# The asynchronous dynamic of some locally non-monotonic Boolean automata networks

Aurore Alcolei
Master thesis supervised by Kévin Perrot and Sylvain Sené

**Abstract**

Studies on the dynamics of Boolean automata networks (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 report explores analogous questions on non-monotonic networks, and focuses more particularly on $\oplus$-BANs (xor-BANs), that are BANs where every local transition function is a $\oplus$-function. Using algorithmic tools, we give a general characterisation of the asynchronous transition graphs for most of the strongly connected $\oplus$-BANs and cactus $\oplus$-BANs. As an illustration of these results, we provide a complete description of the asynchronous dynamics of two particular structures of $\oplus$-BAN, namely the $\oplus$-Flower and the $\oplus$-Cycle Chain BANs. This second work draws new behavioural equivalences between BANs, using rewrites on their graph description.

## Contents

# 1 Introduction

To start this report, I will first present the general context of my study and the motivations underneath (doing a brief overview of the domain). Then I will precise my contribution and give the road map of this document.

## 1.1 Background

A Boolean automata network (BAN) is a discrete interaction network, informally described as a set of Boolean entities (the automata) whose state (in $\mathbb{B} = \{0, 1\}$) may switch (be updated) over time, under the influence of the states of the other entities in the network.

BANs have been introduced by McCulloch and Pitts (under the name of nerve nets) in the middle of the last century [14]. In this seminal paper, BANs are presented both as a model of computation and as an abstract framework in which one can model neural activities. Thereafter, these modelling abilities kept being developed on the biological side and BANs are now a well established model for regulation systems such as neural networks [10, 9] or gene regulation networks [12, 28].

Since Thomas' works [28, 29], BANs used for biological modelling are generally defined to be asynchronous, that is, at most one automaton can be updated at a time. Adding synchronism to a BAN will then amount to allow several automata to switch state simultaneously. Unless stated otherwise, all the results presented in the sequel are stated for asynchronous BANs.

In the context of biological modelling, the Boolean state value of an entity can be thought as the entity being active or not (respectively in state 1 or 0). Following the same idea, the influence that one automaton has on another one describes in which circumstances the corresponding entity may activate, inhibit or be neutral to this other entity (respectively have a positive, negative or no influence on the other entity).

The structure (of influences) that arises from a BAN provides a graph representation of it. This graph representation gives their name of "networks" to BANs and is widely used when it comes to describing the "shape" (type) of a BAN.

When considered as a whole, the states of the automata of a BAN define the configuration in which this BAN is. Hence if one (or several) automaton switches state, this also makes the whole BAN move from one configuration to another. The dynamics of a BAN refers to its possible behaviours along time. There are basically two points of view that one can take to study them and this gives rise to the two main branches of this field of research:

- on the one hand, one can assume that the order in which the entities react to their environment is known (or fixed) in advance. From there, one can predict the whole behaviour of the system and one can try to study what kind of update sequences leads a given BAN to a given configuration. Hence, this point of view is particularly related to computability and complexity issues [13, 22, 8].

- on the other hand, one can assume the networks to be random. This assumption often holds in the study of discrete dynamical system and in biological modelling because it means that there are no predefined orders on the way the automata are updated. In this case, it is not possible to predict the whole behaviour of the system but one will be interested in characterising its asymptotic behaviour that is the set of recurrent configurations in which the network may end, given an initial configuration (such a set exists since the configuration space is finite). Of course,

this highly depends on the structure of the network, and being able to relate the structure of a network to its asymptotic behaviour is one of the main objective of the domain [16, 19, 3, 25, 24].

To position my work in this field of research, I would say that it addresses questions from the second branch but uses algorithmic tools that may echo the first one.

As we will see later, a BAN with $n$ automata is formally defined as a set of Boolean functions $\{f_i : \mathbb{B}^n \to \mathbb{B}\}_{i=1}^n$ such that each function describes the local behaviour of one automaton. These functions are called the local transition functions of the automata. A network is said to be locally monotonic if no automata has both a positive and a negative influence on another one. In other words, a BAN is locally monotonic if all its local transition functions are locally monotonic (in the analytical sense).

Because this mathematical specificity matches quite well with the behaviour of gene regulation networks, locally monotonic BANs have been well studied, both on the applied side [17, 7] and on the theoretical side [11, 23, 16, 19, 21]. However, recent works [20] have brought new interests in local non monotony. On the biological side, it has been shown that, sometimes, gene regulations imply more complex behaviour than what is usually assumed, as this is for example the case when one also takes in account the effect of their byproducts [27]. In this case, locally non monotony may be required for modelling, in particular because this allows to express sensitivity to the environment. On the theoretical side, it has been noticed [18, 21] that non local monotony is often involved when it comes to singular behaviours in BANs. For example it has been shown that the smallest network that is not robust to the addition of synchronism (*i.e.* allowing some automata to update simultaneously) is a locally non-monotonic BAN [18, 21].

Following these lines, the work presented in this report contributes to a better understanding of locally non-monotonic BANs.

### 1.2 Contribution

Because the set of locally non-monotonic BANs is very broad, this work focuses on $\oplus$-BANs, that is, BANs in which the state of each automaton is updated by xoring the state value (or the negated state value) of its incoming neighbours. The choice of $\oplus$-BANs amounts from the xor operator being the only binary Boolean operators to be non-monotonic (with the equivalence) . Hence $\oplus$-BANs are the simplest locally non-monotonic BANs one can think of.

A first attempt to characterise these networks is proposed in [21, 20] and focuses on $\oplus$-circulant networks. Following this constructive approach, I first looked at other BAN structures that combine cycles, such as the double-cycle graphs [2, 15], the flower-graphs [3] and the cycle chains. All these BAN structures belong to the family of cactus graphs (since any two of their simple cycles have at most one automaton in common) which is a well-known family in the (locally) monotonic context [2, 3, 15].

Happily, it turned out that the results I got with these preliminary studies were in fact generalisable to a wider set of $\oplus$-BANs: the strongly connected $\oplus$-BANs with an induced double cycle of size greater than 3. I will give a precise definition of these BANs in the following section. However, let us mention here that this family of BANs contains in particular every strongly connected cactus BAN and that its characterisation heavily relies on that of the Boolean automata double cycles (BADCs).

As mentioned above, Section 2 introduces all the definitions, notations and basic results from BAN theory that are used in this report. Then Section 3 presents (and proves) the

general result obtained about the asynchronous dynamic of strongly connected $\oplus$-BANs with an induced BADC of size greater than 3. To do so, Section 3 also provides a full characterisation of $\oplus$-BADCs, used as a starter for the proof of the main result. Section 4 illustrates the results of Section 3 with a full characterisation of two types of $\oplus$-BANs, the $\oplus$-flower BANs and the $\oplus$-cycle chain BANs. This also provides new bisimulation results specific to $\oplus$-BANs, that is, equivalence results between $\oplus$-BANs. Finally, Section 5 is dedicated to the conclusion and perspectives of this present work.

*Remark.* In the following, proofs are sometimes omitted or shortened. The mention "sketch" implies that a full version can be found in the Appendix.

# 2 Definitions, notations and some basic properties

This section introduces the main definitions and notations used in the rest of this report. It also presents some basic results related to them. In particular it describes the static representation(s) of BANs (as Boolean function sets and as interaction graphs), their dynamics (viewed as transition graphs), and the important notion of bisimulation which is an equivalence relation over the set of Boolean networks. This section ends on a short discussion about the pertinence of these definitions.

## 2.1 Static definition of a BAN

**Generalities on BANs** 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 $\mathcal{N}$ of size $n$ we denote $V = \{1, \dots, n\}$ 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 $\mathcal{N}$. We will shorten by $\overline{x}^i$ the configuration $x$ where the state of the $i^{th}$ automaton is negated and similarly, for any subset $I$ of V, $\overline{x}^I$ 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* $f_i : \mathbb{B}^n \to \mathbb{B}$. 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. By extension we say that $x$ is *stable* if all the automata of $\mathcal{N}$ are stable in $x$. Hence a network $\mathcal{N}$ 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 $f_j(x) \neq f_j(\overline{x}^i)$. 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*, $p = i_0 i_1 \dots i_k$, of *length* $k$ is a sequence of distinct automata such that for all $1 \leq j \leq k$, $i_{j-1} \in I_j$. By extension, a BAN is *strongly connected* if there is a path between every two automata.

A *nude path*, $\pi = i_0 i_1 \dots i_k$, is a particular path such that for all automata $i_j \in \pi$ $(j > 1)$, $i_{j-1}$ is the unique influencer of $i_j$ *i.e.* $I_j = \{i_{j-1}\}$. In other words, for all $1 \leq j \leq k$, $f_{i_j}(x) = x_{i_{j-1}}$ or $f_{i_j}(x) = \overline{x_{i_{j-1}}}$. This allows us to define the *sign* of a nude path as the parity of the number of local functions of the form $f_i(x) = \overline{x_{i-1}}$ that compose it, *i.e.* $sign(\pi) = \left(\sum_{j=1}^n \mathbf{1}_{f_j(x)=\overline{x_{j-1}}}\right) \mod 2$. 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 thereafter.

To get a sense of what a network looks like, it is common to give a graphical representation of it. To every local function $f_i$, 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 $\mathcal{N}$ according to these formulae is the signed directed multigraph $G_{\mathcal{F}} = (V, A_{\mathcal{F}})$, where $V = \{1, \dots, n\}$ is the set of nodes of $G_{\mathcal{F}}$ with one entry points per literal in $\mathcal{F}_i$, 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 $\mathcal{F}_j$ has sign $\sigma_k$ (see Figure $1_a$).

