Intrinsic Universality in Automata Networks III: On Symmetry versus Asynchrony

Martín Ríos Wilson, Guillaume Theyssier

Intrinsic Universality in Automata Networks III: On Symmetry versus Asynchrony*

Martín Ríos-Wilson[1] and Guillaume Theyssier[2]

[1]Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez
[2]Aix-Marseille Université, CNRS, I2M (UMR 7373), Marseille, France

October 21, 2024

Abstract

An automata network is a finite assembly of interconnected entities endowed with a set of local maps defined over a common finite alphabet. These relationships are represented through an interaction graph. Together with the local functions, an assignment known as an update scheme directs the evolution of the network by updating specific subsets of entities at discrete time steps. Despite the scrutiny of interaction graphs and update schemes, their profound impact on automata network dynamics remains largely open. This work investigates the intricate interplay between these aspects, with a focus on how update schemes can counterbalance constraints stemming from symmetric local interactions. This paper is the third of a series about intrinsic universality, a notion that assesses both dynamical and computational complexity, encompassing transient behaviors, attractors, and prediction or reachability problems. We consider four update modes—parallel, block-sequential, local clocks, and general periodic— along with several families of symmetric signed conjunctive boolean networks defined by local constraints on signs. Our main result is to show a diagonal complexity leap in this two-dimensional landscape: the stronger the local constraints the higher the level of asynchrony required to obtain intrinsic universality or increase in complexity. We also show how in some cases asynchronism allows to simulate directed interactions from undirected ones with the same local rules.

1 Introduction

Automata networks are (finite) graphs in which nodes possess states from a finite set denoted as Q. These nodes are equipped with local transition maps, which dictate the evolution of their states during discrete time steps based on the states of neighboring nodes. The concept of automata networks was first introduced during the 1940s [18]. Since then, many applications have been studied in diverse fields, spanning from modeling biological networks [24, 17] to distributed computational models [7, 3, 8, 26, 25]. An automata network is completely described by the action of its global map $F : Q^V \to Q^V$, which defines how the entire set of nodes V within the network evolves collectively. Initially, considering these models merely as discrete functions might seem sufficient for comprehending their properties. However, limiting the study solely to this global map overlooks two crucial aspects, as explored in automata network theory [5, 6, 4, 2, 9, 1, 21]: the interaction graph (depicting effective dependencies among nodes) and the update schedule (dictating the method by which local transition maps are applied during evolution). Indeed, many applications involve constrained interaction graphs, and the update schedule deviates from synchronicity (where all local maps are applied simultaneously at each time step). Imposing restrictions on the network of interactions could constraint limit potential behaviors. Conversely, exploring different update schedules that allow for asynchrony in local updates could extend the range of possible global behaviors beyond those captured by the full synchronous application of global map F.

The present paper is the third of a series that aims to study the concept of intrinsic universality in automata networks. In the first paper [19], basic concepts and definitions are presented together with consequences of universality. In the second paper [20], we develop a proof technique based on an operation of glueing to prove intrinsic

*This research was partially supported by French ANR project FANs ANR-18-CE40-0002 (G.T., M.R.W.), ECOS project C19E02 (G.T., M.R.W.), STIC-AMSUD 22-STIC-02 (G.T., M.R.W.) and ANID FONDECYT Postdoctorado 3220205 (M.R-W).

universality in a concrete family or simulation of another family. Finally, the present paper is an application of these concepts and tools to study and measure the impact of asynchronism in automata networks. It focuses on the interplay between constraints in the interaction graph and extension of behaviors through asynchrony in the update schedule applied. As we show in the next lines, this type of phenomenon emerges naturally in the study of a particular family of (boolean) automata networks: the majority networks.

The effects of asynchronism in majority dynamics To illustrate the main question behind our work, let us briefly consider Boolean majority dynamics: each node holds a state 0 or 1 and changes its state at each time step to take the one which has the most occurrences among its neighbors (no change in case of tie). This majority dynamics can be defined on any graph. A seminal result [10, 16] shows that majority dynamics on undirected graphs cannot have periodic orbits of period more than 2, and that they have polynomially bounded transients; this implies the existence of a polynomial time algorithm to predict the future of a node from any given initial configuration [12].

Considering undirected graphs here can be seen as a property of symmetry of interactions: the influence of a node on a neighbor is the same as the influence of the neighbor on it. It is easy to show that without this symmetry condition, majority networks have no limitation, neither in dynamics complexity (it can have exponential transients and periods), nor in computational complexity (predictions becomes as hard as for arbitrary automata networks). On the other hand, symmetric majority networks under partially asynchronous updates (precisely block-sequential update modes) were shown to have super-polynomial periodic orbits and a PSPACE-complete prediction problem [14, 3]. In a nutshell, the unconstrained family of majority networks is rich and complex, constraining it to symmetric networks radically reduces its complexity, but adding asynchrony to the symmetry constraint allows to recover the initial complexity.

Our results In the two first papers of this series [19, 20] we developed a notion of universality, that implies both dynamical and computational complexity, together with a proof technique to show that some family is universal. We also introduced the formalism of CAN (concrete automata network) which allows to define families of automata networks by just giving a labeling of the underlying communication graph. This formalism captures many well-studied family ranging from majority and threshold dynamics to linear networks and reaction-diffusion networks (see [19] for an overview on these families). A key point here is that a CAN family can be considered either on undirected graphs (symmetric version) or on arbitrary graphs (asymmetric version).

