On the flora of asynchronous locally non-monotonic Boolean automata networks

Aurore Alcolei¹, Kévin Perrot¹, and Sylvain Sené¹,²

¹ Aix-Marseille Université, CNRS, LIF UMR 7279, 13288 Marseille, France (aurore.alcolei@ens-lyon.fr, {kevin.perrot, sylvain.sene}@lif.univ-mrs.fr)
² IXXI, Institut rhône-alpin des systèmes complexes, 69007 Lyon, France

Abstract. Studies on the dynamics of Boolean automata networks (abbreviated by BANs) have mainly focused on monotonic networks, where fundamental questions on the links relating their static and dynamical properties have been raised and addressed. This paper explores analogous questions on non-monotonic networks, ⊕-BANs (xor-BANs), that are BANs where all the local transition functions are ⊕-functions. Using algorithmic tools, we give a general characterization of the asynchronous transition graphs for most of the cactus ⊕-BANs and strongly connected ⊕-BANs. As an illustration of the results, we provide a complete description of the asynchronous dynamics of two particular classes of ⊕-BAN, namely ⊕-Flowers and ⊕-Cycle Chains. This work also leads to new bisimulation equivalences specific to ⊕-BANs.

Keywords: Interaction networks, Boolean automata networks, non-monotonicity, asynchronous dynamics.

1 Introduction

In the lines of [9], this study is a first step towards a better understanding of locally non-monotonic Boolean automata networks (BANs). It focuses on ⊕-BANs, that is, BANs in which the state of an automaton i is updated by xoring the state value (or the negated state value) of the incoming neighbours of i. In other words, in these BANs, every local transition functions is of the form $f_i = \bigoplus_{j \in N^+(i)} \sigma(x_j)$ where $\sigma \in \{id, neg\}$ and $N^+(i)$ denotes the set of incoming neighbours of i [10]. Following a constructive approach, we first looked at some particular BAN structures that combine cycles, such as the double-cycle graphs [2, 7], the flower-graphs [3] and the cycle chains. All these BANs belong to the family of cactus BANs since any two simple cycles in their structure have at most one automaton in common.

Happily we have realized that some of the specific results we got for each of these BANs can in fact be generalised to a wide set of ⊕-BANs: the strongly connected ⊕-BANs with an induced double cycle of size greater than 3.

A specification of these BANs is given in Section 2. This section also introduces all the definitions and notations that will be used in the sequel. Section 3 is dedicated to the presentation and proofs of the general results obtained about
the asynchronous dynamics of strongly connected ⊕-BANs with an induced double cycle of size greater than 3. Similarly to what is done in [7], these results are based on an algorithmic description of the asynchronous transition graph of these BANs. We conclude this paper in Section 4 with a full characterisation of two types of ⊕-BANs, the ⊕-flower BANs and the ⊕-cycle chain BANs, which illustrate the results of Section 3 and provides new bisimulation results specific to ⊕-BANs. In the following, proofs are often omitted or shortened, their full version are put in the appendix and left to the discretion of the reviewers.

2 Definitions and notations

Static definition of a BAN A Boolean automata network (BAN) is defined as a set of Boolean automata that interact with each other. The size of a network corresponds to the number of automata in it. For a network \( N \) of size \( n \) we denote \( V = \{1, \ldots, n\} \) the corresponding set of automata.

A Boolean automaton \( i \) is an automaton whose state has a Boolean value \( x_i \in \mathbb{B} = \{0, 1\} \). The Boolean vector \( x = (x_i)_{i=1}^n \) that gathers together the states of all automata in the network is called a configuration of \( N \). We will shorten by \( \pi \) the configuration \( x \) where the state of the \( i^{th} \) automaton is negated and similarly, for any subset \( I \) of \( V \), \( \pi^I \) will denote the configuration \( x \) where the states of the automata in \( I \) are negated.

The state of an automaton can be updated according to its local transition function \( f_i : \mathbb{B}^n \rightarrow \mathbb{B} \). This local function characterises how the automaton may react in a given configuration; just after being updated, the state of \( i \) has value \( f_i(x) \) where \( x \) is the configuration of the network before the update. We say that \( i \) is stable in \( x \) if \( f_i(x) = x_i \). It is unstable otherwise. Hence a network \( N \) is completely described by its set of local transition functions \( \mathcal{N} = \{f_i\}_{i=1}^n \).

An automaton \( i \) is said to be an influencer of an automaton \( j \) if there exists a configuration \( x \) such that \( f_j(x) \neq f_j(\pi) \). In this case \( j \) is said to be influenced by \( i \). We denote by \( I_j \) the set of influencers of \( j \).

In a BAN, a path \( \pi = i_0i_1\ldots i_k \) of length \( k \) is a sequence of distinct automata such that for all \( 1 \leq j \leq k, i_{j-1} \in I_j \). A BAN is strongly connected if there is a path between every two automata. A nude path is a particular path such that for all \( 1 \leq j \leq k, i_{j-1} \) is the unique influencer of \( i_j \) (\( I_j = \{i_{j-1}\} \)), i.e. \( f_j(x) = x_{j-1} \) or \( f_j(x) = x_{j-1} \). We define the sign of a nude path as the parity of the number of local functions of the form \( f_i(x) = x_{i-1} \) that compose it, i.e. \( \text{sign}(\pi) = \left( \sum_{j=1}^n 1_{f_j(x) = x_{j-1}} \right) \text{ mod } 2 \). A nude path is maximal if any extension of it is not a nude path. We will denote by \( \pi_i \) the maximal nude path that ends in automaton \( i \). Paths and nude paths get their name from the graphical representation that is often associated to BAN as we will see next.

To get a sense of what a network looks like, it is common to give a graphical representation of it. To every local functions \( f_i \), one can associate a Boolean formula \( \mathcal{F}_i \) over the variables \( x_i \). The literal associated to the \( k^{th} \) occurrence of the variable \( x_i \) is denoted by \( \sigma_k(x_i) \) where \( \sigma_k \) is the sign of the literal. Then the interaction graph of \( N \) according to these formulas is the signed directed
graph $G = (V, A)$, where $V = \{1, \ldots, n\}$ is the set of nodes of $G$ with one entry points per literal in $F_j$, and $A$ is the set of arcs defined by $(i, j, \sigma_k) \in A$ if the $k^{th}$ occurrence of the variable $x_i$ in $F_j$ has sign $\sigma_k$ (see Figure 1 (a)).

As we focus on $\oplus$-BANs, all formula $F_i$ involving more than one automaton will be written in Reed-Muller canonical form, that is $F_i = \bigoplus_{j \in I_i} \sigma_j(x_j)$. The type of a BAN will refer to the underlying structure of its interaction graph (modulo the sign of the literals and a renaming of the automata). A type of BANs can be described by a family of graphs, and we will say that two BANs (modulo the sign of the literals and a renaming of the automata) are of the same type if their interaction graphs are isomorphic (we ignore the labels).