With such a definition, the interaction graph of a BAN is not unique, it depends on the choice of the formulae $\mathcal{F}_i$. The uniqueness can be achieved by choosing a normal form for the writing of the formulae. In this context, the *type* of a BAN will refer to the underlying structure of its interaction graph (ignoring the labels). In other words, we will say that two BANs are of the *same type* if they have the same local formulae modulo the sign of the literals and a renaming of the automata, or, equivalently, if their interaction graphs are isomorphic modulo the sign on their arcs.

As we focus on $\oplus$-BANs, we make the natural choice of representing every local function $f_i$ by a formula in Reed-Muller canonical form, that is $\mathcal{F}_i = \bigoplus_{j \in I_i} \sigma_j(x_j)$. Then, *the* interaction graph $G_{\mathcal{N}}$ of a $\oplus$-BAN $\mathcal{N}$ will refer to this particular interaction graph. In this representation, the incoming neighbours of an automaton $i$ in $G_{\mathcal{N}}$ are exactly the influencers of $i$ in $\mathcal{N}$, in other words, the set $\{j \mid (i, j, \sigma_j) \in A\}$ equals $I_i$. Hence there is a one to one correspondence between the paths of $\mathcal{N}$ (resp. nude paths of $\mathcal{N}$) and the paths of $G_{\mathcal{N}}$ (resp. the paths of $G_{\mathcal{N}}$ such that every node in the path, except possibly the head, has only one incoming neighbour). Because of this correspondence we will allow us to switch from one representation to another without making this explicit.

We now have the tools to precise a little more the kind of $\oplus$-BANs we are going to look at.

**Basic interaction structures investigated** The simplest interaction structure that allows for complex behaviour is the cycle structure [26]. A *Boolean automata cycle* (BAC) $\mathcal{C}$ of size $n$ is a BAN defined as a set of local functions $\{f_i\}_{i=1}^n$ such that $f_i(x) = x_{((i-1) \ mod \ n)}$ or $f_i(x) = \overline{x_{((i-1) \ mod \ n)}}$ for all $i \in \{1, \dots, n\}$. Abusing notation we will often express $f_i$ via its formula representation $\mathcal{F}_i = \sigma_i(x_{pred(i)})$ where $pred(i) = (i - 1 \ mod \ n)$ is the only influencer of $i$ in $\mathcal{C}$ 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 $f_i(x) = \bigoplus_{j=1}^{\ell} \sigma_j(pred_j(i))$ where $pred_j(i)$ represents the predecessor of $i$ in each of the incident cycles.

If a BAN is described in terms of simple cycles, $\mathcal{C}_1, \dots, \mathcal{C}_m$, intersecting with each other, we will often represent its size by a vector of $m$ natural numbers $n = (n_1, \dots, n_m)$, 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: $x = (x^1, \dots, x^m) \in \mathbb{B}^{n_1} \times \dots \times \mathbb{B}^{n_m}$ will represent the configuration where each cycle $\mathcal{C}_k$ is in configuration $x^k \in \mathbb{B}^{n_k}$. By extension $x_j^k$ will denote the state of automaton $i_j^k$ which is the $j^{th}$ automaton of cycle $\mathcal{C}_k$.

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* $\oplus$-BANs are special strongly connected $\oplus$-BANs where any two simple cycles intersect each other at most once [4]. The simplest example of such BANs are the $\oplus$-*Boolean automata double-cycles* ($\oplus$-BADCs). These $\oplus$-BANs are described by two cycles $\mathcal{C}_1$, $\mathcal{C}_2$ 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 $(2, 1) = 2 + 1 - 1 = 2$.

## 2.2 Dynamics of a BAN and the specific asynchronous mode

**Update modes and Transition graphs** 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 $F_W : \mathbb{B}^n \to \mathbb{B}^n$ 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 [19] but sufficient for the scope of this work.

We say that a configuration $x'$ is *reachable from* a configuration $x$ (in a mode $M$) if there exists a finite sequence of updates $(W_t)_{t=1}^s$ (in $M$) such that $F_{W_1} \circ \ldots \circ F_{W_s}(x) = x'$. 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 $M$). Note that $x$ is a fixed point of a network $\mathcal{N}$ in the asynchronous update mode if and only if $f_i(x) = x_i$ for all $i \in V$, *i.e.* if and only if $x$ is a stable configuration.

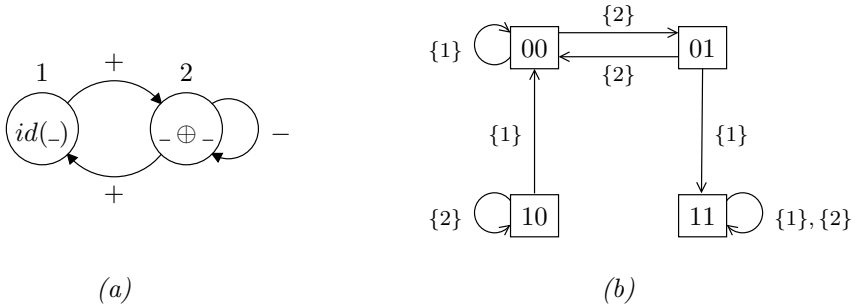
The study of the dynamics of a network under a particular update mode aims at making predictions, *i.e.* given an initial configuration $x$, we want to tell what are the possible sets of configurations in which the network can end asymptotically. These sets are called *attractors* of the network and 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 modelled by a labeled directed graph $G_{\mathcal{N}}^M = (\mathbb{B}^n, \bigcup_{W \in M} F_W)$, 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 $G_{\mathcal{N}}^M$ if and only if $W \in M$ and $F_W(x) = x'$.

The transition graph $G_{\mathcal{N}}^A$ associated to the asynchronous update mode is called the *asynchronous transition graph* of $\mathcal{N}$, 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 $G_{\mathcal{N}}^M$, 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 $G_{\mathcal{N}}^M$ that are connected to this component. Conversely, the unreachable configurations of $M$ are the configurations that do not have any incoming arcs




*(a)* *(b)*

Figure 1: *(a)* The interaction graph of BAN $\{f_1(x) = x_2, f_2(x) = x_1 \oplus \overline{x_2}\}$ and *(b)* its asynchronous transition graph.

but self-loops in $G^M_{\mathcal{N}}$. For example, in Figure $1_b$, the configuration 10 is unreachable, the configuration 11 is a fixed point and its attraction basin the whole set of configurations.

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.

**Basic properties of the asynchronous mode** At this point we pause on the above definitions to state some general remarks about the set of fixed points and unreachable configurations of the ATG of any BAN. These remarks will help us in the sequel to detail the description of the ATGs of the $\oplus$-BANs we study. They will also be necessary preliminaries to the proof of Theorem 3 that we will give in Section 3.

**Lemma 1.** *The set of unreachable configuration is exactly the set of configurations $x$ such that $f_i(\overline{x}^i) = \overline{x_i}$.*

*Proof.* This statement is easily shown by contradiction: Let $x$ be as above and suppose that $x$ is reachable from some configuration $x'$ with $x'_i \neq x_i$ for some $i \in V$. The set of automata in $\mathcal{N}$ that must be updated to go from $x'$ to $x$ is non-empty so let $j$ be the last automaton to be (effectively) updated in the path from $x'$ to $x$ and let $x''$ be the configuration just before the update. We have $\overline{x''}^j = x$ so $f_j(x'') = f_j\left(\overline{\overline{x''}^j}^j\right) = f_j\left(\overline{x}^j\right) = \overline{x_j} = x''_j$ and so the update is not effective. This is a contradiction. □

This simple result is useful when studying the structure of the transition graph because it enables to evince the unreachable configurations when one tries to characterise the set of configurations that are reachable from a given configuration. In particular, we will see (Theorem 3) that in any strongly connected $\oplus$-BAN with an induced $\oplus$-BADC of size greater than 3, the only configurations that cannot be reached from a given unstable configuration are exactly the unreachable configuration. Moreover, Lemma 1 also generalises some intermediate results stated in [15].

As mentioned previously, a configuration $x$ is a fixed point if and only if 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.** *For any BAN $\mathcal{N}$, the maximum number of fixed points in its ATG is $2^k$, where $k$ is the number of automata $i$ such that $\pi_i$ is of length 0 (i.e. $k$ is the number of "intersection automaton" in the interaction graph of $\mathcal{N}$).*

*Proof.* As we have noticed above, a stable configuration is completely determined by the states of the head of its maximal nude paths. Hence there is at most $2^k$ distinct stable configuration in $G^A_{\mathcal{N}}$. □

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 $i$ sucht that $\pi_i$ has length greater than 1 and replacing its 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 from Lemma 1 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 $f_i^R(x) = \overline{f_i(\overline{x}^i)}$. $\mathcal{N}$ 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 also implies that if all the networks of a given type pertain to the same bisimulation class (defined in the next subsection) then their number of unreachable configurations and their number of fixed points are equal.

### 2.3 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 $\phi : \mathbb{B}^n \to \mathbb{B}^n$ 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 by Noual in [19]. To prove a bisimulation relation between two networks we will often prefer to use a stronger condition than the one given in the definition, but which has the advantage of being local.

