \in S^B\}$.*

The concept of projection map can be naturally 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 $\pi_{B'} : X^B \to X^{B'}$ as $\boldsymbol{x} = (x_1, x_2, \ldots, x_m) \mapsto \pi_{B'}(\boldsymbol{x}) = (x_1, x_2, \ldots, x_j)$, and for a set of states $S^B \subseteq X^B$, we define $\pi_{B'}(S^B) = \{\pi_{B'}(\boldsymbol{x}) : \boldsymbol{x} \in S^B\}$.

3.2 Detect attractors in a block

An elementary block does not depend on any other block while a non-elementary block does. Therefore, they can be treated separately. We first consider the case of elementary blocks. An elementary block is in fact a BN; therefore, the definition of attractors in a BN can be directly taken to the concept of an elementary block.

Definition 7 (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 $\pi_B(A_k) \subseteq A_{k'}^B$.*

(a) Transition graph of Block B_1 in G_1. (b) Transition graph of the "realisation".

Fig. 3: Two transition graphs used in Example 3 and Example 4.

Example 3. Consider the BN G_1 in Example 1. Its set of attractors is $\mathcal{A} = \{\{(000), (1*1)\}\}$. 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 B_1 is shown in Figure 3a. Its set of attractors is $\mathcal{A}^{B_1} = \{\{(00), (1*)\}\}$ (nodes are arranged as v_1, v_2). We have $\pi_{B_1}(\{(000), (1*1)\}) = \{(00), (1*)\} \in \mathcal{A}^{B_1}$, i.e., block B_1 preserves the attractors of G_1.

With Definition 7, we have the following lemma and theorem.

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, 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.

We now consider the case of non-elementary blocks.

Definition 8 (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 $\Sigma_{k'}$, where $\Sigma_{k'}$ is an ancestor of Σ_j. Denote $\Omega(\Sigma_j)$ the set of ancestor SCCs of Σ_i.*

For an SCC Σ_j, if it has no parent SCC, then this SCC can form 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.

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 by 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 $\boldsymbol{x}^{B_i} = (x_1, x_2, \ldots, x_s, y_1^i, y_2^i, \ldots, y_t^i)$ be a state of B_i and $\boldsymbol{x}^{B_j} = (y_1^j, y_2^j, \ldots, y_t^j, z_1, z_2, \ldots,$*

$z_u)$ *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.,* $y_k^i = y_k^j$ *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* $\Pi(\boldsymbol{x}^{B_i}, \boldsymbol{x}^{B_j}) = (x_1, x_2, \ldots, x_s, y_1^i, y_2^i, \ldots, y_t^i, z_1, z_2, \ldots, z_u)$*. The notion of crossability naturally extends to two blocks without common nodes: any two states of these blocks are crossable.*

We say $S^{B_i} \subseteq X^{B_i}$ *and* $S^{B_j} \subseteq X^{B_j}$ *are crossable, denoted as* $S^{B_i} \mathcal{C} S^{B_j}$*, if at least one of the sets is empty or the following two conditions hold: 1) for any state* $\boldsymbol{x}^{B_i} \in S^{B_i}$*, there 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 on two crossable non-empty sets of states* S^{B_i} *and* S^{B_j} *is 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 family of sets of states in* B_i *and* $\mathcal{S}^{B_j} = \{S^{B_j} \mid S^{B_j} \subseteq X^{B_j}\}$ *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* $S^{B_i} \in \mathcal{S}^{B_i}$*, there exists a 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 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* $S_i \mathcal{C} S_j\}$*.*

Proposition 1. *Let* V^C *be the set of control nodes shared by two blocks* B_i *and* B_j*, i.e.,* $V^C = V^{B_i} \cap V^{B_j}$ *and let* $S^{B_i} \subseteq X^{B_i}$ *and* $S^{B_j} \subseteq X^{B_j}$*. Then* $S^{B_i} \mathcal{C} S^{B_j}$ *is equivalent to* $\pi_C(S^{B_i}) = \pi_C(S^{B_j})$*.*

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 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 10 (Realisation of a non-elementary 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 in its elementary parent block* B_j *(we can merge* 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* $A_1^{B_j}, A_2^{B_j}, \ldots, A_t^{B_j}$ *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. *the transitions are as follows: 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 $\tilde{\boldsymbol{x}}^{B_{i,j}}$ $\mathcal{C}$ $\tilde{\boldsymbol{x}}^{B_j}$; 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 $A_k^{B_j}$;

In the realisation of a non-elementary block all the nodes of its 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 out-going transition, which is contrary to the fact that the out-going 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 are not consistent with the synchronous semantics and therefore the detected "attractors" may not be attractors of the synchronous BN. Note that the decomposition method mentioned in [7] did not take care of this and therefore produces incorrect results in certain cases. We now give an example to illustrate one of such cases.