The present paper builds upon these tools to tackle the main question introduced above in the two following precise forms that revolve around the question of symmetry breaking by asynchrony:

- how a non-universal symmetric CAN family can become universal when asynchrony is added?
- how asynchrony allows a symmetric CAN family to simulate its asymmetric counterpart?

Moreover, in both cases, we aim at understanding 'how much asynchrony' is required to achieve the result, *i.e.* how far from the fully synchronous update mode one has to go.

The results of this paper can be divided in two main sections: first we introduce the definition of *asynchronous extensions* which is a formalism that allow us to model the asynchronous system via projections of synchronous AN defined over a larger alphabet. Then, we use this formalism (together with the results of the previous papers of the series) to present a case study on a particular family of automata networks. Concretely speaking, our results can be summarized in Table 1 (and the complete detailed version in Table 2). We present a case study based on the family of signed conjunctive networks (which is a CAN family). In this family, a natural local constraint that one can study (and parameterize) is the number of positive and negative links at each node.

More precisely, one can limit the number of negative edges that are incident to each node in the network. We distinguish three cases in this context: the all-positive case, the all-negative case and locally-positive case (at least one positive edge must be incident to each node). On the other hand, we study four different update schemes. In particular, we go from the synchronous update scheme to the general periodic case by adding different elements of asynchronism. Thus, we have a two dimensional space for our analysis: the sign of the connections and the asynchronism. For each fixed sub-family (given by the local constraints on the local connections between the nodes) we establish that there exists an update scheme in the hierarchy for which the family becomes universal or is able to simulate its non-symmetric counterpart. In addition, we show 'lower bounds' on the type of asynchronism required

	Synchronous	Block sequential	Local clocks	Periodic
	**simple**	**universal**	**universal**	**universal**
	**simple**	**simple**	**universal**	**universal**
	**simple**	**simple**	**simple**	**desymm**

Table 1: Summary of the main results on complexity of the dynamics of the network families studied in the current chapter, depending on different update schemes. The word *simple* stands for polynomial both in the sense of the complexity and the dynamics (i.e. prediction problem can be solved in polynomial time in the size of the network and attractors/transient times are polynomial). The word *desymm* stands for *desymmetrization*. For more details, see Table 2

which is illustrated by the 'diagonal' appearing in Table 1, precisely: all-negative networks require block sequential (first level of the hierarchy) to become universal and all-positive requires general periodic for desymmetrization, while locally positive networks require the intermediate level of asynchrony from local clocks update schedules.

Finally, we tackle the question on whether this desymmetrization phenomena that was observed for conjunctive networks takes place on other families or if it is a particular case for this family. We show that there is generally no link to expect between periodic update schedule on a symmetric CAN family and its non-symmetric version by two opposite examples: a family such that no periodic extension can simulate its non-symmetric version and a family such that its non-symmetric version is not capable of simulating any of its periodic extensions.

Organization of the paper

In Section 2 we present the basic definitions and we will also recall some results from the previous part of the series. In Section 3 we present the main formalism that we will use to work with asynchronism: *asynchronous extensions*. In Section 4 we present a case study on symmetric signed conjunctive networks and main results. Finally, in Section 5 we explore the link between asynchronism and symmetry by showing some counterexamples to the desymmetrization phenomena.

2 Preliminaries

2.1 Concrete automata networks

A *graph* is a pair $G = (V, E)$ where V and E are finite sets satisfying $E \subseteq V \times V$. We will call V the set of *nodes* and the set E of *edges*. We call $|V|$ the *order* of G and we usually identify this quantity by the letter n. Usually, as E and V are finite sets we will implicitly assume that there exists an ordering of the vertices in V from 1 to n (or from 0 to $n-1$). Sometimes we will denote the latter set as $[n]$. If $G = (V, E)$ and $V' \subseteq V, E' \subseteq E$ we say that G' is a *subgraph* of G.

Given a (non-directed) graph $G = (V, E)$ and two vertices u, v we say that u and v are neighbors if $(u, v) \in E$. Remark that abusing notations, an edge (u, v) is also denoted by uv. Let $v \in V$, we call $N_G(v) = \{u \in V : uv \in E\}$ (or simply $N(v)$ when the context is clear) the set of neighbors (or *neighborhood*) of v and $\delta(G)_v = |N_G(v)|$ to the *degree* of v. Observe that if $G' = (V', E')$ is a subgraph of G and $v \in V'$, we can also denote by $N_{G'}(v)$ the set of its neighbors in G' and the degree of v in G' as $\delta(G')_v = |N_{G'}(v)|$. In addition, we define the *closed neighborhood* of v as the set $N[v] = N(v) \cup \{v\}$ and we use the following notation $\Delta(G) = \max_{v \in V} \delta_v$ for the *maximum degree* of G. Additionally, given $v \in V$, we will denote by E_v to its set of *incident edges*, i.e., $E_v = \{e \in E : e = uv\}$. We will use the letter n to denote the order of G, i.e. $n = |V|$. Also, if G is a graph whose sets of nodes and edges are not

specified, we use the notation $V(G)$ and $E(G)$ for the set of vertices and the set of edges of G respectively. In the case of a directed graph $G = (V, E)$ we define for a node $v \in V$ the set of its *in-neighbors* by $N^-(v) = \{u \in V : (u, v) \in E\}$ and its *out-neighbors* as $N^+(v) = \{u \in V : (v, u) \in E\}$. We have also in this context the indegree of v given by $\delta^- = |N^-(v)|$ and its *outdegree* given by $\delta^+ = |N^+(v)|$