The simplest interaction structure that allows for complex behaviour is the cycle structure [11]. A Boolean automata cycle (BAC) $C$ of size $n$ is a BAN defined as a set of local functions $\{f_i\}_{i=1}^n$ such that $f_i(x) = x_{i(i-1) \mod n}$ or $f_i(x) = x_{i(i-1) \mod n}^{-1}$ for all $i \in \{1, \ldots, n\}$. Abusing notation we will often express $f_i$ via its formula representation $F_i = \sigma_i(x_{ \text{pred}(i) })$ where $\text{pred}(i) = (i - 1 \mod n)$ is the only influencer of $i$ in $C$ and $\sigma_i$ is its sign (either the identity or the negation function).

In the following, the majority of the networks or patterns we discuss are made of cycles that intersect each other. If an automaton $i$ is the intersection of $\ell$ distinct cycles, then its local transition function will be $f_i(x) = \bigoplus_{j=1}^\ell \sigma_j(\text{pred}_j(i))$ where $\text{pred}_j(i)$ represents the predecessor of $i$ in each of the incident cycles.

If a BAN is described in terms of intersections of $m$ simple cycles, $C_1, \ldots, C_m$, we will often represent its size by a vector of natural numbers $n = (n_1, \ldots, n_m)$, where $n_k$ is the size of the $k^{th}$ cycle. We will also use this vector representation to describe the configurations of the BAN: $x = (x^1, \ldots, x^m) \in \mathbb{B}^{n_1} \times \cdots \times \mathbb{B}^{n_m}$ will represent the configuration where each cycle $C_k$ is in configuration $x^k \in \mathbb{B}^{n_k}$. By extension $x^k_j$ will denote the state of automaton $i_j^k$ which is the $j^{th}$ automaton of cycle $C_k$.

As one can expect, a strongly connected $\oplus$-BAN is a $\oplus$-BAN whose interaction graph is strongly connected. Hence the type of these BANs can always be described as a set of simple cycles and intersection automata. Strongly connected cactus BANs are special strongly connected BANs where any two simple cycles intersect each other at most once [4]. The simplest example of BANs of this form are the $\oplus$-Boolean automata double-cycles ($\oplus$-BADCs). These $\oplus$-BANs are described by two cycles $C_1$, $C_2$ that intersect at a unique automaton $o = i_1^1 = i_1^2$. The $\oplus$-BAN depicted in Figure 1 (a) is in fact a $\oplus$-BADC of size $(2, 1) = 2 + 1 - 1 = 2$.

Asynchronous dynamics of a BAN As previously mentioned, the configuration of a network may change in time along with the local updates that are happening. A local update is formally described by a subset $W$ of $V$ which contains the automata to be updated at a time. We say that $W$ is asynchronous if it has cardinality 1, that is, $W = \{i\}$ for some $i \in V$.

An update $W$ makes the system move from a configuration $x$ to a configuration $x'$ where $x'_i = f_i(x)$ if $i \in W$, and $x'_i = x_i$ otherwise. This defines a global function $F_W : \mathbb{B}^n \rightarrow \mathbb{B}^n$ over the set of configurations.
A network evolves according to a particular mode $M \subseteq \mathcal{P}(V)$ if all its moves are due to updates from $M$. The asynchronous mode of a BAN of size $n$ is then defined by the set $A = \{\{i\}\}_{i=1}^{n}$ of asynchronous updates, it is non-deterministic. Note that our definition of update mode is not fully general [8] but sufficient for the scope of this paper.

We say that a configuration $x'$ is reachable from a configuration $x$ (in a mode $M$) if there exists a finite sequence of updates $(W_i)_{i=1}^{s}$ (in $M$) such that $F_{W_s} \circ \ldots \circ F_{W_1}(x) = x'$. Then, a configuration is unreachable (in $M$) if it cannot be reached from any other configuration but itself (in $M$). Finally a fixed point (of $M$) is a configuration $x$ such that $F_W(x) = x$ for every update $W$ (in $M$).

The study of the dynamics of a network under a particular update mode aims at making predictions, i.e. given an initial configuration $x$, we want to tell what are the possible sets of configurations in which the network can end asymptotically. These sets are called attractors of the network and the set of configurations from which they can be reached are their attraction basins. Notice that a fixed point is an attractor of size 1.

The dynamics of a network $\mathcal{N}$ according to an update mode $M$ can be modeled by a labeled directed graph $G_M^{\mathcal{N}} = (\mathbb{B}^n, \bigcup_{W \in M} F_W)$, called the $M$-transition graph of $\mathcal{N}$, such that:

- the set of vertices $\mathbb{B}^n$ corresponds to the $2^n$ configurations of $\mathcal{N}$.
- the arcs are defined by the transition graph of the functions $F_W$ for all $W \in M$, that is, $x \xrightarrow{W} x'$ is an arc of $G$ if and only if $W \in M$ and $F_W(x) = x'$.

The transition graph $G_A^{\mathcal{N}}$, associated to the asynchronous update mode is called the asynchronous transition graph of $\mathcal{G}$, shorten ATG. Figure 1 (b) shows the ATG of the $\oplus$-BADC depicted on the left.

In terms of transition graph, an attractor of $\mathcal{N}$ for the mode $M$ corresponds to a terminal strongly connected component of $G_M^{\mathcal{N}}$, that is, a strongly connected component that does not admit any outgoing arcs. The attraction basin of an attractor corresponds to the set of configurations in $G_M^{\mathcal{N}}$ that are connected to this component. Conversely, the unreachable configurations of $M$ are the configurations that do not have any incoming arcs but self-loops in $G_M^{\mathcal{N}}$. 

---

**Fig. 1.** (a) The interaction graph of BAN $\{f_1(x) = x_2, f_2(x) = x_1 + x_2\}$ and (b) its asynchronous transition graph.
To this extent, most of the results presented in the following are expressed in terms of walks and descriptions of the asynchronous transition graphs of the networks we study.

**Bisimulation equivalence relation** We conclude this section with a quick reminder on bisimulation which is an equivalence relation over the set of BANs that expresses the fact that two networks “behaves the same way” (up to a renaming of their automata and/or of their configurations). More precisely, the equivalence of $\mathcal{N}$ and $\mathcal{N}'$ means that, for any update mode $M$, the transition graphs $G^M_\mathcal{N}$ and $G^M_\mathcal{N}'$ are isomorphic.

**Definition 1.** Two BANs $\mathcal{N}$ and $\mathcal{N}'$ bisimulate each other if there exist two bijections $\varphi : V \to V'$ over the set of automata and $\phi : \mathcal{B}^n \to \mathcal{B}'^n$ over the set of configurations such that for any update $W \subseteq V$ in $\mathcal{N}$, the corresponding update $\varphi(W)$ acts the same way in $\mathcal{N}'$, that is, for all configurations $x$, $\phi(F_W(x)) = F'_{\varphi(W)}(\phi(x))$.