Example 4. 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 3a and its attractor is $(00) \to (10) \to (11)$. 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 3b, which contains two states with two out-going transitions. This is contrary to the synchronous semantics. Moreover, recovering attractors with the attractors in this graph will lead to a non-attractor state of the original BN, i.e., (001).

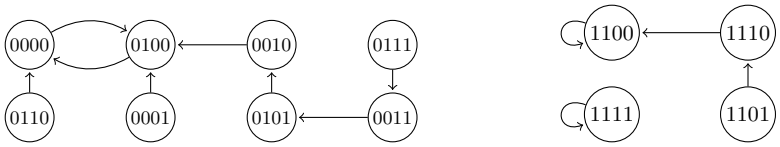
For asynchronous networks, however, such a distinction is not necessary since the situation of multiple out-going transitions is in consistent with the asynchronous updating semantics. Definition 10 forms the basis for a key difference between this decomposition method for synchronous BNS and the one for asynchronous BNs proposed in [9].

Constructing realisations for a non-elementary block is a key process for 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 some examples.

Example 5. Consider the BN in Figure 2a. Its Boolean functions are given as follows:

$$\begin{cases} f_1 = x_1 \wedge x_2, & f_2 = x_1 \vee \neg x_2, & f_3 = \neg x_4 \wedge x_3, & f_4 = x_1 \vee x_3, \\ f_5 = x_2 \wedge x_6, & f_6 = x_5 \wedge x_6, & f_7 = (x_1 \vee x_6) \wedge x_8, & f_8 = x_7 \vee x_8. \end{cases} \tag{2}$$

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 2b. Block B_1 has two attractors, i.e., $\{(0*)\}$ and $\{(11)\}$. Regarding the first attractor, block B_3 has a realisation by setting the nodes v_1 and v_2 (nodes from its parent block B_1) to contain transitions $\{(00) \to (01), (01) \to (00)\}$. The transition graph of this realisation is shown in Figure 4a. Regarding the



(a) Realisation 1 of Example 5. (b) Realisation 2 of Example 5.

Fig. 4: Transition graphs of two realisations in Example 5.

second attractor, block B_3 has a realisation by setting nodes v_1 and v_2 to contain only the transition $\{(11) \rightarrow (11)\}$. Its transition graph is shown in Figure 4b.

A realisation of a non-elementary block takes care of the dynamics of the undetermined nodes, providing a transition system of the block. Therefore, we can extend the attractor definition to realisations of 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 above definition, we can extend Definition 10 by allowing B_j to be a non-elementary block as well. 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 for non-elementary blocks requires the knowledge of the attractors of its 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 before it. The blocks are ordered topologically. For easier explanation of the order, we introduce the concept of topological credit as follows. For simplification, we refer topological credit as credit in the remaining part of the paper.

Definition 12 (Topological credit). *Given a BN G, the an elementary block B_i of G has a topological 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 topological credit 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.*

3.3 Recovering attractors for the original BN

After computing attractors for all the blocks, we need to recover attractors for the original BN, with the help of the following theorem.

Theorem 2. *Let G be a BN and let B_i be one of its blocks. Denote $\Omega(B_i)$ as the block formed by all B_i's ancestor blocks and denote $\mathcal{X}(B_i)$ as the block formed by merging B_i with $\Omega(B_i)$. $\mathcal{X}(B_i)$ is in fact an elementary block, which is also a BN. The attractors of block B_i are in fact the attractors of $\mathcal{X}(B_i)$.*

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 $S^{B_{i,j}}$. If B_i and B_j are both elementary blocks, $\mathcal{A}^{B_i}$ $\mathcal{C}$ $\mathcal{A}^{B_j}$ and $\cup_{A \in \Pi(\mathcal{A}^{B_i}, \mathcal{A}^{B_j})} A = S^{B_{i,j}}$.*

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. 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. We start from detecting attractors of elementary blocks (credit 0), and continue to detect blocks with higher credits 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 for any two elementary blocks (credits 0) to recover the attractor states of the two merged blocks. The resulting merged block will form a new elementary block, i.e., one with credit 0. The attractors can be easily identified from the set of attractor states. 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 algorithm are discussed in the next section. 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 $\cup_{S \in \Pi(\mathcal{A}^{B_i}, \mathcal{A}^{B_j})} S = S^{B_{i,j}}$.*