We call a *configuration* x to any element $x \in Q^V$. If $S \subseteq V$ we define the restriction of a configuration x to V as the function $x|_S \in Q^S$ such that $(x|_S)_v = x_v$ for all $v \in S$. In particular, if $S = \{v\}$, we write x_v. In addition, we abuse notation and we call a function $c : Q^A \mapsto Q$ where $A \subseteq V$ a *partial configuration*.

A *multiset* over Q is a map $m : Q \to \mathbb{N}$ (recall that $0 \in \mathbb{N}$). In addition, a k-bounded multiset over Q is a map $m : Q \to [k] = \{0, \dots, k\}$, the set of such multisets is denoted $[k]^Q$. For instance a multiset in $[2]^Q$ is actually a set. Note that when Q is finite (which will always be the case below), any multiset is actually a bounded multiset. To any (partial) configuration $c \in Q^A$, we associate the multiset $m(c)$ which to any $q \in Q$ associates its number of occurrences in c, *i.e.*

$$m(c) = q \mapsto \#\{a \in A : c(a) = q\}.$$

Definition 1. *Given a directed graph $G = (V, E)$, a vertex label map $\lambda : V \to (Q \times \mathbb{N}^Q \to Q)$ and an edge label map $\rho : E \to (Q \to Q)$, we define the concrete automata network (CAN) $\mathcal{A} = (G, \lambda, \rho)$. A* family of concrete symmetric automata networks *(CAN family) $\mathcal{F}$ is given by a family of graphs $\mathcal{G}$, an alphabet Q and a set of local labeling constraints $\mathcal{C} \subseteq \Lambda \times R$ where $\Lambda = \{\phi : Q \times \mathbb{N}^Q \to Q\}$ is the set of possible vertex labels and $R = 2^{\{\psi : Q \to Q\}}$ is the set of possible sets of neighboring edge labels. We say a CAN (G, λ, ρ) belongs to $\mathcal{F}$ if $G \in \mathcal{G}$ and for any vertex v of G with incident edges E_v it holds $(\lambda(v), \rho(E_v)) \in \mathcal{C}$.*

Note that the labeling constraints defining a CAN family are local and in that sense, the communication graph structure does not have to be a priori fixed. More precisely, there is no a priori constraints in the choice of the class $\mathcal{G}$.

However, in this work we focus in three cases exclusively:

1. $\mathcal{G}$ is the class of undirected graphs.
2. $\mathcal{G}$ is the class of bounded degree graphs.
3. $\mathcal{G}$ is the class of all connected graphs.

In the first case, i.e., in the case in which G is an undirected graph, a CAN is called *Concrete Symmetric Automata Network* or CSAN.

Many interesting examples in the literature can be described as CSAN families. We refer the reader to [19] for more details.

If $\mathcal{F}$ is a CSAN family, we call the non-symmetric version of $\mathcal{F}$ the family of CAN defined by the extension of $\mathcal{F}$ to directed graphs. More precisely, if $\mathcal{F} = (\mathcal{G}, \mathcal{C})$ where $\mathcal{G}$ is a family of undirected graphs, its extension will be a family $(\overrightarrow{\mathcal{G}}, \mathcal{C})$ where $\overrightarrow{\mathcal{G}}$ is a family of graphs containing all the posible orientations for the graphs in $\mathcal{G}$. We note the non-symmetric version of $\mathcal{F}$ by $\overrightarrow{\mathcal{F}}$.

Observe that a CAN induces a discrete dynamical system via the semantics of labels defined above. In fact, labels on edges are state modifiers, and labels on nodes give a map that describes how the node changes depending on the set of sates appearing in the neighborhood, after application of state modifiers.

Formally speaking, we use the following notation: Let $\Sigma(Q)$ be the set of permutations on alphabet Q. Given $\sigma \in \Sigma(Q)^V$ an assignation of permutations for each node in the graph and $x \in Q^A$ a partial configuration with $A \subseteq V$, we define the partial configuration $x^\sigma : A \mapsto Q$ as $x^\sigma(a) = \sigma(a)(x(a))$ where $\sigma(a)$ denotes the permutation associated to vertex a by the mapping σ.