This definition of bisimulation for BANs has been introduced in [8]. We recall here some general results about it.

**Theorem 1 ([8]).** Let $\mathcal{N} = \{f_i\}_{i=1}^n$ be a BAN and $\mathcal{N}' = \{f'_i\}_{i=1}^n$ be its dual network defined as $f'_i(x) = f_i(x)$ then $\mathcal{N}$ and $\mathcal{N}'$ bisimulate each other.

**Theorem 2 ([8]).** Let $\mathcal{N} = \{f_i\}_{i=1}^n$ be a BAN and $\mathcal{N}' = \{f_i^+\}_{i=1}^n$ be its canonical network defined as (i) $f_i^+(x) = x_i$ if $f_i(x) = x_i$ or $x_i^-$, and (ii) $f_i^+(x) = f_i(x_i^-)$ otherwise, where $I = \{i \in V \mid \text{sign}(\pi_i) = 1\}$ is the set of automata whose maximal incoming nude path has negative sign. Then $\mathcal{N}$ and $\mathcal{N}'$ bisimulate each other.

Theorem 1 is of importance because it tells us that all the results stated in the sequel will also hold for $\Leftrightarrow$-BANs, which are the dual BANs of the $\oplus$-BANs since all their local functions are of the form $f_i(x) = \Leftrightarrow_{j \in I_i} \sigma(x_j)$. Furthermore, Theorem 2 is very useful when studying particular types of networks because it reduces a lot the number of cases to study. Indeed, it says that one only needs to focus on networks with positive nude paths to characterise the whole set of possible transition graphs BANs. For example, it states that there are only three different cases of $\oplus$-BADCs to study: the positive ones, the negative ones and the mixed ones, that respectively correspond to the case where $f_o(x) = x^1 \oplus x^2$ and $f_o(x) = x^1 \oplus x^2$. There is actually only one class of $\oplus$-BADCs since: (i) the equality $x^1 \oplus x^2 = x^1 \oplus x^2$ implies that positive and negative $\oplus$-BADCs are trivially equivalent; (ii) a positive $\oplus$-BADC bisimulates a mixed $\oplus$-BADC of same structure by taking $\phi(x) = x^V$.

To prove a bisimulation relation between two networks we will often use a stronger condition than the one given in Definition 1.

**Lemma 1.** Two BANs $\mathcal{N} = \{f_i\}_{i=1}^n$, $\mathcal{N}' = \{f'_i\}_{i=1}^n$ bisimulate each other if there exists a bijection $\varphi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ and a set $\{\phi_i : \mathcal{B} \to \mathcal{B}'\}_{i=1}^n$ of
(non constant) Boolean functions such that for all automata $i$, $\phi_i \in \{\text{id}, \text{neg}\}$, and for all configurations $x \in \mathbb{B}^n$, $\phi_i(f_i(x)) = f'_{\phi(i)}(\phi(x))$ where $\phi(x)$ is defined componentwise by $\phi(x)_i = \phi_{\varphi^{-1}(i)}(x_{\varphi^{-1}(i)})$.

**Proof.** The proof is straightforward since the equality $\phi_i(f_i(x)) = f'_{\phi(i)}(\phi(x))$ between the local functions induces the equality $\phi(F_W(x)) = F'_{\phi(W)}(\phi(x))$ between the global functions for any update $W$. \hfill \Box

We will make great use of Theorem 2 and Lemma 1 in Section 4, when we will give new bisimulation results specific to $\oplus$-BANs.

## 3 General results on $\oplus$-BANs

This section presents the main theorem of this paper: a connexity result that characterises the shape of the ATG of any strongly connected $\oplus$-BAN with an induced BADC of size greater than 3.

**Theorem 3.** In a strongly connected $\oplus$-BAN with an induced BADC of size greater than 3, any configuration which is not stable in a quadratic number of asynchronous updates.

This theorem tells us that the ATG of any strongly connected $\oplus$-BAN which is not a cycle or a clique is characterised by (see Figure 2):

- its fixed point(s) $S$ (if any).
- its unreachable configuration(s) $U$ (if any).
- a unique strongly connected component reachable from any configuration of $U \setminus S$ and connected to any configuration of $S \setminus U$.

We start this section with some general remarks and results about the set of fixed points and unreachable configurations of any ($\oplus$-)BANs. These remarks will help us describe precisely the general form of the ATGs, and will serve as preliminaries to the proof of Theorem 3 that we will give in the end of this section.

### 3.1 Fixed points and unreachable configurations

According to the definition, a configuration $x$ is a fixed point for a mode if it has no outgoing arcs but self-loops in the transition graph associated to this mode. In the asynchronous update mode this means that for all $i$ in $V$, $f_i(x) = x_i$. Hence, in a fixed point, the state of the automata along a nude path is completely determined by the head of this nude path. This leads to the following bound on the number of possible fixed points, that is related to the set of work $[1,5,6]$.

**Lemma 2.** In any BAN $\mathcal{N}$, the maximum number of fixed points in the asynchronous mode $A$ is $2^k$, where $k$ is the number of automata $i$ such that $\pi_i$ is of length 0 (i.e. $i$ is an “intersection node” in some interaction graph of $\mathcal{N}$).
Fig. 2. General ATG shape of strongly connected $\oplus$-BANs with an induced BADC of size greater than 3.

Proof. It is enough to note that a configuration $x$ is stable in $A$ only if every automata along a nude path share the same state value in $x$. In other words, $x$ is completely determined by the states of the intersection nodes of $\mathcal{N}$.

This bound is rough and we believe that it is possible to lower it for subclasses of networks. However, if we define the contraction of a network to be the network obtained by removing any automaton $i$ whose incoming maximal nude path $\pi_i$ has length greater than 1 and replacing the variable $x_i$ by the variable associated to the head of $\pi_i$ in the remaining local functions, then any BAN whose contraction results in the trivial network $\{f_i(x) = x_i\}_{i \in V}$ reaches the bound of $2^k$ fixed points.

Also, notice that in the asynchronous mode, the unreachable configurations of a network $\mathcal{N} = \{f_i\}_{i=1}^n$ are exactly the fixed points of the reverse network $\mathcal{N}^R = \{f_i^R\}_{i=1}^n$ defined by $f_i^R(x) = f_i(\overline{x})$. $\mathcal{N}$ and $\mathcal{N}^R$ are of the same type hence the maximum number of fixed points for the type of $\mathcal{N}$ will also be its maximum number of unreachable configurations. Moreover, this implies that if all the networks of a given type pertain to the same bisimulation class then the number of unreachable configurations and the number of fixed points will be equal.