4 A BDD-based Implementation

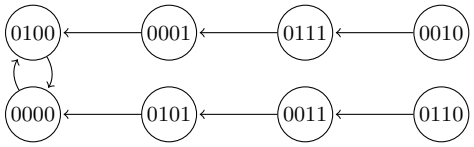
We describe the SCC-based attractor detection method in Algorithm 1. As mentioned in Section 2.2, we encode BNs in BDDs; hence most operations in this algorithm is performed with BDDs. Algorithm 1 takes a BN G and its corresponding transition system $\mathcal{T}$ as inputs, and outputs the set of attractors of G. In this algorithm, we denote by DETECT($\mathcal{T}$) a basic function for detecting attractors of a given transition system $\mathcal{T}$. Lines 23-26 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 of 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 credits by the main **for** loop in line 4. Otherwise, all the parent blocks are considered in the **for** loop in lines 13-21. By iteratively applying the cross operation in line 16 to the attractor sets of the ancestor blocks in the ascending order, the attractor states of a new block formed

Algorithm 1 SCC-based decomposition algorithm

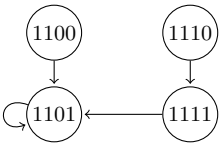
procedure SCC_DETECT($G, \mathcal{T}$)
  $B :=$ FORM_BLOCK(G); $\mathcal{A} := \emptyset$; $B_a := \emptyset$; $k :=$ size of B;
  initialise dictionary $\mathcal{A}^\ell$; *//$\mathcal{A}^\ell$ is a dictionary storing the set of attractors for each block*
  for $i := 1; i <= k; i++$ **do**
    if B_i 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($\mathcal{T}^{B_i}$); $\mathcal{A}^\ell.add((B_i, \mathcal{A}_i))$;
    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 parent blocks of B_i (ascending ordered);
        $B_c := B_1^p$; *//B^p is ordered based on credits*
        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**
            $A := \Pi(\mathcal{A}^\ell.getAtt(B_c), \mathcal{A}^\ell.getAtt(B_j))$; $\mathcal{A}_i^p := D(A)$;
              //$D(A)$ returns all the attractors from attractor states sets A
            $\mathcal{A}^\ell.add(B_{c,j}, \mathcal{A}_i^p)$;
          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
      $\mathcal{A}^\ell.add((B_i, \mathcal{A}_i))$; *//the add operation will not add duplicated elements*
      $\mathcal{A}^\ell.add((B_{i,ancestors}, \mathcal{A}_i))$; *//$B_{i,ancestors}$ means B_i and all its ancestor blocks*
      for any $B^p \in \{B_1^p, B_2^p, \ldots, B_m^p\}$ **do** *//$B_1^p, B_2^p, \ldots, B_m^p$ are parent blocks of B_i*
        $\mathcal{A}^\ell.add((B_{i,p}, \mathcal{A}_i))$;
      end for
    end if
  end for
  for $B_i \in B$ and B_i has no child block **do**
    $\mathcal{A} = D(\Pi(\mathcal{A}^\ell.get(B_i), \mathcal{A}))$;
  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

by merging all the parent blocks are computed as assured by Corollary 1. The attractors



(a) Realisation 1 of B_2.

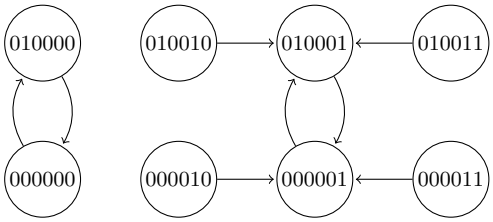


(b) Realisation 2 of B_2.

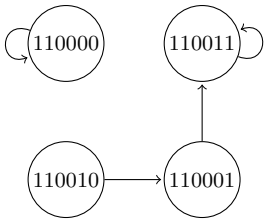
Fig. 5: Two realisations used in Example 6.

are then identified from the attractor states with one more operation. The correctness of the algorithm is stated as Theorem 4.

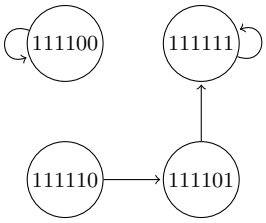
Theorem 4. *Algorithm 1 correctly identifies the set of attractors of a given BN G.*



(a) The first realisation of B_4



(b) The second realisation of B_4



(c) The third realisation of B_4

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

We continue to illustrate in Example 6 how Algorithm 1 detects attractors.