We remark that the dynamics induced by a CSAN can be very different from the one induced by a CAN. In Figure 1 we show an example of a non-directed network and a directed network. Both networks are conjunctive networks, i.e. the local rule in each case is the same and it is an AND function of all incoming neighbors' states (equivalently it is the function giving the minimum state among the incoming neighbors over state set $\{0, 1\}$). Observe that the initial condition is the same (white circles are nodes in state 1 and black nodes are in state 0). However,

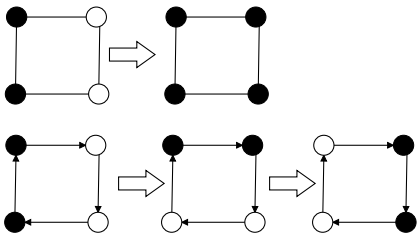


Figure 1: Example of two conjunctive networks (each node takes the minimum state among incoming neighbors). In the upper part a non-directed network and in the lower part a directed network. The nodes in black are in state 0 and the nodes in white are in state 1.

the symmetric version reaches a configuration in which any node is in state 0 (and thus, it cannot change) but the directed one exhibits a periodic orbit.

Now we define the dynamics. We do so by defining a function $F : Q^V \to Q^V$ which is defined in folklore as a *global function* for the network.

Definition 2. *Given a CAN (G, λ, ρ), its associated global map $F : Q^V \to Q^V$ is defined as follows. For all node $v \in V$ and for all $x \in Q^n$:*

$$F(x)_v = \lambda_v(x_v, m((x|_{N^+(v)})^{\rho(E_v)})).$$

The function F is also called an *abstract automata network*. For more detail see [19].

Given an initial configuration $x \in Q^V$, we define the *orbit* of x as the sequence $\mathcal{O}(x) = (F^t(x))_{t\geq 0}$. We define the set of *limit configurations* or *recurrent configurations* of F as $L(F) = \bigcap_{t\geq 0} F^t(Q^V)$. Observe that since Q is finite and F is deterministic, each orbit is eventually periodic, i.e. for each $x \in Q^V$ there exist some $\tau, p \in \mathbb{N}$ such that $F^{t+p}(x) = F^t(x)$ for all $x \in Q^V$ and $t > \tau$. Note that if x is a limit configuration then, its orbit is periodic. In addition, any configuration $x \in Q^V$ eventually reaches a limit configuration in finite time. We denote the set of orbits corresponding to periodic configurations as $\mathrm{Att}(F) = \{\mathcal{O}(x) : x \in L(F)\}$ and we call it the set of *attractors* of F. Given an orbit $\mathcal{O}(x) = (F^t(x))_{t\geq 0}$ we use the notation $x(t) = F^{t-1}(x)$ for $t \geq 1$, to denote the elements in the sequence which defines the orbit. For the case $t = 0$, we write $x(0) = F^0(x) = x$. Moreover, we abuse notation and sometimes we call an orbit $\mathcal{O}(x)$ as $x(\cdot)$ or simply x when the context is clear.

We define the *global period* or simply the *period* of an orbit $\overline{x}(\cdot) \in \mathrm{Att}(F)$ by $p(\overline{x}) = \min\{p \in \mathbb{N} : \overline{x}(p) = \overline{x}(0)\}$. If $p(\overline{x}) = 1$ we say that $\overline{x}$ is a *fixed point* and otherwise, we say that $\overline{x}$ is a *limit cycle*.

2.2 Signed conjunctive automata networks

The CSAN family at the heart of this paper is the family of *symmetric signed conjunctive networks* and three subfamilies of it defined below. From now on, since we are only working with networks defined over non-directed graphs, we are going to call them just *signed conjunctive networks* or simply *conjunctive networks*. We define for $Q = \{0, 1\}$ the maps $\mathrm{Id} : Q \mapsto Q$ and $\mathrm{Switch} : Q \mapsto Q$. given by $\mathrm{Id}(q) = q$ and $\mathrm{Switch}(q) = 1 - q$.

Definition 3. *We define symmetric conjunctive networks as the CSAN family where all edges are labeled by the identity map or the switch map, i.e, $\rho(e) \in \{Id, Switch\}$ for all $e \in E$, and all nodes have the same* conjunctive *local map given by*

$$\lambda(v)(q)(X) = \begin{cases} 0 & \text{if } X(0) \geq 1, \\ 1 & \text{else.} \end{cases}$$

for any $v \in V, q \in Q$ and $X \in \mathbb{N}^Q$.

An edge $e \in E$ such that $\rho(e) = \mathrm{Id}$ is called a *positive edge* and a an edge $e \in E$ such that $\rho(e) = \mathrm{Switch}$ is called a *negative edge*.

We classify symmetric conjunctive networks according to the type of labels assigned to the edges. If no constraint is given then, we call the family signed conjunctive networks. If all the edges are negative, we call them *all-negative*

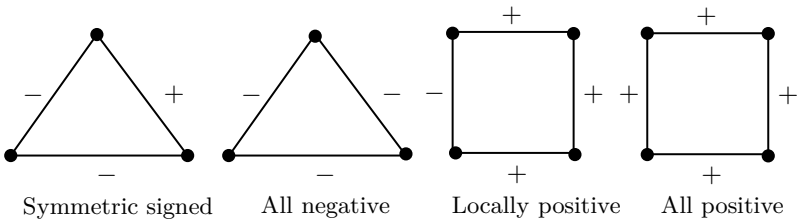


Figure 2: Symmetric signed conjunctive networks

networks. More precisely, a conjunctive network (G, λ, ρ) is all-negative if $\forall e \in E, \rho(e) = \text{Switch}$. If we have a local constraint forcing the network to have at least one edge in each neighborhood labeled as the identity map, i.e., at least one edge is positive, we call them *locally positive.* Formally, a conjunctive network (G, λ, ρ) is locally positive, if for each $v \in V$ there exists one edge $e \in E_v$ such that $\rho(e) = \text{Id}$. Finally, if for some conjunctive network (G, λ, ρ), all its edges are labeled by the identity map, i.e. $\forall e \in E, \rho(e) = \text{Id}$, we call it *all positive* conjunctive network (see Figure 2). We will refer, in the next lines, the families of the previously described networks as *all-negative networks*, *locally positive networks* and *all-positive networks.*

In particular, we will use the notation $\mathcal{F}_{\text{neg}}, \mathcal{F}_{\text{locally-pos}}$ and $\mathcal{F}_{\text{pos}}$ to denote the all-negative, the locally positive and the all-positive families respectively.