3.2 Proof of Theorem 3

Let $\mathcal{N}$ be a strongly connected $\oplus$-BAN with an induced BADC $B$ of size greater than 3, let $x$ be its initial (unstable) configuration, and let $x'$ be the configuration to reach. The idea behind the proof of Theorem 3 is to take advantage of the high expressiveness of $\oplus$-BADCs and to use $B$ as a “state generator” that sends information across the network in order to set up the state of every automata of $\mathcal{N}$ to their value in $x'$. More precisely, the proof of Theorem 3 is based on the following two lemmas.

Lemma 3. In a $\oplus$-BADC, every configuration which is not unreachable can be reached from any other (unstable) configuration in $O(n^2)$ (and the bound is tight).
Proof. First let us recall that all \( \ominus \)-BADCs of same size \((n_1, n_2)\) are equivalent with respect to bisimulation. This means in particular that their ATGs are isomorphic and so proving that Lemma 3 holds for one \( \ominus \)-BADC of each size is enough to prove Lemma 3 completely. Hence in the following we only deal with positive \( \ominus \)-BADC. However, one will notice that the proof below is easy to adjust to any \( \ominus \)-BADC.

The proof presents an algorithm that explains how to walk from one configuration to another in the ATG of any positive \( \ominus \)-BADC that has at least one cycle of size greater than 3. The algorithm can be tuned to deal with BADCs where \( n_1 \) and \( n_2 \) are both less than or equal to 2 but this multiplies the number of cases that need to be considered and masks the general dynamics. So for the special BADCs of size \((n_1, n_2) = (1, 2)\) (or vice-versa) and \((n_1, n_2) = (2, 2)\) we prefer to prove Lemma 3 by looking directly at the form of their ATG. These ATGs are drawn in Figure 3 and they all satisfy Lemma 3 as desired.

Now, without loss of generality we assume that \( n_1 \geq 3 \).

1. From any configuration with at least one automata in state 1 (i.e. unstable in the case of positive \( \ominus \)-BADC) one can reach a configuration \( x \) where \( x_o = f_o(x) \) and \( x_i = f_i(x) \) for all \( i \neq o \) (i.e. \( x_o = x_{n_1} \oplus x_{n_2} \) and \( x_j^o = x_{j-1}^o \) for all \( i_j^o \neq o \)). This is possible for example using the following steps:

   - In a linear number of updates, set \( x_{n_1} \) to 1 and \( x_{n_2} \) to 0: Let \( i_{j}^o \) be the automaton in state 1 that is the closest to \( i_{n_1} \), and update every automata on the path from \( i_{j}^o \) to \( i_{n_1} \). If \( k = 1 \) then this simply propagates the state 1 of automaton \( j \) on every automata up to automaton \( n_1 \) in \( C_1 \); if \( k = 2 \) then the state 1 of \( i_j^o \) propagates from \( j \) to \( n_2 \) in \( C_2 \) then from \( 1 \) to \( n_1 \) in \( C_1 \). The more subtle point is that by the time \( o = i_{1}^o \) is updated, we have \( x_{n_1} = 0 \) and \( x_{n_2} = 1 \) which gives \( f_o(x) = 1 \) as claimed. Hence these first updates set \( i_{n_1} \), to 1. To finish, if \( x_{n_2} \neq 0 \) (hence \( x_{n_2} = 1 \)) update all the automata of \( C_2 \) from 1 (i.e. \( o \)) to \( n_2 \).

   - In a quadratic number of updates, set \( C_1 \) into the alternating configuration such that \( x_{n_1} = 1 \), i.e. to 11(01)\(n_1/2-1\) if \( n_1 \) is even and to 0(01)\((n_1-1)/2\) if \( n_1 \) is odd:

     for \( j = n_1 \) to 2 do: update the automata of \( C_1 \) from 1 to \( j \) then the ones of \( C_2 \) from 2 to \( n_2 \). The invariant is the following: after each iteration, \( x^1[n_1, j] = (10)^{(n_1-j)/2} \) and \( x^1[n_1+1, j] = x_{n_1}^o \oplus x_{n_2}^o = 1 \oplus x_{j}^1 = x_{j+1}^1 \). Indeed we start with \( x_{n_1} = 1 \) and \( x_{n_2} = 0 \) so by the end of the first iteration \( x_{n_1} = x_{n_2} = x_o = 1 \oplus 0 = 1 \). Then for the \( j^{th} \) iteration, since we start with \( x^1[n_1, j+1] = (10)^{(n_1-j-1)/2} \) and with \( f_o(x) = x_{j+1} \) we end up with \( x^1[n_1, j+1] = x_{n_2} = x_{j+1} \) and so \( x^1[n_1, j] = (10)^{(n_1-j)/2} \).

   - Similarly force \( C_2 \) to alternate in a quadratic number of updates (while preserving the alternating configuration in \( C_1 \)):

     for \( j = n_2 - 1 \) to 2 do: update the automata of \( C_2 \) from 1 to \( j \) then the one of \( C_1 \) from \( n_1 \) to \( 2 \).

     The invariant is: after each iteration, \( x_{n_2} \) is unchanged, \( x^1_2 = x^2_2 = x_o \), \( f_o(x) \neq x_o \) and \( x^1[2, n_2] \) and \( x^2[n_2, j] \) are both alternating. The first two
Fig. 3. The ATGs of the positive BADCs of size (1, 2) (top) and (2, 2) (bottom).
2. Let \( x \) which is in \( \sum \). This bound is tight since going from the configuration \( \hat{x} \) with at least one automaton \( i \) in one of the following configurations:
- for \( j = n_1 \) to \( 2 \) (in \( C_1 \)): update the automaton \( i^1 \) if \( \hat{x}^1 \neq x^1 \);
- for \( j = n_2 \) to \( 2 \) (in \( C_2 \)): update the automaton \( i^2 \) if \( \hat{x}^2 \neq x^2 \);
- update the automaton \( i^k \).
These updates are efficient since for all \( i \in \{ i^1, o \} \), if \( \hat{x} \neq x' \) then \( x' = f_i(\hat{x}) \), which is the value returned by the update of \( i \). Then, by definition of \( \hat{x} \), automaton \( o \) already has the right state. And, finally, by definition of \( i^k \), \( x^k = f_i(x') \), which is the value returned by \( f_i \) after every other automaton has been updated.

The second sequence takes a linear number of steps, so the whole sequence remains quadratic. This bound is tight since going from the configuration \( x = (10^{n_1-1}, 01^{n_2-1}) \) to a configuration \( x' \) where \( x'_i = f_i(x') \) for all automata \( i \neq o \) (for example the configuration \( x' = (0(10)^{n_1-1}, 01^{n_2-1}) \) if \( m \) and \( n \) are odd) requires at least \( \sum_{j=1}^{n_1} j + \sum_{j=1}^{n_2} j = n_1(n_1-1) + n_2(n_2-1) \) updates, which is in \( \theta((n_1 + n_2)^2) \).  

\[\Box\]
Remark 1. Note that if we allow synchronous transitions, then every configuration is reachable from any unstable configuration. By the lemma above this is immediate if the target configuration is not unreachable, but the algorithm also tells us that if $x$ is unreachable, one can still reach the configuration $x' = \overline{x}$ for $|C_1| > 1$ (since in that case the state of the first automaton of $C_{1-i}$ is stable).