Example 6. Consider the BN shown in Example 5 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 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 Figures 5a and 5b. We get two attractors for block B_2, i.e., $\mathcal{A}_2 = \{\{(0*00)\}, \{(1101)\}\}$. Those two attractors are also attractors for the merged block $B_{1,2}$, i.e., $\mathcal{A}_{1,2} = \mathcal{A}_2$. In Example 5, we have shown the two realisations of B_3 regarding the two attractors of B_1. Clearly, B_3 has two attractors, i.e., $\mathcal{A}_3 = \{\{(0*00)\}, \{(1100)\}, \{(1111)\}\}$. B_4 has two parent blocks. Therefore, we need

Algorithm 2 Leaf-based optimisation

1: **procedure** LEAF_DETECT(G)
2: form an elementary block B by removing all the leaves of G;
3: $\mathcal{A}^B :=$ SCC_DETECT (B); $\Phi^B := \cup_{A^B \in \mathcal{A}^B} A^B$; *//detect attractors of B*
4: $\mathcal{T} :=$ transition system of G with state space restricted to $\mathcal{M}_G(\Phi^B)$;
5: $\mathcal{A} :=$ DETECT $(\mathcal{T})$;
6: **return** $\mathcal{A}$.
7: **end procedure**

to merge the two parent blocks to form a single parent block. Since the attractors of the merged block $B_{1,3}$ are the same as B_3, we directly obtain the attractors of $B_{1,3}$, i.e., $\mathcal{A}_{1,3} = \mathcal{A}_3 = \{\{(0*00)\}, \{(1100), (1111)\}\}$. There are three attractors so there will be three realisations for block B_4. The transition graph of the three realisations are shown in Figure 6. From the transition graphs, we easily get the attractors of B_4, i.e., $\mathcal{A}_4 = \{\{(0*0000)\}, \{(0*0001)\}, \{(110000)\}, \{(110011)\}, \{(111100)\}, \{(111111)\}\}$. Now the attractors for all the blocks have been detected. We can now obtain all the attractors of the BN by several cross operations. We start from the block with the largest credit, i.e., block B_4. The attractors of B_4 in fact cover blocks B_1, B_3 and B_4. The remaining block is B_2. We perform a cross operation on $\mathcal{A}_2$ and $\mathcal{A}_4$ and based on the obtained result we detect the attractors of the BN, i.e., $\mathcal{A} = D(\Pi(\mathcal{A}_2, \mathcal{A}_4) = \{\{(0 * 000000)\}, \{(0*000001)\}, \{(11010011)\}, \{(11010000)\}, \{(11011111)\}, \{(11011100)\}\}$.

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 with only one leaf node. Therefore, we treat leaf nodes in a special way. Formally, leaf nodes are recursively defined as follows.

Definition 13. *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 2 outlines the leaf-based decomposition approach for attractor detection. We now show that Algorithm 2 can identify all attractor states of a given BN.

Theorem 5. *Algorithm 2 correctly identifies all the attractor states of a given BN G.*

5 Experimental Results

We have implemented the decomposition algorithm presented in Section 4 in the model checker MCMAS [14]. In this section, we demonstrate the efficiency of our method by comparing our method with the state-of-the-art decomposition method mentioned in [8] which is also based on BDD implementation. We generate 33 random BN models with different number of nodes using the tool ASSA-PBN [15, 16] and compare the

model ID	# nodes	# non-leaves	# attractors	original models $t_{M_2}[s]$	original models $t_{M_1}[s]$	original models Speedup	models with leaves removed $t_{M_2}[s]$	models with leaves removed $t_{M_1}[s]$	models with leaves removed Speedup
1	100	7	32	4.56	0.86	5.3	0.58	0.02	29.0
2	120	9	1	18.13	0.95	19.1	1.10	0.04	27.5
3	150	19	2	201.22	1.66	121.2	0.74	0.02	37.0
4	200	6	16	268.69	7.04	38.2	0.97	0.02	48.5
5	250	25	12	533.57	11.16	47.8	0.90	0.04	22.5
6	300	88	1	–	–	N/A	238.96	65.33	3.7
7	450	43	8	–	60.82	N/A	3704.33	0.17	21790.2

Table 1: Selected results for the performance comparison of methods M_1 and M_2.

performance of the two methods on these 33 models. All the experiments are conducted on a computer with an Intel Xeon W3520@2.67GHz CPU and 12GB memory.