2.3 Intrinsic simulations and universality

In [19], we introduced a formalism of intrinsic simulation and universality between family of automata networks, and a proof technique to obtain universality results. To put it concisely, this approach allows to derive many results on the computational and dynamical complexity of a family from a single proof on a finite list of finite automata networks from the family. We will briefly recall the main ingredients here in the context of concrete automata networks.

At the core of the approach is the notion of simulation between individual automata networks: orbits of the simulated network are embedded into orbits of the simulator via a block encoding (one node is simulated by a group of node, and to one state corresponds a pattern on this group of nodes, *i.e.* a partial configuration on this group of nodes).

Definition 4. *Let $F : Q_F^{V_F} \to Q_F^{V_F}$ and $G : Q_G^{V_G} \to Q_G^{V_G}$ be abstract automata networks. A* block embedding *of $Q_F^{V_F}$ into $Q_G^{V_G}$ is a collection of blocks $D_i \subseteq V_G$ for each $i \in V_F$ which forms a partition of V_G together with a collection of patterns $p_{i,q} \in Q_G^{D_i}$ for each $i \in V_F$ and each $q \in Q_F$ such that $p_{i,q} = p_{i,q'}$ implies $q = q'$. This defines an injective map $\phi : Q_F^{V_F} \to Q_G^{V_G}$ by $\phi(x)_{D_i} = p_{i,x_i}$ for each $i \in V_F$. We say that G simulates F via block embedding ϕ if there is a time constant T such that the following holds on $Q_F^{V_F}$:*

$$\phi \circ F = G^T \circ \phi.$$

From there, simulation between families and universality are intuitively clear: a family $\mathcal{F}_1$ simulates a family $\mathcal{F}_2$ if individual automata networks of $\mathcal{F}_2$ are uniformly simulated by those of $\mathcal{F}_1$, and a family is universal if it can simulate all automata networks. For these notions to be effective with respect to the analysis of computational complexity of decision problems, one has to be a bit careful with the details.

Definition 5. *A concrete family $\mathcal{F}_1$ simulates a concrete family $\mathcal{F}_2$ in time $T(n)$ and space $S(n)$ if for any automata network $(G_2, \lambda_2, \rho_2) \in \mathcal{F}_2$, an automata network $(G_1, \lambda_1, \rho_1) \in \mathcal{F}_1$ that simulates it can be produced in DLOGSPACE (G_1 is output by a DLOGSPACE transducer on input G_2) where G_1 has size $S(n)$ and the time constant of the simulation is $T(n)$ where n is the size of G_2. A concrete family is* universal *if it can simulate any concrete family in polynomial space and time. A concrete family is* strongly universal *if it can simulate any concrete bounded degree family in linear space and constant time.*

Remark 1. *In [19] a definition of simulation between families is given in a more general context (not necessarily concrete families), therefore the definition of (strong) universality is stated differently. However, concerning concrete families, the definition given above is equivalent as the one of [19] because there is a concrete family (the "game of life" family) which has been proven strongly universal in [20, Theorem 18]. More precisely, the "game of life" family is universal and a bounded degree variant of it is strongly universal by [20, Lemma 13 and 14].*

2.4 Consequences of universality

Universality might be interesting to study per se, but we are mainly using it for two reasons: first it occurs rather naturally in concrete families as we will show later, and second, it implies both dynamical and computational complexities. To simplify, we will only consider 3 decisions problems here that correspond to prediction of node states and reachability in orbits.

Problem (Unary Prediction, U-PRED)**.**

Parameters: *a concrete family $\mathcal{F}$ over alphabet Q*

Input:

1. *a concrete automata network $\mathcal{A}$ over alphabet Q with global map F such that $\mathcal{A} \in \mathcal{F}$,*
2. *a node v*
3. *an initial condition $x \in Q^V$,*
4. *a state $q \in Q$,*
5. *a natural number t represented in unary.*

Question: $F^t(x)_v = q$*?*

Note that since t is given in unary, the sequence of configurations $x, \dots, F^t(x)$ is computable in polynomial time, so this problem U-PRED. This is no longer the case if we give t in binary as in the following variant.