Then for all automaton $j$ of $C_i$, $f_j(x') = \overline{x}_j = x_j$, so the synchronous update of $C_i$ changes the configuration of the system from $x'$ to $x$.

Lemma 4. In a $\oplus$-BAN $\mathcal{N}$, if $i$ and $j$ are two automata such that there is a path from $i$ to $j$, then for any configuration $x$ such that $i$ is unstable in $x$ there exists a configuration $x'$ reachable from $x$ such that $j$ is unstable in $x'$.

Proof. This result is based on the fact that, in a $\oplus$-BAN, making a stable automaton become unstable can be achieved by only switching the state of one of its incoming neighbours. Indeed, for an automaton $i$ stable in $x$ we have $x_i = f_i(x) = \bigoplus_{j \in I_i} \sigma_j(x_j)$, so switching the state of one of its neighbours $k \in I_i \setminus \{i\}$ leads to a configuration $x' = \overline{x}$ such that $x'_i = x_i = f_i(x) = \bigoplus_{j \in I_i} \sigma_j(x_j) = \overline{x}_k \oplus \left( \bigoplus_{j \in I_i \setminus \{k\}} \sigma_j(x_j) \right)$, that is, $i$ is unstable in $x'$.

So let $i$ and $j$ be two automata as described in Lemma 4, let $p = i_0, i_1, \ldots, i_k$ be a shortest path (in the interaction graph of $\mathcal{N}$) from $i = i_0$ to $j = i_k$ and let $i_\ell$ denotes the last automaton in $p$ that is unstable. Then updating along $p$ from $i_\ell$ to $i_{k-1}$ (so that nothing happens if $\ell = k$, i.e. if $j$ is unstable) will lead to a configuration where $j$ is unstable. This is quite immediate from the remark above. The only subtlety is the choice of the path which must ensure that the update of one automaton only affects the next automaton on the path but not the ones after it. This is true in particular if one take $p$ to be a shortest path since this ensures that for all automata $i_\ell, i_\ell' \in p$, there are no arcs from $i_\ell$ to $i_\ell'$ if $\ell + 1 < \ell'$.

Theorem 3 states that, in a strongly connected $\oplus$-BAN with an induced BADC of size greater than 3, any configuration that is not unreachable can be reached from any configuration which is not stable in a quadratic number of asynchronous updates. Let us now give its proof on the basis of the two previous lemmas.

Proof. Let $B$ be an induced BADC of size greater than 3 in the BAN $\mathcal{N}$ and let $x$ and $x'$ respectively be the initial configuration and the target configuration described in Theorem 3. The configuration $x$ is not stable so, by Lemma 4, it is possible to go from $x$ to a configuration $y$ where one automaton of $B$, hence $B$, is not stable. Then, using Lemmas 4 and 3, we claim that it is possible to set the state of every automata $i$ outside of $B$ to its value in $x'$ while keeping $B$ in an unstable configuration.

The idea is as follows: let $i$ be an automaton that is not in $B$ and let $p = i_0, i_1, \ldots, i_k$ be a shortest path (in the interaction graph of $\mathcal{N}$) from $B$ to $i_k = i$. Then, applying the algorithm from Lemma 3, we know how to reach a
configuration where $i_0$ is unstable and so, using the algorithm from Lemma 4, we know how to reach a configuration where $i$ is unstable. From this configuration we can set the state of $i$ to $x'_i$ by updating $i$ if necessary. So, if we can guarantee that this process preserves the instability in $B$, then we can use it repetitively on the automata outside of $B$ to reach a configuration where $B$ is unstable and where all automata outside of $B$ are in the state specified by $x'$. Once this is done we only need to set $B$ to its right value to reach $x'$. Since $B$ is unstable, this can be done by using the algorithm from Lemma 3, assuming that the restriction of $x'$ to $B$ is not unreachable for $B$ ($B$ is viewed as a $\oplus$-BADC whose local transition functions are fixed by its surrounding environment in $x'$). If this is not the case we use the same kind of trick that what is done in the second step of the proof of Lemma 3 when the stable state of the target configuration is not the central node $o$: if $i$ is an automaton of $N$ such that $f_i(x') = x'_i$, and if $p = i_0 \ldots i_k$ is a shortest path from $i = i_0$ to $B$, then we first reach the configuration $\hat{x}$ such that (i) $\hat{x}_j = x'_j$ if $j \notin p$, (ii) $i_k(\in B)$ is stable in $\hat{x}$ (so the restriction of $\hat{x}$ to $B$ is reachable for $B$), and (iii) the state values of the automata in $p$ are “alternating”, i.e. if we set up the state of the automata of $p$ to their value in $x'$ from $i_k$ to $i_1$ then every time an automaton $i_j$ is about to be set up, its predecessor in $p$ must be in an unstable state so as to enable $\ell$ to switch state if necessary. With such conditions it is easy to go from $\hat{x}$ to $x'$: one only needs to update $p$ back up as described in the previous sentence; then if $i_0$ is not already in state $x'_{i_0}$, it can still be switched to the right state since $f_i(x') = x'_i$.

The configuration $\hat{x}$ described above can be computed inductively by taking the $k^{th}$ iteration, $\hat{x}^k$, of: (i) $\hat{x}^0 = x'$, (ii) $\hat{x}^j = y^j$ if $j \notin \{i_{k-1}, i_k\}$, (iii) $i_k(\in B)$ is stable in $\hat{x}$, and $\hat{x}^j$ is the solution of the equation $f_i(\hat{x}^j) = \hat{x}^j$, and $x'^j = x'^j$. Actually, setting the automata outside of $B$ to their state in $x'$ cannot be done in any order. Indeed, the algorithm from Lemma 4 often requires to switch the state of some automata outside of $B$. Hence we need to guarantee that the automata that have already been treated are not switched again while processing the other automata. A way to ensure that is to compute a breadth first search tree of root $B$ and to treat the automata in the order given by the tree from the leaves to the root, using the branches of the tree as the paths from $B$ to the automata to be treated. An example of such ordering is given in the picture below.

Finally, to conclude the proof above, we need to precise a way of using the algorithm from Lemma 4 that ensures that the instability of $B$ as well as the state of the automata that are not in $B$ or on the path from $B$ to the automaton to be set up, are preserved by the updates. So let $x$ be the starting configuration, let $p$ be the (shortest) path from $B$ to the automata to be set up, and let $j \neq i_0$ be an influencer of $i_0$ in $B$ (i.e. $j \in (B \setminus \{i_0\}) \cap I_{i_0}$). Then, since $B$ is supposed to be unstable in $x$, one can use Lemma 3 to put the system in a configuration $y$ where $i_0$ is unstable, and where $y_j$ is such that $y_j = f_j(y^i)$ if there is an arc from $i_1$ to $i_0$, and $y_j = f_j(y^{i_0 \rightarrow i_1})$ if there are no arcs from $i_1$ to $i_0$. This
is possible since $B$ is of size at least 3, and so one can ask a third automaton of $B$ (different from $i_0$ and $j$) to be stable in $y$, which makes $y$ reachable. The algorithm does not modify the state of the automata outside of $B$.