**Lemma 3.** *Two BANs $\mathcal{N} = \{f_i\}_{i=1}^n$ and $\mathcal{N}' = \{f'_i\}_{i=1}^n$ bisimulate each other if there exists a bijection $\varphi : \{1,\dots,n\} \to \{1,\dots,n\}$ and a set $\{\phi_i : \mathbb{B} \to \mathbb{B}\}_{i=1}^n$ of (non constant) Boolean functions such that for all automata $i$, $\phi_i \in \{id, neg\}$, and for all configurations $x \in \mathbb{B}^n$, $\phi_i(f_i(x)) = f'_{\varphi(i)}(\phi(x))$ where $\phi(x)$ is defined componentwise by $\phi(x)_i = \phi_{\varphi^{-1}(i)}(x_{\varphi^{-1}(i)})$.*

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

In her thesis [19], Noual also shows some general bisimulation results that we recall (and extend) here.

**Theorem 1** ([19])**.** *Let $\mathcal{N} = \{f_i\}_{i=1}^n$ be a BAN and $\mathcal{N}^\perp = \{f_i^\perp\}_{i=1}^n$ be its* dual *network defined as $f_i^\perp(x) = \overline{f_i(x)}$ then $\mathcal{N}$ and $\mathcal{N}^\perp$ bisimulate each other.*

*Proof.* Take $\varphi$ to be the identity on $V$ ($\varphi = id_V$) and define $\phi : \mathbb{B}^n \to \mathbb{B}^n$ such that $\phi(x) = \overline{x}$ (*i.e.* $\phi_i = neg$), then $\phi_i(f_i(x)) = \overline{f_i(x)} = \overline{f_i(\overline{x})} = \overline{f_i(\overline{\phi(x)})} = f_i^\perp(\phi(x))$ □

**Theorem 2** ([19]). *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) $f_i^+(x) = x_j$ *if* $f_i(x) = x_j$ *or* $\overline{x_j}$, *and* (ii) $f_i^+(x) = f_i(\overline{x}^I)$ *otherwise, where* $I = \{i \in V \mid sign(\pi_i) = 1\}$ *is the set of automata whose maximal incoming nude path has negative sign. Then* $\mathcal{N}$ *and* $\mathcal{N}^+$ *bisimulate each other.*

*Proof.* Take $\varphi = id_V$ and for all $i \in V$ defined $\phi_i$ to be: *(i)* the Boolean identity if $\pi_i = i$ (*i.e.* $\pi_i$ of length 0 and $i$ is an intersection automaton), or if $sign(\pi_i) = 0$; *(ii)* the Boolean negation otherwise (if $sign(\pi_i) = 1$ *i.e.* $i_i n I$). Then by induction on each nude path $\pi = i_0 i_1 \dots i_k$ of $\mathcal{N}$ we prove that we prove that for all $i_j \in \pi$, $\phi_{i_j}(f_{i_j}(x)) = f^+_{\varphi(i_j)}(\phi(x)) = f^+_{i_j}(\phi(x))$ $(*)$. Hence $(*)$ hold for all automata in $\mathcal{N}$.

The detail of the proof by induction is as follows:

- *(base case: $j = 0$)* $\pi_{i_0} = i_0$ so $\phi_{i_0}(f_{i_0}(x)) = f_{i_0}(x) = f_{i_0}(\overline{\overline{x}^I}) = f_{i_0}(\overline{\phi(x)}^I) = f^+_{i_0}(\phi(x))$. Hence $(*)$ holds for the head of $\pi$.

- *(induction step: $j > 1$)* suppose that $(*)$ holds for $i_{j-1}$ then:

  *(i)* if $sign(\pi_{i_j}) = sign(\pi_{i_{j-1}})$ (*i.e.* $f_{i_j}(x) = x_{i_{j-1}}$) then $\phi_{i_j} = \phi_{i_{j-1}}$ and so $\phi_{i_j}(f_{i_j}(x)) = \phi_{i_{j-1}}(x_{i_{j-1}}) = f^+_{i_j}(\phi(x))$;

  *(ii)* if $sign(\pi_{i_j}) \neq sign(\pi_{i_{j-1}})$ (*i.e.* $f_{i_j}(x) = \overline{x_{i_{j-1}}}$) then $\phi_{i_j} = \overline{\phi_{i_{j-1}}}$ and so $\phi_{i_j}(f_{i_j}(x)) = \overline{\phi_{i_{j-1}}(\overline{x_{i_{j-1}}})} = \phi_{i_{j-1}}(x_{i_{j-1}}) = f^+_{i_j}(\phi(x))$.

□

Theorem 1 is of importance because it tells us that all the results stated in the sequel for $\oplus$-BANs will also hold for $\Leftrightarrow$-BANs, which are their dual BANs (BANs with local functions are of the form $f_i(x) = \underset{j \in I_i}{\Leftrightarrow} \sigma(x_j)$). Moreover, 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 the networks with positive nude paths to characterise the whole set of possible transition graphs of a given type of networks.

We will make great use of Theorem 2 and Lemma 3 in Section 3 and 4.

**A note about the definitions in Section 2**

In the literature, the notion of interaction graph is slightly different from the one presented in Section 2.1: there is an arc between two automata $i$ and $j$ in $G_{\mathcal{N}}$ if and only if $i$ is an influencer of $j$ (*i.e.* there exists a configuration $x \in \mathbb{B}^n$ such that $f_j(x) \neq f_j(\overline{x}^i)$). In other words, the interaction graph of an network is usually a simple representation of the influence that the automata have on each other. Then, in the case of locally monotonic networks it is also common to sign the arcs of $G_{\mathcal{N}}$ to add extra information on the type of influence (positive or negative) that one automaton has on another.

While their graph representation is a useful tool to study BANs, I could not work with the usual definition of interaction graph for two main reasons.

The first one was that two networks with the same interaction graph (in the original sense) do not necessarily have the same behaviour (it is enough to think about a BADC where the central automaton is either a $\oplus$-(xor) or a $\vee$-(or) function). Hence the result of canonicity present in [19] and that we remind in the above section could not hold as stated! (while the idea behind it was perfectly right).

The second reason was that classically interaction graphs were used to describe locally monotonic BANs, where the notion of positive/negative influence makes sense for every

nodes and was added as extra bits of information to the graph representation. However, in the context of locally non-monotonic BANs the interaction graphs were left with very few information and this graph representation was not precise enough to state any interesting result about there corresponding networks. That is why we came up with another definition of interaction graph, that fully described the network it is associated with. In the end, this definition is very close to the usual graph description of Boolean circuits.

This new definition of interaction graph also seems reasonable because it preserves the connectivity of the original definition (which we call *influence graph* from now on). Indeed, there is a path $p = i_0 \dots i_k$ in the influence graph of $\mathcal{N}$ if and only if there is at least one path $p = i_0 \dots i_k$ with same automata in any interaction graph of $\mathcal{N}$. Hence the strong connectivity of a BAN or the presence of a cycle in its influence graph are invariant of the choice interaction graph for $\mathcal{N}$. As a consequence, the results already stated in the literature adapt quite well to the new definition.

Moreover, it appears that most of the networks usually studied are actually $\diamond$-BANs, that is, BANs in which every local functions is obtained using an unique Boolean operator $\diamond$ in $\{\wedge, \vee, \oplus, \Leftrightarrow\}$. It turns out that, for all of these BANs, the influence graph and the interaction graph matches if one make the natural choice of representing the local formulae of the interaction graph using the $\diamond$ operator. Hence most of the time it was assumed that a BAN was completely described by its influence graph. In particular, this was the case when the bisimulation result between a network and its canonical representation was first stated (the author was at the same time working on $\vee$-BAN).

# 3 General results on $\oplus$-BANs

This section presents the main theorem of my work: 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 that is not unreachable can be reached from any configuration which is not stable in a quadratic number of asynchronous updates.*

This theorem told 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 $S \setminus U$ (within a quadratic number of updates).

The above results are in fact a generalisation of the characterisation and connexity result that we got for $\oplus$-BADCs in the first time of our study. Furthermore, the proof of this theorem relies on the proof of these preliminary results. For this reasons, and in order to get the reader used to the kind of reasoning we will develop in the rest of this report, we start this section with a full characterisation of the asynchronous dynamic of $\oplus$-BADC and we will only give the proof of Theorem 3 in the second part of this section.

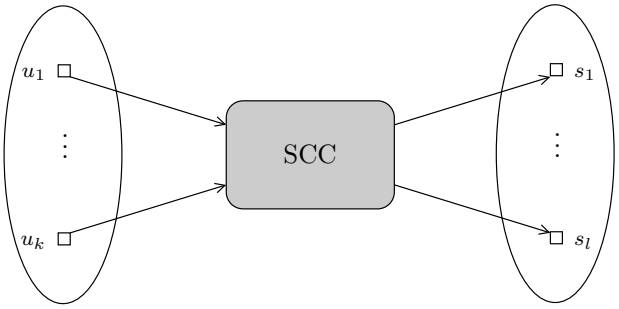



Figure 2: General ATG shape of strongly connected $\oplus$-BANs with an induced BADC of size greater than 3.

### 3.1 Preliminary results on $\oplus$-BADC

Besides the role they play in the dynamic of many $\oplus$-BANs, $\oplus$-BADCs are interesting in themselves because they are the most simple (yet non trivial) locally non-monotonic BANs one can build.

In the following, I will prove that the asymptotic behaviour of $\oplus$-BADCs in the asynchronous mode is very simple: every $\oplus$-BADC has a unique fixed point which is reachable in a linear number of updates. This result might not be very exciting with regards to its complexity but it provides a good example to present the kind of reasoning used in the sequel. Moreover, despite their apparent simplicity $\oplus$-BADCs are in fact very expressive and in particular I will also show that their structure provides a powerful way to generate reachable configurations.