Problem (Binary Prediction, B-PRED)**.**

Parameters: *a concrete family $\mathcal{F}$ over alphabet Q*

Input:

1. *a concrete automata network $\mathcal{A}$ over alphabet Q with global map F such that $\mathcal{A} \in \mathcal{F}$,*
2. *a node v*
3. *an initial condition $x \in Q^V$,*
4. *a state $q \in Q$,*
5. *a natural number t represented in binary.*

Question: $F^t(x)_v = q$*?*

In problem B-PRED, t can be an exponentially large number, but, in order to compute $F^t(x)$ it is not necessary to memorize the whole sequence of configurations $x, \dots, F^t(x)$. It is actually sufficient to memorize one configuration to compute the next one, so that B-PRED is a polynomial space problem. Another polynomial space problem is the classical reachability problem between configurations.

Problem (Reachability, REACH)**.**

Parameters: *a concrete family $\mathcal{F}$ over alphabet Q*

Input:

1. *a concrete automata network $\mathcal{A}$ over alphabet Q with global map F such that $\mathcal{A} \in \mathcal{F}$*
2. *an initial configuration $x \in Q^V$.*
3. *a target configuration $y \in Q^V$.*

Question: *is there some t such that $F^t(x) = y$?*

We can now state the main consequences of (strong) universality, that will be used as necessary conditions for universality in the remaining of the paper.

Theorem 6. *If a concrete family $\mathcal{F}$ is universal, then problem* U-PRED *is PTIME-complete, and* B-PRED *and* REACH *are PSPACE-complete. Moreover the family admits superpolynomial transients and limit cycles (i.e. length of the largest transient and limit cycle grows faster than any polynomial in the number of nodes). If $\mathcal{F}$ is strongly universal, the family admits exponential transients and limit cycles.*

Proof. The complexity part follows by [19, Corollary 1] and the dynamics part from [19, Theorem 15]. □

2.5 A sufficient condition for universality: coherent gadgets

The universality of a family involves infinitely many simulations between individual automata networks. In [20] we developed a proof technique that boils down to showing the existence of a finite set of networks of the families having a finite set of compatible pseudo-orbits. A pseudo-orbit is a sequence of configurations where the state change from one configuration to the next follows the automata network rule for a fixed subset of nodes, and is arbitrary for the other nodes.

These building blocks (automata networks equipped with particular pseudo-orbits) are called *coherent gadgets*. The general idea is that the particular pseudo-orbits of these gadgets can be *glued* together to form pseudo-orbits of larger and larger networks and eventually produce a set of genuine orbits in some network of the family that holds a simulation of a target automata network. The core of the approach is that these gadgets do not simulate pieces of networks or nodes directly, but they simulate finite maps with inputs and outputs (like Boolean gates). They do so by using a common interface made of two things:

- inputs and outputs in a gadget are copies of a fixed subnetwork C;
- pseudo-orbits associated to the gadgets have the same set of traces (*i.e.* sequences of states during the evolution) on each copy of C.

To make an analogy with electronic equipment in the real world, C specifies a physical connector and the common traces on each copy of C specify a communication protocol. Like in the real world, the physical connections are symmetric and it is the dynamics of the gadgets that induces a direction of communication from input to outputs. Of course, the pseudo-orbits also have to realize the finite map being simulated through the interface.

In [20], the approach is developed for an arbitrary abstract set of finite maps in the context of an arbitrary (not necessarily concrete or symmetric) family of automata networks. Then, particular sets of finite maps are identified such that a family having coherent gadgets for them is universal. In the present paper, we will focus on a single set of finite maps that turns out to be sufficient for universality: monotone Boolean maps with two inputs and two outputs. We therefore directly define what having *coherent monotone gadgets* means.

Definition 7. *A CSAN family $\mathcal{F}$ over alphabet Q has* coherent monotone gadgets *if there are three networks in the family,* AND *(the AND gadget) and* OR *(the OR gadget), and C (the common interface) such that the three following groups of conditions hold (i in sub/super-scripts stands for "input" and o for "output"):*

- *glueing interface conditions:*
  - *the nodes of C are partitioned in two sets C_i and C_o ;*
  - *there are four disjoint copies of C in* AND*: $\phi^i_{\mathrm{AND},k} : V_C \to V_{\mathrm{AND}}$ for $k = 1, 2$ (called inputs) and $\phi^o_{\mathrm{AND},k} : V_C \to V_{\mathrm{AND}}$ for $k = 1, 2$ (called outputs) are labeled graph embeddings of C into* AND *with disjoint images ;*
  - *the C_i part of each input is only connected to the rest of the network through the input : $N_{AND}(\phi^i_{\mathrm{AND},k}(C_i)) \subseteq \phi^i_{\mathrm{AND},k}(V_C)$ for $k = 1, 2$; and symmetrically for outputs, the C_o part is only connected through the output : $N_{AND}(\phi^o_{\mathrm{AND},k}(C_o)) \subseteq \phi^o_{\mathrm{AND},k}(V_C)$;*
  - *the same holds for network* OR*, with respective inputs $\phi^i_{\mathrm{OR},k} : V_C \to V_{\mathrm{OR}}$ and outputs $\phi^o_{\mathrm{OR},k} : V_C \to V_{\mathrm{OR}}$ for $k = 1, 2$;*
- *coding conditions:*
  - *there are two* state configurations *$s_k \in Q^{V_C}$ for $k = 0, 1$ to code Boolean values on inputs and outputs ;*