We name our proposed decomposition method M_1 and the one in [8] M_2. There are two possible implementations of the DETECT function used in Algorithm 1 as mentioned in [8]: 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 [8] uses similar leaf reduction technique, we make comparisons on both the original models and the models whose leaves are removed 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 that both methods fail to process. Among the other 22 cases, our method is faster than M_2 in 16 cases, which is approximately 73%. After removing leaf nodes, there are 5 cases that both methods fail to process. Among the other 28 cases, our method is faster than M_2 in 25 cases, which is approximately 89%. We demonstrate the results for 7 models in Table 1 and the remaining result can be found in [17]. Since our method considers the dependency relation between different blocks, the attractors of all the blocks need to be computed; while method M_2 can ignore the blocks with only leaf nodes. Therefore, the performance of our method is more affected by the leaf nodes. This is why the percentage that our method is faster than M_2 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_2. The "–" in Table 1 means the method fails to process the model within 3 hours. The speedup is therefore not applicable (N/A) for this result. The speedup is computed as t_{M_2}/t_{M_1}, where t_{M_1} is the time cost for M_1 and t_{M_2} is the time cost for M_2. All the time shown in Table 1 is in seconds. In general, we obtain a larger speedup when the number of attractors is relatively small. This is due to 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. Summarising, our new method shows a significant improvement on the state-of-the-art decomposition method.

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 totally different from the one in [9], which is designed for asynchronous networks. Secondly, our method considers the dependency relation among the sub-networks. The method in [8] does not rely on this relation and only takes the detected attractors in sub-networks to restrict the initial states when recovering the attractors for the original network. In this way, the decomposition method in [8] potentially cannot scale up very well for large networks, 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 [9] towards synchronous BNs. Experimental results show that our method is significantly faster than the one in [8]. Lastly, we have shown that the method proposed in [7] cannot compute correct results in certain cases.

Our current implementation is based on BDDs. One future work is to use SAT-solvers to implement the DETECT function as SAT-based methods are normally more efficient in terms of attractor detection for synchronous BNs [6].

References

1. Kauffman, S.: Homeostasis and differentiation in random genetic control networks. Nature **224** (1969) 177–178
2. Huang, S.: Genomics, complexity and drug discovery: insights from Boolean network models of cellular regulation. Pharmacogenomics **2**(3) (2001) 203–222
3. Somogyi, R., Greller, L.D.: The dynamics of molecular networks: applications to therapeutic discovery. Drug Discovery Today **6**(24) (2001) 1267–1277
4. Garg, A., Xenarios, L., Mendoza, L., DeMicheli, G.: 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. Volume 4453 of LNCS., Springer (2007) 62–76
5. Garg, A., Di Cara, A., Xenarios, I., Mendoza, L., De Micheli, G.: Synchronous versus asynchronous modeling of gene regulatory networks. Bioinformatics **24**(17) (2008) 1917–1925
6. Dubrova, E., Teslenko, M.: A SAT-based algorithm for finding attractors in synchronous Boolean networks. IEEE/ACM Transactions on Computational Biology and Bioinformatics **8**(5) (2011) 1393–1399
7. Guo, W., Yang, G., Wu, W., He, L., Sun, M.: A parallel attractor finding algorithm based on Boolean satisfiability for genetic regulatory networks. PLOS ONE **9**(4) (2014) e94258
8. Yuan, Q., Qu, H., , Pang, J., Mizera, A.: Improving BDD-based attractor detection for synchronous Boolean networks. Science China Information Sciences **59**(8) (2016) 080101
9. Mizera, A., Pang, J., Qu, H., Yuan, Q.: Taming asynchrony for attractor detection in large Boolean networks (technical report). Available online at `http://arxiv.org/abs/1704.06530` (2017)
10. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. Journal of Theoretical Biology **22**(3) (1969) 437–467
11. Shmulevich, I., Dougherty, E.R.: Probabilistic Boolean Networks: The Modeling and Control of Gene Regulatory Networks. SIAM Press (2010)
12. Lee, C.Y.: Representation of switching circuits by binary-decision programs. Bell System Technical Journal **38**(4) (1959) 985–999
13. Akers, S.B.: Binary decision diagrams. IEEE Transactions on Computers **100**(6) (1978) 509–516

14. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: An open-source model checker for the verification of multi-agent systems. International Journal on Software Tools for Technology Transfer (2015)
15. Mizera, A., Pang, J., Yuan, Q.: ASSA-PBN: a tool for approximate steady-state analysis of large probabilistic Boolean networks. In: Proc. 13th International Symposium on Automated Technology for Verification and Analysis. Volume 9364 of LNCS., Springer (2015) 214–220
16. Mizera, A., Pang, J., Yuan, Q.: ASSA-PBN 2.0: A software tool for probabilistic Boolean networks. In: Proc. 14th International Conference on Computational Methods in Systems Biology. Volume 9859 of LNCS., Springer (2016) 309–315
17. Mizera, A., Pang, J., Qu, H., Yuan, Q.: Benchmark Boolean networks. `http://satoss.uni.lu/software/ASSA-PBN/benchmark/attractor_syn.xlsx`