**Asymptotic behaviour** By definition the interaction graph associated to a BADC is a double cycle graph with one intersecting point $o$. Hence Theorem 2 states that for a given type of BADC, there are only three different bisimulation classes to study: the *positive* one, the *negative* one and the *mixed* one, that respectively correspond to the case where $f_o(x) = x_1^1 \oplus x_1^2$, $f_o(x) = \overline{x_1^1} \oplus \overline{x_1^2}$ and $f_o(x) = \overline{x_1^1} \oplus x_1^2$. We now show that, in the case of $\oplus$-BADCs, these three classes actually reduce to one.

**Lemma 4.** *The set of $\oplus$-BADC of size $(n_1, n_2)$ admits exactly one bisimulation class.*

*Proof.* (sketch) This is induced by the fact that the positive $\oplus$-BADC representative bisimulates both the negative and the mixed $\oplus$-BADC representatives: In the first case, the equality $x_1^1 \oplus x_1^2 = \overline{x_1^1} \oplus \overline{x_1^2}$ implies that the positive and the negative $\oplus$-BADCs are trivially equivalent; in the second case the bijection $\phi(x) = \overline{x}^V$ over the set of configuration satisfies the condition from definition 1, proving that positive $\oplus$-BADCs and mixed $\oplus$-BADCs bismulate each other. □

Because of these reductions, we only need to focus on the asymptotic behaviour of positive $\oplus$-BADCs to fully characterise the asymptotic behaviour of any $\oplus$-BADCs.

There are several ways to compute the fixed points of a $\oplus$-network. One way is to compute the solution of the linear system $Ax = x$ where $A$ is the adjacency matrix associated to the interaction graph of the network. Another 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. This basic backtracking algorithm is not

an efficient way of computing fixed points but it is useful to prove that some configurations cannot be stable. In particular, it helps us show that the asymptotic behaviour of a $\oplus$-BANs is quite simple: from any configuration, the system converges to a unique fixed point.

**Lemma 5.** *A positive $\oplus$-BADC of size $(n_1, n_2)$ admits a unique fixed point $x = (0^{n_1}, 0^{n_2})$.*

*Proof.* In a positive $\oplus$-BADC $D$, 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 $D$, 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 $f_o(x) = x_{n_1} \oplus x_{n_2} = 0 \neq x_o = 1$. Furthermore, we can check that $x = 0^n$ is indeed a fixed point since for all $i$, $f_i(x) = 0$: *(i)* if $i = i^k_j \neq o$ then $f_i(x) = x^k_{j-1} = 0$, and *(ii)* if $i = o$ then $f_i(x) = x_{n_1} \oplus x_{n_2} = 0 \oplus 0 = 0$. □

Using the remarks from Section 2.2 and the lemma above we can thus conclude that any $\oplus$-BADC admits exactly one fixed point and one unreachable configuration in its transition graph.

**Connexity** We now go one step further and show that the rest of the ATG is made of a unique strongly connected component that can reach its fixed point in a linear number of updates if $n$ is greater than 3. These results are base on the following lemma:

**Lemma 6.** *In a positive $\oplus$-BADC $D$ of size greater than 3, if $D$ is in an unstable configuration $x$ then for all automaton $i$ there exists an other unstable configuration $y$ where $i$ is unstable and that is reachable from $x$ in a linear number of updates.*

*Proof.* The proof uses the fact if that if $D$ is unstable then there exists at least one automaton $i^j_k$ in state 1. Hence one can reach a configuration where at least one influencer of the central automaton $o$ of $D$ is in state 1 by propagating this state along $\mathcal{C}_k$. This leads to a configuration where $o$ is unstable and thus able to propagates instability to every other automaton in the network. More precisely, in a positive $\oplus$-BADC, if the central automaton receives a value 1 from one of its influencers then it can switch state as many times as desired by sending its own state along the opposite cycle. To make this explicit, suppose that $x_{pred_1(o)} = 1$ then updtating the automata along $\mathcal{C}_2$ will lead to a configuration where $x_{pred_2(o)} = x_o$ and so $f_o(x) = x_{pred_1(o)} \oplus x_{pred_2(o)} = 1 \oplus x_o = \overline{x_o}$. Hence it is possible to set any automaton $i^k_j$ of $D$ to some state $b$, by setting $o$ to $b$ and then propagating $b$ along $\mathcal{C}^k$ until reaching $j$. Moreover, one can ensure that this will be possible again, if in the end at least one of the two predecessors of $o$ is in state 1. The only threat for breaking this rule is when $i^k_j = i^k_{n_k}$ and $b = 0$, *i.e.* when $i^k_j$ is an influencer of $o$ and when we want to set it to 0. In that case we need to set the opposite predecessor to 1 right after starting propagating $b$ in $\mathcal{C}_k$, that is right after setting $i^k_2$ to $b$. This is only possible if $D$ is at least of size 3. □

Lemma 6 is interesting because it reveals the "generator nature" of $\oplus$-BADCs. From there one can deduce two propositions characterising the ATG of a $\oplus$-BADC.

**Lemma 7.** *In a positive $\oplus$-BADC of size $(n_1, n_2)$, the configuration $0^n$ is reachable from any other configuration in a linear number of updates.*

*Proof.* Using the algorithm from Lemma 6 we know how to set $o$ to the state 0 in a linear number of updates from any unstable configuration (which is the case of any configuration different from $0^n$, by Lemma 5). Then it is enough to propagate this state in $\mathcal{C}_1$, $\mathcal{C}_2$ to reach $0^n$ in another linear number of steps. □

**Lemma 8.** *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.* (sketch) The proof uses the same generator trick as in the proof above. The idea is first to reach a "highly expressive" configuration $y$ from which any other configuration can be reached in a linear number of updates. $y$ is said to be highly expressive because it contains only unstable automata except the automaton $i$ that is suppose to be stable in the configuration to reach ($i$ exists since $x'$ is reachable). From there, updating the automata in an order such that every automaton is updated before one of its influencer, except $i$ that is updated last, will lead to the desired configuration. The quadratic factor arises when one sets the network to $y$ because one needs to set almost every automata in an unstable state and this take a linear number of updates for each automata, as in Lemma 6. □

Summarising all the results we gave in this subsection, we can say that the ATG of a $\oplus$-BADC contains exactly one fixed point, one unreachable configuration, and one strongly connected component (SCC), such that the maximal distance between two reachable configurations is quadratic and the maximum distance between one configuration and the fixed point is linear (in $n$).

### 3.2 Proof of Theorem 3

We now give the main ideas of the proof of Theorem 3. Let $\mathcal{N}$ be a strongly connected $\oplus$-BAN with an induced BADC $D$ 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 $D$ 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 Lemmas 6 and 8 and on the following lemma:

**Lemma 9.** *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.* (sketch) The proof is based on the fact that, in a $\oplus$-BAN, making a stable automaton become unstable can only be achieved by switching the state of one of its incoming neighbours (because the state of an automaton depends on the parity of the number of its incoming neighbours in state 1). So let $i$ and $j$ be two automata as described in Lemma 9, let $p = i_0, i_1, \dots, i_k$ 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 straightforward 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 automata after it, and this is true if one takes a shortest path. □

*Proof.* (sketch for Theorem 3) Putting things together we can now describe the algorithm underlying the proof of Theorem 3: the network $\mathcal{N}$ starts in an unstable configuration so, by Lemma 9, it can reach a configuration $y$ where an automaton of its induced $\oplus$-BADC $D$ (hence $D$) is unstable. According to Lemma 6 and the developments above, if $D$ is unstable then the state of any of its automata can be switched and this process can be repeated

as many times as necessary while leaving $D$ in an unstable configuration. Through this extent, $D$ can be viewed as a "state generator" that can propagate instabilities through the network. Indeed, $\mathcal{N}$ is strongly connected so there is a path from $D$ to every automata out of $D$. As a consequence, using the algorithm from Lemma 9, we are able to set the state of every of these automata to their value in $x'$. One subtlety is to find an order to process the automata so as to guarantee that setting an automaton to its state in $x'$ will not switch the state of the automata that have already been set up. We deal with this problem by using a breadth first search tree of root $D$: we set up the automata from the leaves to the root and use the branches of the tree to be the paths used in the algorithmic process of Lemma 9. Finally, when everything is fixed outside of $D$ ($D$ is still in an unstable configuration) we apply Lemma 8 to fix $D$ and reach $x'$.

There are two subtleties to the above description. The first one is that switching the state of the automata outside of $D$ can leave $D$ in a stable configuration, hence blocking the mechanism. It is possible to avoid that by modifying the states within $D$ according to the change of its environment, but in some case this requires to have at least 3 automata in $D$, hence the condition in Theorem 3. The second subtlety is in the last step, when it comes to setting the states within $D$. In fact, this only works if the restriction of $x'$ to $D$ is reachable in $D$, that is if $x'_{|B}$ is not unreachable in the ATG of $D$ with surrounding environment $x'_{\mathcal{N}\setminus B}$. Unfortunately this is not necessarily the case in any reachable configuration of $\mathcal{N}$. So, if this condition is not satisfied one needs to use an additional trick whose idea is the following: $x'$ is reachable so there exists $i \in \mathcal{N}$ such that $f_i(\overline{x'}^i) = x'_i$. Let $p = i_0 \dots i_k$ be a a shortest path from $i$ to $D$, then first reach the configuration $y$ such that every automata are in the state of $x'$ except the automata of $p$ that are alternating and such that $y_{i_k} \neq x_{i_k}$ (hence the restriction of $y$ to $D$ is reachable). Then set up the state of the automata from $i_k$ to $i_0$ using the instability of their predecessor in $p$ if necessary, or, for the case $i_0 = i$ using the fact that $f_i(\overline{x'}^i) = x'_i$. □

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 along this path in $x'$ (*i.e.* the number of automata such that $x'_i \neq x'_{pred(i)}$ where $pred(i)$ is the predecessor of $i$ in the path; this corresponds to the number of 01 or 10 patterns in the positive case), then if this number is less than a constant in any path the algorithm will also run in linear time. In particular, if $x'$ is a fixed point, then the total number of alternating states is at most $k$, where $k$ is the number of intersection automata in $\mathcal{N}$ and so $x'$ is reachable in $O(k \cdot n)$ updates, which is linear if $k$ is a constant, as this is for example the case with the $\oplus$-BA Flowers defined in the next section. Finally we need to insist on the fact that this algorithm does not always provides the most efficient sequence of updates (for example it does not take in 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 [15]).