– *there are two* context configurations $c_A \in Q^{\hat{V}_{\mathrm{AND}}}$ *and* $c_O \in Q^{\hat{V}_{\mathrm{OR}}}$ *where* $\hat{V}_{\mathrm{AND}}$ *is the set of nodes of* AND *outside inputs and outputs (and similarly for* $\hat{V}_{\mathrm{OR}}$*) ;*

– *a* time constant $T \geq 1$ *;*

– *four* standard traces $\tau_{a,b} \in (Q^{V_C})^{0,\dots,T}$ *for each pair* $a, b \in \{0,1\}$ *(each representing a transition from Boolean value* a *to Boolean value* b*), and such that they respect state coding at the initial and final steps:* $\tau_{a,b}(0) = s_a$ *and* $\tau_{a,b}(T) = s_b$ *;*

- *correct computation conditions:*

  – *for each pair of initial input Boolean values* $a_{i,1}$ *and* $a_{i,2}$*, each pair of final input Boolean values* $b_{i,1}$ *and* $b_{i,2}$*, and each pair of initial output values* $a_{o,1}$ *and* $a_{o,2}$*, there is a* $(V_{\mathrm{AND}} \setminus \hat{V}_{\mathrm{AND}})$*-pseudo-orbit* $(x^t)_{0 \leq t \leq T}$ *of network* AND *such that:*

    * x^0 *correctly codes the initial input and output values using state and context configurations:* $x^0_{\hat{V}_{\mathrm{AND}}} = c_A$ *and* $x^0_{\phi^i_{\mathrm{AND},k}(V_C)} = s_{a_{i,k}}$ *and* $x^0_{\phi^o_{\mathrm{AND},k}(V_C)} = s_{a_{o,k}}$ *for* $k = 1, 2$ *;*

    * x^T *correctly codes the final input values and codes at the outputs the AND map applied to the initial input values:* $x^T_{\hat{V}_{\mathrm{AND}}} = c_A$ *and* $x^T_{\phi^i_{\mathrm{AND},k}(V_C)} = s_{b_{i,k}}$ *and* $x^T_{\phi^o_{\mathrm{AND},k}(V_C)} = s_{a_{i,1} \wedge a_{i,2}}$ *for* $k = 1, 2$ *;*

  – *the same holds for network* OR *computing the logical or of the initial inputs as final output values.*

As said above, the priced paid in the technicality of the above definition has a nice counterpart: it gives a sufficient condition for strong universality.

Theorem 8. *[20, Corollary 8] If a CSAN family has coherent monotone gadgets then it is strongly universal.*

Showing the existence of monotone gadgets will be the main tool in the case study of examples below. However, the theoretical framework developed in [20] allows to obtain weaker results using other types of gadgets. We will use one type of gadgets suitable to show the simulation of the family of conjunctive networks on directed graphs, conjunctive gadgets, that simulate the two finite maps corresponding to the copy (duplication) and logical conjunction.

Definition 9. *A concrete family* $\mathcal{F}$ *has* coherent conjunctive gadgets *if it satisfies Definition 7 where* AND *and* OR *are replaced by the two following finite maps:* $\mathrm{AND} : \{0,1\}^2 \to \{0,1\}$ *and* $\mathrm{COPY} : \{0,1\} \to \{0,1\}^2$ *with* $\mathrm{AND}(x, y) = x \cdot y$ *and* $\mathrm{COPY}(x) = (x, x)$*.*

Remark 2. *Observe that the only conditions in Definition 7 that depend on the actual boolean functions (and not on the structure of the graphs) are correct computation conditions. Thus, in the case of coherent conjunctive gadgets, the conditions for* $\mathrm{COPY}(x) = (x, x)$ *are the existence of a* $V_{\mathrm{COPY}} \setminus \hat{V}_{\mathrm{COPY}}$ *pseudo-orbit computing the copy of the initial input as final output values.*

Theorem 10. *If a concrete family has coherent conjunctive gadgets, then it simulates the family of conjunctive directed networks in linear time and polynomial space. In particular it admits superpolynomial cycles.*

Proof. The simulation result follows from [20], Corollary 1 and Theorem 21. The simulation results implies the existence of superpolynomial cycles since conjunctive networks admit themselves superpolynomial cycles (see the proof of [20, Theorem 20] for more details). □

3 Update schemes via projections and asynchronous extensions

One of the most studied types of update schemes are the *periodic update schemes*, *i.e.* modes where map μ is periodic. This class contains well-studied particular cases, for instance: parallel update scheme (also called synchronous), in which all the nodes of the networks are updated at the same time (see Figure 3) and also the block sequential update schemes in which each node is updated once every p steps (but not necessarily all at the same time). In addition, we explore a new class of update schemes which contains all the rest that it is called *local clocks*. In this class, each node v is updated once every p_v steps but the frequency of update p_v might depend on the node. This scheme can be seen intuitively as follows which justify the name: each node possesses an internal clock that ticks periodically and triggers an update of the node.