From there one can start applying Lemma 4: let $i_\ell$ be the last automaton in $p$ that is unstable, then, if $\ell \leq 1$, start updating $p$ from $i_\ell$ to $i_1$. This leaves $N$ in a configuration $y'$ such that $B$ is unstable. Indeed,

- either nothing happened ($\ell > 1$) and so $B$ is still unstable (because $i_0$ is unstable in $y$ for example).
- or only $i_1$ has been updated and so: (i) if there is an arc between $i_1$ and $i_0$, then $y'_j = y_j = f_j(y^{i_1\ldots i_\ell}) = f_j(y')$ and so $j$ is unstable in $y'$; (ii) if there are no arcs from $i_1$ to $i_0$ then the neighbourhood of $i_0$ has not changed so $i_0$ is still unstable in $y'$.
- or $i_0$ and $i_1$ have been updated and so: (i) if $i_0$ has no self loop and there is an arc from $i_1$ to $i_0$ then $i_0$ is still unstable (because it has changed and an odd number of its incoming neighbours have changed too); (ii) if there are no arcs from $i_1$ to $i_0$ then $y'_j = y_j = f_j(y^{i_1\ldots i_\ell}) = f_j(y')$ and so $j$ is unstable in $y'$; (iii) if $i_0$ has a self loop then $i_0$ is not an influencer of $j$ (because $B$ is an induced BADC of size 3 and $j$ has been chosen to be the predecessor of $i_0$ different from $i_0$) so $f_j(y^{i_1\ldots i_\ell}) = f_j(y^{i_0\ldots i_\ell})$, so $y'_j = f_j(y^{i_0\ldots i_\ell})$ which means as previously that $j$ is unstable in $y'$.

Now, let $\ell' = \max(2, \ell) - \ell'$ is the last automaton of $p$ to be unstable in $y'$—then, since $B$ is unstable in $y'$, we can use Lemma 3 again to reach a configuration $y''$ such that $y''_{i_0} = f_{i_0}(y^{i_1\ldots i_\ell})$ and $y''_i = y'_i$ for all automata $i$ that are not in $B$. Moreover, since $p$ was chosen to be a shortest path, no automaton in $B$ influences the automata of index greater than 2 in $p$. So the last automaton of $p$ that is unstable $y''$ is $i_{\ell'}$ too. Hence we can finish running the algorithm of Lemma 4 (by updating the automata along $p$ from $i_{\ell'}$ to $i_{n-1}$) and be sure that this leads to a configuration where $i_{n-1}$ is unstable. We also know that in this configuration $B$ is unstable since $i_0$ has state $f_{i_0}(y^{i_1\ldots i_{n-1}})$. This last algorithm concludes the proof of Theorem 3. ☐
This algorithm described above is quadratic in the worst case. However, its complexity highly depends on the structure of the network and/or the final configuration \(x'.\) For example, if every automaton in \(\mathcal{N}\) is at constant distance from an induced BADC of size greater than 3, then this algorithm becomes linear in \(n.\) Similarly, since the number of passes that are needed along a path depends on the number of alternating states (i.e. 01 or 10 patterns) along this path in \(x',\) then if this number is less than a constant in any path the algorithm will also run in linear time. Finally we need to insist on the fact that this algorithm does not always provide the most efficient sequence of updates (for example it does not take into account the starting configuration) hence the complexity of this algorithm is only an upper bound on the length of the shortest path between two configurations. Let us notice that this bound might nevertheless be reached, as when one move from configuration \(10^{n-1}\) to configuration \((10)^{n/2}\) in a positive \(\oplus\)-BADC of size \(n\) (these considerations on 01 patterns are similar to the notion of expressiveness defined on the monotonic case in [7]).

4 Study of some specific \(\oplus\)-BANs

We now give a complete characterization of two specific types of \(\oplus\)-BAN: the \(\oplus\)-BA Flowers and the \(\oplus\)-BAC Chains. For each of these two types of BANs, we describe their bisimulation classes and give their number of fixed points and unstable configurations. This illustrates the results of Section 3, and introduces new bisimulations that are general enough to be used in the study of other types of \(\oplus\)-BAN.

4.1 \(\oplus\)-BA Flowers

A \(\oplus\)-BA Flower (\(\oplus\)-BAF) with \(m\) petals is defined as a set of \(m\) cycles that intersect at a unique automaton \(o = i_1 = \ldots = i_k\) (\(\oplus\)-BADC correspond to the case \(m = 2\)). There are at most two bisimulation classes for a given type of flower (i.e. for a given number of petals \(m\) and size \((n_1, \ldots, n_m))\).

Lemma 5. The set of \(\oplus\)-BAF with \(m\) petals of size \((n_1, \ldots, n_m)\) admits one bisimulation class if \(m\) is even and two if \(m\) is odd.

Proof. Similarly to what is done in Section 2 for the \(\oplus\)-BADCs, we restrict our study to the canonical \(\oplus\)-BAFs, that are the \(\oplus\)-BAFs such that the only negative literals are in the local function of \(o\) (Theorem 2). Then, because of the identity \(b_1 \oplus b_2 = \overline{b_1} \oplus \overline{b_2}\) for all Boolean values \(b_1\) and \(b_2\), the sign of any pair of negative literals cancel in \(f_o\), and so there are at most two equivalence classes: the positive one, that have only positive literals because all negative literals cancel (i.e. there is an even number of negative paths in the original BAF), and the negative one that have exactly one negative literal in \(f_o\) (i.e. there is an odd number of negative paths in the original BAF). In the case where \(m\) is even, the bijection \(\phi(x) = x^V\) over the set of configurations defines an isomorphism between the ATGs of the negative and the positive \(\oplus\)-BAF of
same type, therefore the negative and positive classes coincide. In the case where $m$ is odd, the two classes are distinct since, in particular, they do not have the same number of fixed points, as this is shown in Lemma 6.

\[\square\]

**Lemma 6.** A positive $\oplus$-BAF with $m$ petals has a unique stable configuration, $0^n$, if $m$ is even and two stable configurations, $0^n$ and $1^n$, if $m$ is odd. A negative $\oplus$-BAF (with an odd number of petals) does not have any fixed point.