# 4 Study of some specific ⊕-BANs

We now give a complete characterisation of two specific types of ⊕-BAN: the ⊕-BA Flowers and the ⊕-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 ⊕-BAN.

## 4.1 ⊕-BA Flowers

A ⊕-BA Flower (⊕-BAF) with $m$ petals is defined as a set of $m$ cycles that intersect at a unique automaton $o = i_1^1 = \ldots = i_1^k$ (⊕-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 10.** *The set of ⊕-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 3.1 for the ⊕-BADCs, we restrict our study to the canonical ⊕-BAFs, that are the ⊕-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 $\phi(x) = \overline{x}^V$ over the set of configurations defines an isomorphism between the ATGs of the negative and the positive ⊕-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 11. □

**Lemma 11.** *A positive ⊕-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 ⊕-BAF (with an odd number of petals) does not have any fixed point.*

*Proof.* We use the same approach as the one explained in the proof of Lemma 5, fixing the state of one automaton and cascading this choice onto the other in order to reach a stable configuration or a contradiction. For example, in a positive ⊕-BAF $\mathcal{F}$ with an even number of petals, we can prove that 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 $\mathcal{F}$, 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 ⊕-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. □

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

- if $\mathcal{F}$ is a ⊕-BAF with an even number of petals then $\mathcal{F}$ and its reverse network $\mathcal{F}^R$ both have ATGs isomorphic to the ATG of the positive ⊕-BAF, consequently

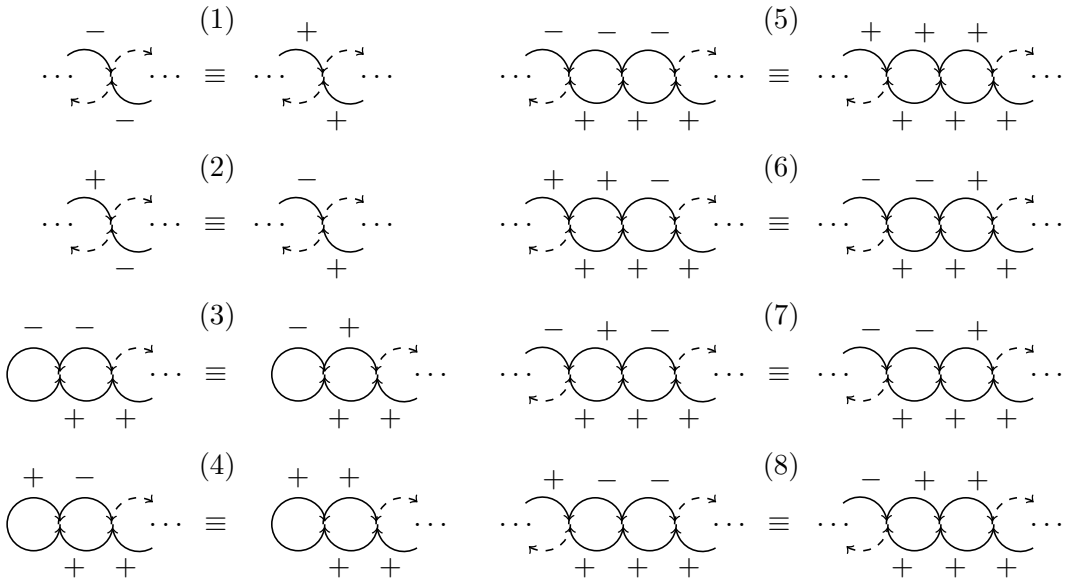



Figure 3: Table of $\oplus$-equivalences.

$G^A_{\mathcal{F}}$ has exactly one unreachable configuration, one fixed point, and one SCC of size $2^n - 2$.

- if $\mathcal{F}$ is a $\oplus$-BAF with an odd number of petals then the ATG of $\mathcal{F}$ can have four different forms depending on the size of $\mathcal{F}$ and its bisimulation class. Indeed, if $\mathcal{F}$ 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 $\mathcal{F}$ and $\mathcal{F}^R$ are in the same bisimulation class ; otherwise they are in different classes. Hence the ATG of $\mathcal{F}$ has one of the following forms: *(i)* a unique SCC of size $2^n$ if $\mathcal{F}$ and $\mathcal{F}^R$ are in the negative class ; *(ii)* two unreachable configurations, two fixed points, and one SCC if $\mathcal{F}$ and $\mathcal{F}^R$ are in the positive class; *(iii)* two fixed points, and one SCC if $\mathcal{F}$ is in the positive class and $\mathcal{F}^R$ in the negative class ; *(iv)* two unreachable configurations, one SCC if $\mathcal{F}$ is in the negative class and $\mathcal{F}^R$ 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 $\mathcal{C}_k$ intersects the cycle $\mathcal{C}_{k+1}$ at a unique point $o_k = i^k_1 = i^{k+1}_{\ell_k}$. As previously, we characterise the bisimulation classes of this type of BANs.

**Lemma 12.** *The set of $\oplus$-BACCs 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.* (sketch) The proof of Lemma 12 is based on the equivalences presented on Figure 3. 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 3 presents a set of interaction graph rewriting rules 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 the other, to prove that the two corresponding BANs are equivalent.

As in the case of $\oplus$-BAFs, the proof is done in two steps. We first show that the set of $\oplus$-BACCs of length $m$ and size $(n_1, \ldots, n_m)$ is divided into two classes: the positive class and the negative class, that respectively correspond to the set of BACCs whose interaction graph reduces to a graph where all arcs are positive, and the set of BACCs whose interaction graph reduces to a graph where all arcs are positive except the arc $(i^1_{n_1}, i^1_1)$ that is negative. This result is shown by induction on the index of the "right most" negative arc of the BACC, using the equivalences of Figure 3 (from left to right) to push this negative arc to the left. In a second step, we prove that in the case where $m-1$ is not a multiple of 3, the two classes coincide since one can reduce the negative interaction graph to the positive one, using again a combination of equivalences from Figure 3. □

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

**Lemma 13.** *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 $(101)^{\frac{m-1}{3}}$, if $(m-1) \equiv 0 \mod 3$.*

**Lemma 14.** *A negative $\oplus$-BACC (of length $m \equiv 1 \mod 3$) has no fixed point.*

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 12. 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 $m - 1 \equiv 0 \mod 3$. More precisely 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 Conclusions and perspectives

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. By this means, it contributes to improve our understanding of the wild domain of non-monotonic Boolean automata networks.

The algorithmic approach used to explore the transition graphs of the BANs under study appears as a good strategy when one wants to get an insight of the mechanisms behind the asymptotic behaviours of some networks. In our case, it emphasises the great expressiveness of $\oplus$-BANs and their high level of "(re)activity" (in comparison with $\vee$-BANs for example).

The notion of bisimulation also reveals to be a powerful tool for factorising proofs when it comes to the study of a particular family of BANs. Even though finding a proper set of interaction graph rewritings may be a bit challenging, it results in a very interesting and comprehensive tool that highlights which characteristics of the interaction graphs really matter in the dynamical behaviours of the BANs.

We believe that most of the results we get could be refined or extended to some other types of ($\oplus$)-BANs. For example it should be possible to allow some arcs between or inside the cycles of a $\oplus$-BADC without changing the general shape of its corresponding ATG. These kind of refinement draw a logical line for further works. Another interesting question would be directed to the study and comparison of asymptotic behaviours under different update modes. The algorithms we describe and the ATG we get for strongly connected $\oplus$-BANs with an induced BADC of size greater than 3 suggest that the addition of $k$-synchronism, that is when one allows $k$ automata to update simultaneously, make the set of unreachable configuration disappear if $k$ is greater than the size of the smallest cycle in an induced BADC of the network.

I end this report with great thanks to my supervisors, Kévin Perrot and Sylvain Sené, for their enthusiasm, their availability and the precious advice they gave to me. I also would like to thank them for giving me the chance to attend to the EJCIM-2015 and to write my first paper. Finally I would like to thank people from the LIF for the warm atmosphere of the lab.

## References

[1] J. Aracena, A. Richard, and L. Salinas. Maximum number of fixed points in AND-OR-NOT networks. *Journal of Computer and System Sciences*, 80:1175–1190, 2014.

[2] J. Demongeot, M. Noual, and S. Sené. Combinatorics of Boolean automata circuits dynamics. *Discrete Applied Mathematics*, 160:398–415, 2012.

[3] G. Didier and É. Remy. Relations between gene regulatory networks and cell dynamics in Boolean models. *Discrete Applied Mathematics*, 160:2147–2157, 2012.

[4] E. S. El-Mallah and C. J. Colbourn. The complexity of some edge deletion problems. *IEEE Transactions on Circuits and Systems*, 35:354–362, 1988.

[5] M. Gadouleau, A. Richard, and É. Fanchon. Reduction and fixed points of Boolean networks and linear network coding solvability. 2014. arXiv:1412.5310.

[6] M. Gadouleau, A. Richard, and S. Riis. Fixed points of Boolean networks, guessing graphs, and coding theory. 2014. arXiv:1409.6144.

[7] C. Georgescu, W. J. R. Longabaugh, D. D. Scripture-Adams, E. David-Fung, M. A. Yui, M. A. Zarnegar, H. Bolouri, and E. V. Rothenberg. A gene regulatory network armature for T lymphocyte specification. *Proceedings of the National Academy of Sciences*, 105:20100–20105, 2008.

[8] E. Goles and S. Martínez. *Neural and automata networks: dynamical behavior and applications*, volume 58 of *Mathematics and Its Applications*. Kluwer Academic Publishers, 1990.

[9] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA*, 79:2554–2558, 1982.

[10] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the USA*, 81:3088–3092, 1984.

[11] A. S. Jarrah, R. Laubenbacher, and A. Veliz-Cuba. The dynamics of conjunctive and disjunctive Boolean network models. *Bulletin of Mathematical Biology*, 72:1425–1447, 2010.

[12] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22:437–467, 1969.

[13] P. Koiran. *Puissance de calcul des réseaux de neurones artificiel*. PhD thesis, École normale supérieure de Lyon, 1993.

[14] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Journal of Mathematical Biophysics*, 5:115–133, 1943.

[15] T. Melliti, M. Noual, D. Regnault, S. Sené, and J. Sobieraj. Asynchronous dynamics of Boolean automata double-cycles. In *Proceedings of UCNC*, 2015. In press, arXiv:1310.5747.

[16] T. Melliti, D. Regrault, A. Richard, and S. Sené. On the convergence of Boolean automata networks without negative cycles. In *Proceedings of Automata*, volume LNCS 8155, pages 124–138. Springer, 2013.

[17] L. Mendoza, D. Thieffry, and E. R. Alvarez-Buylla. Genetic control of flower morphogenesis in arabidopsis thaliana: a logical analysis. *Bioinformatics*, 15:593–606, 1999.

[18] M. Noual. Synchronism vs asynchronism in boolean networks. arXiv:1104.4039, 2011.

[19] M. Noual. *Updating automata networks*. PhD thesis, école normale supérieure de Lyon, 2012. http://tel.archives-ouvertes.fr/tel-00726560.

[20] M. Noual, D. Regnault, and S. Sené. About non-monotony in Boolean automata networks. *Theoretical Computer Science*, 504:12–25, 2012.

[21] M. Noual, D. Regnault, and S. Sené. Boolean networks synchronism sensitivity and XOR circulant networks convergence time. In *Full Papers Proceedings of Automata'12*, volume 90 of *Electronic Proceedings in Theoretical Computer Science*, pages 37–52. Open Publishing Association, 2012.

[22] P. Orponen. Computing with truly asynchronous threshold logic networks . *Theoretical Computer Science*, 174:123–136, 1997.

[23] E. Remy, B. Mossé, C. Chaouiya, and D Thieffry. A description of dynamical graphs associated to elementary regulatory circuits. *Bioinformatics*, 19:172–178, 2003.

[24] A. Richard. Negative circuits and sustained oscillations in asynchronous automata networks. *Advances in Applied Mathematics*, 44:378–392, 2010.

[25] A. Richard and J.-P. Comet. Necessary conditions for multistationarity in discrete dynamical systems. *Discrete Applied Mathematics*, 155:2403–2413, 2007.

[26] F. Robert. *Discrete iterations: a metric study*, volume 6 of *Springer Series in Computational Mathematics*. Springer, 1986.

[27] D. Thieffry and R. Thomas. Dynamical behaviour of biological regulatory networks–ii. immunity control in bacteriophage lambda. *Bulletin of mathematical biology*, 57:277–97, 1995.

[28] R. Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42:563–585, 1973.

[29] R. Thomas. Regulatory networks seen as asynchronous automata: a logical description. *Journal of Theoretical Biology*, 1991.

# 6 Appendix

This Appendix compiles the details of the proofs sketched in the core of the document. Lemmas and proofs are presented in the same order.

**Lemma 3** Two BANs $\mathcal{N} = \{f_i\}_{i=1}^n$, $\mathcal{N}' = \{f'_i\}_{i=1}^n$ bisimulate each other if there exists a bijection $\varphi : \{1, \dots, n\} \to \{1, \dots, n\}$ and a set $\{\phi_i : \mathbb{B} \to \mathbb{B}\}_{i=1}^n$ of (non constant) Boolean functions such that for all automata $i$, $\phi_i \in \{id, neg\}$, and for all configurations $x \in \mathbb{B}^n$, $\phi_i(f_i(x)) = f'_{\varphi(i)}(\phi(x))$ where $\phi(x)$ is defined componentwise by $\phi(x)_i = \phi_{\varphi^{-1}(i)}(x_{\varphi^{-1}(i)})$.

*Proof.* This is almost a rephrasing of the definition: Suppose that $\varphi$ and $\{\phi_i : \mathbb{B} \to \mathbb{B}\}_{i=1}^n$ satisfy the conditions above, then

1. The map $\phi$ is a bijection since it is defined componentwise by a set of bijective functions ($\phi_i \in \{id, neg\}, \forall i$).

2. $G_{\mathcal{N}_1}$ and $G_{\mathcal{N}_2}$ are isomorphic by $\phi$ since

$$
\begin{aligned}
x \xrightarrow{W} y \in G_{\mathcal{N}_1} \quad &\Leftrightarrow \quad y_i = \begin{cases} f_i(x) & \text{if } i \in W \\ y_i & \text{otherwise} \end{cases}, \quad \forall i \in \{1, \dots, n\} \\
&\Leftrightarrow \quad \phi_i(y_i) = \begin{cases} \phi_i(f_i(x)) & \text{if } i \in W \\ \phi_i(y_i) & \text{otherwise} \end{cases}, \quad \forall i \in \{1, \dots, n\} \\
&\Leftrightarrow \quad \phi_i(y_i) = \begin{cases} g_{\varphi(i)}(\phi(x)) & \text{if } i \in W \\ \phi_i(y_i) & \text{otherwise} \end{cases}, \quad \forall i \in \{1, \dots, n\} \\
&\Leftrightarrow \quad \phi(y)_{\varphi(i)} = \begin{cases} g_{\varphi(i)}(\phi(x)) & \text{if } \varphi(i) \in \varphi(W) \\ \phi(y)_{\varphi(i)} & \text{otherwise} \end{cases}, \quad \forall i \in \{1, \dots, n\} \\
&\Leftrightarrow \quad \phi(y)_j = \begin{cases} g_j(\phi(x)) & \text{if } j \in \varphi(W) \\ \phi(y)_j & \text{otherwise} \end{cases}, \quad \forall j \in \{1, \dots, n\} \\
&\Leftrightarrow \quad \phi(x) \xrightarrow{\varphi(W)} \phi(y) \in G_{\mathcal{N}_2}
\end{aligned}
$$

□

**Lemma 4** The set of $\oplus$-BADC of size $(n_1, n_2)$ admits exactly one bisimulation class.

*Proof.* This is induced by the fact that the positive $\oplus$-BADC representative bisimulate both the negative and the mixed $\oplus$-BADC representatives: In the first case, the equality $x_1^1 \oplus x_1^2 = \overline{x_1^1} \oplus \overline{x_1^2}$ implies that positive and negative $\oplus$-BADCs are trivially equivalent; in the second case the bijection $\phi(x) = \overline{x}^V$ over the set of configuration satisfies the condition from definition 1, proving that positive $\oplus$-BADCs and mixed $\oplus$-BADCs bismulate each other.

To prove the second statement we need to show that $\phi(F_W(x)) = F'_W(\phi(x)), \forall x \in \mathbb{B}^n$, where $F$ denotes the global functions of the positive $\oplus$-BADC and $F'$ the ones of the negative $\oplus$-BADC. We check that it is the case for each component $i$:

- if $i \notin W$, then $\phi(F_W(x))_i = \overline{F_W(x)_i} = \overline{x_i} = \phi(x)_i = F'_W(\phi(x))_i$
- if $i = o \in W$, $\phi(F_W(x))_i = \overline{F_W(x)_i} = \overline{x_{n_1} \oplus x_{n_2}} = \overline{\phi(x)_{n_1} \oplus \phi(x)_{n_2}} = \phi(x)_{n_1} \oplus \overline{\phi(x)_{n_2}} = F'_W(\phi(x))_i$
- if $o \neq i_j^k \in W$, $\phi(F_W(x))_j^k = \overline{F_W(x)_j^k} = \overline{F_j^k(x)} = \overline{x_{j-1}^k} = \phi(x)_{j-1}^k = f'^k_j(\phi(x)) = F_W(\phi(x))_j^k$