**Proof.** There are several ways to compute the fixed points of a $\oplus$-network. One way is to fix the state of one automaton and propagate the information that this choice implies on the state of the other automata in the network, making new choices when necessary, until having completely fixed the configuration or until reaching a contradiction. For example, in a positive $\oplus$-BAF $F$ with an even number of petals, any configuration $x$ that contains an automaton $i$ in state 1 is unstable. Indeed suppose for the sake of contradiction that $x$ is stable, then $o$, and so every automata in $F$, are in state 1 (because updates for $o$ to $i$ lead to $x_i = x_o$), so $x = 1^n$. But $1^n$ is not stable since $f_o(x) = \bigoplus_{k=1}^m 1 = 0$. Similarly we prove that in a negative $\oplus$-BAF with an odd number of petals, if a configuration contains an automaton in state 0, respectively an automaton in state 1, then it cannot be stable, and so the network has no fixed points.

\[\square\]

The results above enable us to fully characterise the $\oplus$-BAFs of a given type:

- if $F$ is a $\oplus$-BAF with an even number of petals then $F$ and its reverse network $F^R$ both have ATGs isomorphic to the ATG of the positive $\oplus$-BAF, consequently $G^F_A$ has exactly one unreachable configuration, one fixed point, and one SCC of size $2^n - 2$.
- if $F$ is a $\oplus$-BAF with an odd number of petals then the ATG of $F$ can have four different forms depending on the size of $F$ and its bisimulation class. Indeed, if $F$ has an even number of petals of even sizes and a self loop, or if it has an odd number of petals of even sizes and no self loop, then $F$ and $F^R$ are in the same bisimulation class; otherwise they are in different classes. Hence the ATG of $F$ has one of the following forms: (i) a unique SCC of size $2^n$ if $F$ and $F^R$ are in the negative class; (ii) two unreachable configurations, two fixed points, and one SCC if $F$ and $F^R$ are in the positive class; (iii) two fixed points, and one SCC if $F$ is in the positive class and $F^R$ in the negative class; (iv) two unreachable configurations, one SCC if $F$ is in the negative class and $F^R$ in the positive class.

### 4.2 $\oplus$-BAC Chains

A $\oplus$-BAC Chain ($\oplus$-BACC) of length $m$ is described by a set of $m$ cycles and $m - 1$ intersection automata, $o_k$, such that for all $1 \leq k < m$, the cycle $C_k$ intersects the cycle $C_{k+1}$ at a unique point $o_k = i^k_{y_k} = i^{k+1}_{y_k}$. As previously, we characterise the bisimulation classes of this type of BANs.

**Lemma 7.** The set of $\oplus$-BACs of length $m$ and size $(n_1, \ldots, n_m)$ admits one bisimulation class if $m - 1$ is not a multiple of 3 and two if $m - 1$ is a multiple of 3.
Proof. We give here a “pictorial” proof of Lemma 7 using the equivalences presented in Figure 4, numbered from (1) to (8). These equivalences have to be understood as follows: given a $\oplus$-BAN such that the left pattern of an equivalence appears in its interaction graph, then this BAN is equivalent to the BAN that has the same interaction graph except that the left pattern has been replaced by the right pattern of the equivalence, no matter what is the number and the type of arcs going out of the vertices with the outgoing dashed arcs. In other words Figure 4 presents a set of interaction graph rewritings that produce equivalent networks according to the bisimulation relation. Hence it is enough to prove that the interaction graph of a BAN can be rewritten into another one, to prove that the two corresponding BANs are equivalent.

Equivalences (1) and (2) only translate the well known identities $b_1 \oplus b_2 = b_1 \oplus b_2$ and $b_1 \oplus b_2 = b_1 \oplus b_2$ for any Boolean values $b_1$ and $b_2$. The proofs of the other equivalences are a bit longer but do not present any difficulties. Let us present one of them (the third one).

$\diamond$ Proof of Equivalence (3). Let $\mathcal{N} = \{f_i\}$ and $\mathcal{N}' = \{f'_i\}$ be two $\oplus$-BAN whose interaction graphs that only differ by the pattern shown in Equivalence (3). We denote by $C_1$, $C_2$ (respectively $o_1, o_2$) the two cycles (respectively intersection automata) of the pattern and by $C'_2$ the upper half cycle of $C_2$. We are going to prove that $\mathcal{N}$ bisimulates $\mathcal{N}'$ by using the conditions from Lemma 1: we take $\varphi$ to be the identity over the set of automata and $\phi_i$ to be the Boolean identity if automaton $i$ does not belong to $C_1$, $C'_2$ or $\{o_1\}$ and the Boolean negation otherwise. Then, we need to check that $\phi_i(f_i(x)) = f'_i(\phi_i(x))$ for all automata $i$ in the network. This is immediate for all automata that do not belong to $C_1 \cup C'_2 \cup \{o_1, o_2\}$ since we use the identity everywhere. Then, if $i \in C_1 \cup C'_2$, we also have $\phi_i(f_i(x)) = \phi_i(\text{pred}(i)) = \text{pred}(i) = \phi_{\text{pred}(i)}(\text{pred}(i)) = f'_i(\phi(x))$. So it
only remains to check that the equality holds for the automata \( o_1 \) and \( o_2 \). This is the case since:

1. \[ \phi_{o_1}(f_{o_1}(x)) = \phi_{o_1}(\text{pred}_1(o_1) \oplus \text{pred}_2(o_1)) = \overline{\text{pred}_1(o_1)} \oplus \overline{\text{pred}_2(o_1)} = \overline{f'_{o_1}(\phi(x))}, \]

2. \[ \phi_{o_2}(f_{o_2}(x)) = \phi_{o_2}(\text{pred}_1(o_2) \oplus \text{pred}_2(o_2)) = \overline{\text{pred}_1(o_2)} \oplus \overline{\text{pred}_2(o_2)} = \overline{f'_{o_2}(\phi(x))}. \]

Given the set of equivalences of Figure 4, we prove Lemma 7 in two steps: first we show that the interaction graph of any \( \oplus \)-BACC can be rewritten into an interaction graph with at most one negative sign on the arc from \( i_{n_1} \) to \( o_1 = i_{1} \). Then we prove that, in the case where \( m - 1 \) is not a multiple of 3, this negative can be removed by an other sequence of rewrites. The first point implies that for any \( m \) and \( n \), the set of \( \oplus \)-BACCs of length \( m \) and size \( n \) is made of at most two bisimulation classes: the positive class, that contains the BACCs whose interaction graph reduces to a graph where all arcs are positive, and the negative class, that contains the BACCs whose interaction graph reduces to a graph where all arcs are positive except the arc \( (i_{n}, i_{1}) \) which is negative. The second point says that, in fact, this two classes are only one if \( m - 1 \) is not a multiple of 3, since one can reduce the negative interaction graph to the positive one.

\( \diamond \) **Proof of the first point.** As usually we focus on the canonical BAN, since this already reduces the number of cases to consider. Then using Equivalences (1) and (2) from Figure 4 we can reduce the interaction graph of any \( \oplus \)-BACC to a graph where all negative paths are “on the top” that is the only negative arcs allowed are the ones between the intersection points and there left predecessor, \( (i_{n_k}, o_k) \).