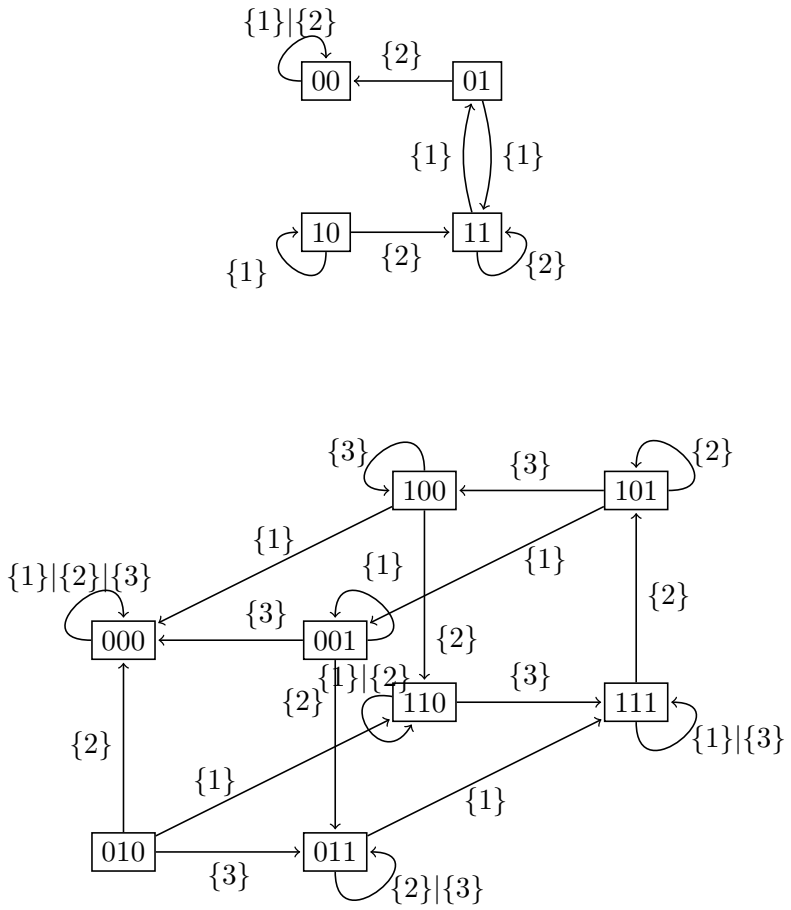



Figure 4: The ATGs of the positive BADCs of size $(1, 2)$ (left) and $(2, 2)$ (right)

□

**Lemma** 8 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 $\oplus$-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 8 holds for one $\oplus$-BADC of each size is enough to prove Lemma 8 completely. Hence in the following we only deal with *positive* $\oplus$-BADC. However, one will notice that the proof below is easy to adjust to any $\oplus$-BADC.

The proof presents an algorithm that explains how to walk from one configuration to an other in the ATG of any positive $\oplus$-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 8 by looking directly at the form of their ATG. These ATGs are drawn in Figure 4 and they all satisfy Lemma 8 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 $\oplus$-BADC) one can reach a configuration $x$ where $x_o = f_o(x)$ and $x_i = \overline{f_i(x)}$ for all $i \neq o$ (*i.e.* $x_o = x_{n_1} \oplus x_{n_2}$ and $x_j^k = \overline{x_{j-1}^k}$ for all $i_j^k \neq o$). 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 $i_j^k$ be the automaton in state 1 that is the closest to $i_{n_1}$ and update every automata on the path from $i_j^k$ 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 $\mathcal{C}_1$ ; if $k = 2$ then the state 1 of $i_j^2$ propagates from $j$ to $n_2$ in $\mathcal{C}_2$ then from 1 to $n_1$ in $\mathcal{C}_1$. The more subtle point is that by the time $o = i_1^1$ 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 $\mathcal{C}_2$ from 1 (*i.e.* $o$) to $n_2$.
   - In a quadratic number of updates, set $\mathcal{C}_1$ 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 $\mathcal{C}_1$ from 1 to $j$ then the ones of $\mathcal{C}_2$ from 2 to $n_2$.

     The invariant is the following : after each iteration, $x^1[n_1, j] = (10)^{(n_1-j)/2}$ and $x_{n_2} = x_j^1 = x_o$ (hence $f_o(x) = x_{n_1} \oplus x_{n_2} = 1 \oplus x_j^1 = \overline{x_j^1}$ ). 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 $x^1[n_1, j+1] = (10)^{\frac{n_1-j+1}{2}}$ and with $f_o(x) = \overline{x_{j+1}^1}$ we end up with $x_j^1 = x_{n_2} = x_o = \overline{x_{j+1}^1}$ and so $x^1[n_1, j] = (10)^{(n_1-j)/2}$.
   - Similarly force $\mathcal{C}_2$ to alternate in a quadratic number of updates (while preserving the alternating configuration in $\mathcal{C}_1$):

     for $j = n_2 - 1$ to 2 do: update the automata of $\mathcal{C}_2$ from 1 to $j$ then the one of $\mathcal{C}_1$ from $n_1$ to 2.

     The invariant is: after each iteration, $x_{n_2}$ is unchanged, $x_2^1 = x_j^2 = x_o$, $f_o(x) \neq x_o$ and $x^1[2, n_2]$ and $x^2[n_2, j]$ are both alternating. The first two statements of this invariant are direct translation of the instructions. The last two require the invariant hypotheses. By the previous point the invariant is satisfied before entering the loop. Hence, right after its update $x_o \neq x_2^1$ and $x_o \neq x_{j+1}^2$. So after its updtate $x_j^2 = x_o \neq x_{j+1}^2$ (hence $x^2[n_2, j]$ is alternating), and updating $\mathcal{C}_1$ in reverse order leaves it alternating. This also restores the fact that $x_o \neq f_o(x)$ since the state of $i_{n_1}$ has been switched with the update of $\mathcal{C}_1$ while the state of $i_{n_2}$ has been left unchanged.
   - By the end of the two previous steps the system is in a configuration such that $f_j^k(x) = \overline{x_j^k}$ for all automata $i_j^k$ but $i_2^1$ and $i_2^2$. The last thing to do to reach a configuration where $f_i(x) = \overline{x_i}$ for all automata $i$ but $o$, is thus to update $\mathcal{C}_1$ and $\mathcal{C}_2$ in reverse order (from $n_1$, respectively $n_2$, to 2) and then update the central automaton $o$. This takes a linear number of updates.

   Hence, the whole sequence takes a quadratic number of updates and it results in one of the following configurations:

   · $(0(10)^{\frac{n_1-1}{2}}, 0(10)^{\frac{n_2-1}{2}})$ if $n_1$ and $n_2$ are odd,

   · $((10)^{\frac{n_1}{2}}, 1(01)^{\frac{n_2-1}{2}})$ if $n_1$ is even and $n_2$ is odd,

· $((01)^{\frac{n_1}{2}}, (01)^{\frac{n_2}{2}})$ if $n_1$ and $n_2$ are even,

· $(1(01)^{\frac{n_1-1}{2}}, (10)^{\frac{n_2}{2}})$ if $n_1$ is odd and $n_2$ is even.

2. Let $x$ denote the resulting configuration, then any configuration $x'$ with at least one automaton $i_j^k$ in stable state (*i.e.* such that $x'^k_j = f_j^k(x')$) is reachable from $x$. Indeed, $x_o = f_o(x)$ and $x_i = \overline{f_i(x)}$ for all $i \neq o$, so in a linear number of updates we can move from the configuration $x$ to the configuration $\hat{x}$ where $\hat{x}_o = x'_o$ and $\hat{x}_i = \overline{f_i(\hat{x})}$ for all $i \notin \{i_k^j, o\}$. This is achieved by following instruction: if $i_k^j \neq o$ and $x'_o \neq x_o$, update $o$ and the automata from $n_k$ to $j$ in $\mathcal{C}_k$.

   Then, reaching $x'$ from $\hat{x}$ is straightforward: one simply needs to switch the state of the automata when necessary:

   - for $j = n_1$ to 2 (in $\mathcal{C}_1$): update the automaton $i_j^1$ if $\hat{x}_j^1 \neq x'^1_j$;
   - for $j = n_2$ to 2 (in $\mathcal{C}_2$): update the automaton $i_j^2$ if $\hat{x}_j^2 \neq x'^2_j$;
   - update the automaton $i_k^j$.

   These updates are efficient since for all $i \notin \{i_k^j, o\}$, if $\hat{x}_i \neq x'_i$ then $x'_i = \overline{x_i} = f_i(\hat{x})$, 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 $i_k^j$, $x'^k_j = f_j^k(x')$, 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 $x = (10^{n_1-1}, 10^{n_2-1})$ to a configuration $x'$ where $x'_i = \overline{f_i(x')}$ for all automata $i \neq o$ (for example the configuration $x' = (0(10)^{\frac{n_1-1}{2}}, 0(01)^{\frac{n_2-1}{2}}$ if $m$ and $n$ are odd) requires at least $\sum_{j=1}^{n_1} j + \sum_{j=1}^{n_2} j = \frac{n_1(n_1-1)}{2} + \frac{n_2(n_2-1)}{2}$ updates, which is in $\theta((n_1+n_2)^2)$.

□

*Remark.* 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 $x' = \overline{x}^{\mathcal{C}_i}$ for $|\mathcal{C}_i| > 1$ (since in that case the state of the first automaton of $\mathcal{C}_{1-i}$ is stable). Then for all automaton $j$ of $\mathcal{C}_i$, $f_j(x') = \overline{x'_j} = x_j$, so the synchronous update of $\mathcal{C}_i$ changes the configuration of the system from $x'$ to $x$.

**Lemma 9** 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 $x'$ such that $j$ is unstable (hence can be switched) in $x'$ that is reachable from $x$.