Then, by induction on the position of the “right most” negative arc, we use Equivalences (5), (6), (7) and (8) to push this negative arc to the left, hence proving that any \( \oplus \)-BACC is bisimulable by a \( \oplus \)-BACC of same structure with at most two negative arcs on its first two cycles.

Finally, Equivalences (3) and (4) reduce the four base cases \(+,+,−,+,−,−,−\) to two: the positive case \(+,+\) and the negative case \(−,−\).

\( \diamond \) **Proof of the second point.** Consider the interaction graph of a negative \( \oplus \)-BACC of length \( m \). By Equivalence (2), this network is bisimulated by the \( \oplus \)-BACC of same structure with only one negative path on the first or on the second bottom half-cycle. Then, viewing the BACC upside-down, we can reuse Equivalences (6) and (8) alternatively so as to push this negative path to the right. Every time we apply Equivalences (4) and (6) successively the negative arc is pushed 3 half-cycles to the right. Finally, Equivalence (8) tells us that if the negative arc is pushed to the second to last bottom half-cycle then the \( \oplus \)-BACC is in the positive class. This is possible if \( m - 1 \equiv 1 \mod (3) \) or if \( m - 2 \equiv 1 \mod (3) \) (i.e. \( m - 1 \equiv 2 \mod (3) \)), depending on if we start from the first or from the second bottom half-cycle respectively. In other words, this is the case if \( m - 1 \) is not a multiple of 3.
Note that the equivalences presented in Figure 4 are exhaustive, i.e. any other equivalences can be deduced from these eight equivalences. So, the argument above also proves that it is impossible to bisimulate a positive $\oplus$-BACC with a negative $\oplus$-BACC if $m-1$ is a multiple of 3. In other words, if $m-1 \equiv 0 \mod (3)$ there are always two bisimulation classes, the positive one and the negative one.

For every class of $\oplus$-BACCs of a given length and size, we have then studied their number of fixed points.

**Lemma 8.** A positive $\oplus$-BACC of length $m$ and size $n$ has a unique fixed point, $0^n$, if $(m-1) \not\equiv 0 \mod 3$, and has two fixed points, $0^n$ and $(101)^{(m-1)/3}$, if $(m-1) \equiv 0 \mod 3$.

**Proof.** In a stable configuration every nodes of a given node path have the same state, hence from now on we focus on determining the states of the intersection automata $o_k$. As this is done in Section 4.1 for $\oplus$-BAF, we determined the fixed points of a $\oplus$-BACC by fixing the state of one of its automata and propagating the information induced until having to make a new choice or reaching a fixed point or a contradiction. Here, we start by fixing the “left most” automaton and by induction on the two possible cases ($x_{o_1} = 0$ and $x_{o_1} = 1$) we show that this completely determines the state of the other automata if we want to get a fixed point.

1. If $x_{o_1} = 0$, then $o_1$ is stable if and only if $x_{o_2} = 0$ and, recursively, for all $1 < k \leq m-2$, if $x_{o_{k-1}} = 0$ and $x_{o_k} = 0$ then $o_k$ is stable if and only if $x_{o_{k+1}} = 0$. Hence $0^n$ is the unique fixed point such that $x_0 = 0$.

2. Similarly, if $x_{o_1} = 1$ then $o_1$ is stable if and only if $x_{o_2} = 0$. Then, we have three induction cases for all $1 < k \leq m-2$: (1) if $x_{o_{k-1}} = 1$ and $x_{o_k} = 0$ then $o_k$ is stable if and only if $x_{o_{k+1}} = 1$; (2) if $x_{o_{k-1}} = 0$ and $x_{o_k} = 1$ then $o_k$ is stable if and only if $x_{o_{k+1}} = 1$; (3) if $x_{o_{k-1}} = 1$ and $x_{o_k} = 1$ then $o_k$ is stable if and only if $x_{o_{k+1}} = 0$. Hence the only way for the last intersection automaton, $o_{m-1}$, to be stable when $x_{o_1} = 1$ is that $(m-1) \equiv 0 \mod (3)$, and the corresponding configuration is $(101)^{(m-1)/3}$.

The proof above also shows Lemma 9 below.

**Lemma 9.** A negative $\oplus$-BACC (of length $m \equiv 1 \mod (3)$) has no fixed points.

**Proof.** Suppose $x_{o_1} = 0$ then $o_1$ cannot be stable no matter what is the state of $o_2$ in the configuration. Hence, if $x$ is a stable configuration $x_1$ must be 1. This forces $x_{o_2}$ to be 1 too (otherwise the automaton $o_1$ is not stable). From this point, finding the end of the stable configuration amounts to finding a stable configuration starting with a 1 for a $\oplus$-BACC of size $m-1$, which is impossible from the lemma above (Lemma 8). So there are no stable configurations for the negative $\oplus$-BACC of length $m \equiv 1 \mod (3)$.
So, similarly to the case of $\oplus$-BAFs, we can completely characterise the ATG of a $\oplus$-BACC $\mathcal{N}$ of length $m$ and size $n$ if $m - 1 \neq 0 \mod 3$, since in this case there is only one bisimulation class: $G^A_{\mathcal{N}}$ has exactly one unreachable configuration, one unique fixed point, and one SCC of size $2^n - 2$.

Conversely, the case where $m - 1$ is a multiple of 3 is more complex because there are no easy ways to tell whether a network belongs to the positive or the negative class other than to compute its reduction graph as this is done in the proof of Lemma 7. Moreover, the class of the reverse network also depends on the length of each half-cycle in the $\oplus$-BACC, so describing each possible cases would be tedious. However, summarising the results above, we can still state that there is at most two fixed points and two unreachable configurations in the transition graph of a $\oplus$-BACC of length $m - 1 \equiv 0 \mod 3$, or, to be more precise we can say that its transition graph has one of these four forms:

- a SCC of size $2^n - 4$, two fixed points and two unstable configurations (case $\mathcal{N}$ and $\mathcal{N}^R$ are from the positive class);
- a SCC of size $2^n - 2$ and two fixed points (case $\mathcal{N}$ is positive and $\mathcal{N}^R$ is negative);
- a SCC of size $2^n - 2$ and two unreachable configurations (case $\mathcal{N}$ is negative and $\mathcal{N}^R$ is positive);
- a SCC of size $2^n$ (case both $\mathcal{N}$ and $\mathcal{N}^R$ are negative).

5 Perspectives

This paper contributes to the growing comprehension of non-monotonic Boolean automata networks. The notion of bisimulation reveals to be a powerful proof factoring tool for the comprehension of their dynamical properties. Through general results and their application to particular classes of interaction graphs, the present work launches the description of asymptotic dynamical behaviours of $\oplus$-BANs under the asynchronous update mode. It should now be enriched with the study of larger classes of interaction graphs, as well as the study and comparison of asymptotic behaviours under different update modes.

References