*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 $x' = \overline{x}^k$ such that $x'_i = x_i = f_i(x) = \bigoplus_{j \in I_i} \sigma_j(x_j) = \overline{\sigma_k(x_k)} \oplus \left(\bigoplus_{j \in I_i \setminus \{k\}} \sigma_j(x_j)\right) = \overline{\sigma_k(x'_k)} \oplus \left(\bigoplus_{j \in I_i \setminus \{k\}} \sigma_j(x'_j)\right) = \overline{\bigoplus_{j \in I_i} \sigma_j(x'_j)} = \overline{f_i(x')}$, that is, $i$ is unstable in $x'$.

So let $i$ and $j$ be two automata as described in Lemma 9, 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** 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

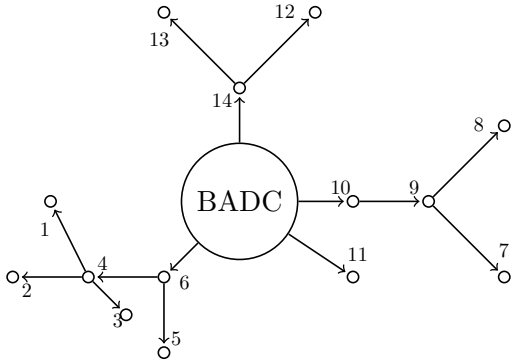
*Proof.* Let $D$ be an induced BADC of size greater than 3 in the BAN $\mathcal{N}$ and let $x$ and $x'$ respectively be the initial configuration and the target configuration described in Theorem 3. The configuration $x$ is not stable so, by Lemma 9, it is possible to go from $x$ to a configuration $y$ where one automaton of $D$, hence $D$, is not stable. Then, using Lemmas 9 and 8, we claim that it is possible to set the state of every automata $i$ outside of $D$ to its value in $x'$ while keeping $D$ in an unstable configuration.

The idea is as follows: let $i$ be an automaton that is not in $D$ and let $p = i_0 i_1 \dots i_k$ be a shortest path (in the interaction graph of $\mathcal{N}$) from $D$ to $i_k = i$. Then, applying the algorithm from Lemma 8, we know how to reach a configuration where $i_0$ is unstable and so, using the algorithm from Lemma 9, 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 $D$, then we can use it repetitively on the automata outside of $D$ to reach a configuration where $D$ is unstable and where all automata outside of $D$ are in the state specified by $x'$. Once this is done we only need to set $D$ to its right value to reach $x'$. Since $D$ is unstable, this can be done by using the algorithm from Lemma 8, assuming that the restriction of $x'$ to $D$ is not unreachable for $D$ ($D$ 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 8 when the stable state of the target configuration is not the central node $o$: if $i$ is an automaton of $\mathcal{N}$ such that $f_i(\overline{x'^i}) = x'_i$, and if $p = i_0 \dots i_k$ is a shortest path from $i = i_0$ to $D$, 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 $D$ is reachable for $D$), 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 $i_\ell$ 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 $f_i(\overline{x'^i}) = x'_i$.

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)* $\hat{x}^\ell_j = y^{\ell-1}_j$ if $j \notin \{i_{\ell-1}, i_\ell\}$, $\hat{x}^\ell_{i_\ell} = \overline{x'_\ell}$, and $\hat{x}^\ell_{i_{\ell-1}}$ is the solution of the equation $f_{i_{\ell-1}}(\hat{x}^\ell) = \overline{\hat{x}^\ell i_{\ell-1}}$.

Actually, setting the automata outside of $D$ to their state in $x'$ cannot be done in any order. Indeed, the algorithm from Lemma 9 often requires to switch the state of some automata outside of $D$. 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 $D$ 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 $D$ 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 9 that ensures that the instability of $D$ as well as the state of the automata that are not in $D$ or on the path from $D$ 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 $D$ to the automata to be set up, and let $j \neq i_0$ be an influencer of $i_0$ in $D$ (*i.e.* $j \in (B \setminus \{i_0\}) \cap I_{i_0}$). Then, since $D$ is supposed to be unstable in $x$, one can use Lemma 8 to put the system in a configuration $y$ where $i_0$ is unstable, and where $y_j$ is such that $y_j = \overline{f_j(\overline{y}^{i_1})}$ if there is an arc from $i_1$ to $i_0$, and $y_j = \overline{f_j(\overline{y}^{\{i_0,i_1\}})}$ if there are no arcs from $i_1$ to $i_0$. This is possible since $D$ is of size at least 3, and so one can ask a third automaton of $D$ (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 $D$.
From there one can start applying Lemma 9: 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 $\mathcal{N}$ in a configuration $y'$ such that $D$ is unstable. Indeed,

- either nothing happened ($\ell > 1$) and so $D$ 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 $y'_j = y_j = \overline{f_j(\overline{y}^{i_1})} = \overline{f_j(y')}$ 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 $y'_j = y_j = \overline{f_j(\overline{y}^{\{i_0,i_1\}})} = \overline{f_j(y')}$ 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 $D$ is an induced BADC of size 3 and $j$ has been chosen to be the predecessor of $i_0$ different from $i_0$) so $f_j(\overline{y}^{\{i_1\}}) = f_j(\overline{y}^{\{i_0,i_1\}})$, so $y'_j = \overline{f_j(\overline{y}^{\{i_0,i_1\}})}$ 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 $D$ is unstable in $y'$, we can use Lemma 8 again to reach a configuration $y''$ such that $y''_{i_0} = \overline{f_{i_0}(\overline{y'}^{\{i_{\ell'},\dots,i_{n-1}\}})}$ and $y''_i = y'_i$ for all automata $i$ that are not in $D$. Moreover, since $p$ was chosen to be a shortest path, no automaton in $D$ 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 9 (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 $D$ is unstable since $i_0$ has state $\overline{f_{i_0}(\overline{y'}^{\{i_{\ell'},\dots,i_{n-1}\}})}$. This last algorithm concludes the proof of Theorem 3. □

**Lemma 12** The set of $\oplus$-BACC of length $m$ and size $(n_1, \dots, n_m)$ 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 12 using the equivalences presented in Figure 3. 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 3 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 an other one, to prove that the two corresponding BANs are equivalent.

The equivalences *(1)* and *(2)* only translate the well known identities $b_1 \oplus b_2 = \overline{b_1} \oplus \overline{b_2}$ and $\overline{b_1} \oplus b_2 = b_1 \oplus \overline{b_2}$ 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):

*Proof of Equivalence* (3) *in Figure 3.* Let $\mathcal{N} = \{f_i\}$ and $\mathcal{N}' = \{f'\}$ be two $\oplus$-BAN whose interaction graphs that only differ by the pattern shown in Equivalence *(3)*. We denote by $\mathcal{C}_1$, $\mathcal{C}_2$ (respectively $o_1$,$o_2$) the two cycles (respectively intersection automata) of the pattern and by $\mathcal{C}_2^u$the upper helf cycle of $\mathcal{C}_2$. We are going to prove that $\mathcal{N}$ bisimulates $\mathcal{N}'$ by using the conditions from Lemma 3: 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 $\mathcal{C}_1, \mathcal{C}_2^u$ 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 $\mathcal{C}_1 \cup \mathcal{C}_2^u \cup \{o_1, o_2\}$ since we use the identity everywhere. Then, if $i \in \mathcal{C}_1 \cup \mathcal{C}_2^u$, we also have $\phi_i(f_i(x)) = \phi_i(pred(i)) = \overline{pred(i)} = \phi_{pred(i)}(pred(i)) = f'_i(\phi(x))$. So it only remains to check that the equality holds for the automata $o_1$ and $o_2$. This is the case since :

1. $\phi_{o_1}(f_{o_1}(x)) = \phi_{o_1}(\overline{pred_1(o_1)} \oplus pred_2(o_1)) = \overline{\overline{pred_1(o_1)} \oplus pred_2(o_1)}$
$= \overline{\phi_{pred_1(o_1)}(pred_1(o_1)) \oplus \phi_{pred_2(o_1)}(pred_2(o_1))} = f'_{o_1}(\phi(x)),$ and
2. $\phi_{o_2}(f_{o_2}(x)) = \phi_{o_2}(\overline{pred_1(o_2)} \oplus pred_2(o_2)) = \overline{pred_1(o_2)} \oplus pred_2(o_2)$
$= \phi_{pred_1(o_2)}(pred_1(o_2)) \oplus \phi_{pred_2(o_2)}(pred_2(o_2)) = f'_{o_2}(\phi(x)).$

□

Given the set of equivalences of Figure 3, we prove Lemma 12 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 $i^1_{n_1}$ to $o_1 = i^1_1$. 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 $(i^1_{n_1}, i^1_1)$ 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.

*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 the equivalences *(1)* and *(2)* from Figure 3 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, $(i^k_{n_k}, o_k)$.

Then, by induction on the position of the "right most" negative arc, we use the 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 the equivalences *(3)* and *(4)* reduce the four base cases (++,+−,−+,−−) to two: the positive case (++) and the negative case (−+). □

*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 the equivalences *(6)* and *(8)* alternatively so as to push this negative path to the right. Every time we apply the 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 3 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. □

**Lemma 13** A positive $\oplus$-BACC of length $m$ and size $n$ has a unique fixed point, $0^n$, if $m-1$ is not a multiple of 3, and has two fixed points, $0^n$ and $(101)^{\frac{m-1}{3}}$, if $(m-1) \equiv 0 \mod (3)$.

*Lemma 13.* In a stable configuration every nodes of a given nude 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 $0^m$ 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 the following:

**Lemma 14** 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 13). So there are no stable configurations for the negative $\oplus$-BACC of length $m \equiv 1 \mod (3)$. □