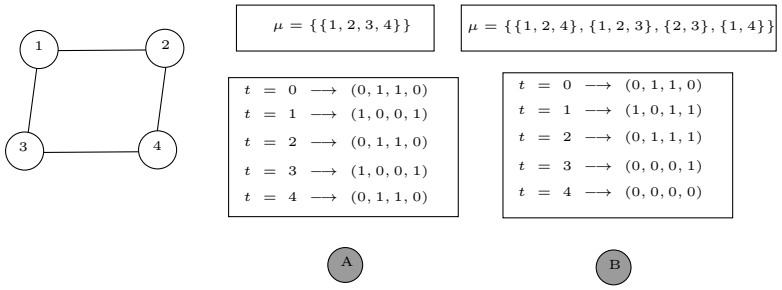


Figure 3: Synchronous update scheme and general periodic update scheme for the same conjunctive automata network. Local function is given by the minimum (AND function) over the set of states of neighbors for each node. A) Synchronous or parallel update scheme. In this case μ has period 1 and all nodes are updates simultaneously. Observe that dynamics exhibits an attractor of period 2 B) General periodic update scheme over a conjunctive network. In this case μ has period 4 and dynamics reach a fixed point after 4 time steps. Observe that there is no restriction on how many times a node is updated. For example, 1 is updated 3 times every 4 time steps but 4 is updated only 2 times every 4 time steps.

Definition 11. *We say that an update scheme μ is a* periodic update scheme *if there exists $p \in \mathbb{N}$ such that $\mu(n+p) = \mu(n)$ for all $n \in \mathbb{N}$. Moreover, we say that μ is*

- *a* block sequential scheme *if there are subsets (called blocks) $B_0, \ldots, B_{p-1} \subseteq V$ forming a partition [1] of V such that $\mu(n) = B_{n \bmod p}$,*
- *a* local clocks scheme *if for each $v \in V$ there is a local period $\tau_v \in \mathbb{N}$ and a time shift $0 \leq \delta_v < \tau_v$ such that $v \in \mu(n) \iff \delta_v = n \bmod \tau_v$.*

For a concrete example on how these update schemes work, see Figure 4 in which different dynamics for a simple conjunctive network under block sequential and local clocks update schemes are shown.

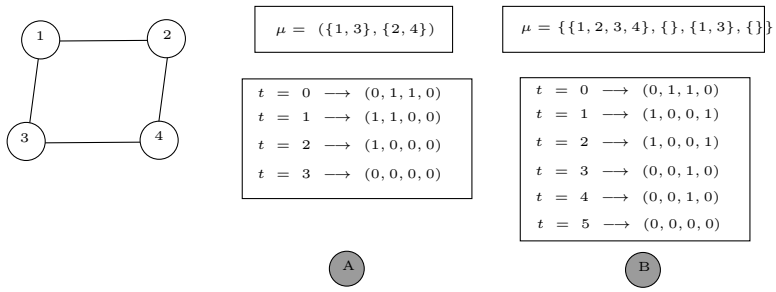
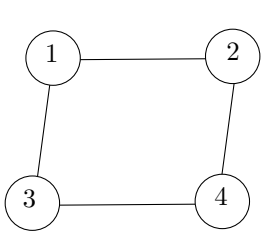


Figure 4: Block sequential and local clocks update schemes over a simple conjunctive network. Local functions are given by the minimum (AND) over the states of the neighbors of each node. A) Block sequential update scheme. In this case function μ is defined by two blocks: $\{1,3\}$ and $\{2,4\}$. Dynamics reaches a fixed point after 3 time steps. B) Local clocks update scheme. In this case each node has an internal clock with different period. Nodes 1 and 3 are updated every two steps ($\tau_1 = \tau_3 = 2$) and nodes 2 and 4 are updated every 4 time steps (i.e. $\tau_2 = \tau_4 = 4$). Shift parameter is 0 for all nodes $\delta_1 = \delta_2 = \delta_3 = \delta_4 = 0$. Dynamics reaches a fixed point after 3 time steps.

Block sequential and local clocks schemes are clearly periodic schemes. Moreover, any block sequential scheme given by $B_0, \ldots, B_{p-1} \subseteq V$ is a local clocks scheme given by $\tau_v = p$ and $\delta_v = i \iff v \in B_i$ for all $v \in V$. As already said, block sequential schemes can thus be seen as local clocks schemes where all nodes share the same update frequency. General periodic update schemes allows different time intervals between two consecutive updates of a node, which local clocks schemes obviously can't do (see Figure 3, and Figure 5). We will see later the tremendous consequences that such subtle differences in time intervals between updates at each node can have. For now let us just make the formal observation that the inclusions between these families of update schedules are strict when focusing on the sets of maps μ.

[1] For technical reasons, we consider that some of these sets may be empty. This assumption does not have any significant effect in the dynamics.



$$\mu = \{\{1,2,4\},\{1,2,3\},\{2,3\},\{1,4\}\}$$

$$\begin{array}{rcl} t = 0 & \longrightarrow & (0,1,1,0) \\ t = 1 & \longrightarrow & (1,0,1,1) \\ t = 2 & \longrightarrow & (0,1,1,1) \\ t = 3 & \longrightarrow & (0,0,0,1) \\ t = 4 & \longrightarrow & (0,0,0,0) \end{array}$$

Figure 5: General periodic update scheme over a conjunctive network. In this case μ has period 4 and dynamics reaches a fixed point after 4 time steps. Observe that there is no restriction on how many times a node is updated. For example, 1 is updated 3 times every 4 time steps but 4 is updated only 2 times every 4 time steps.

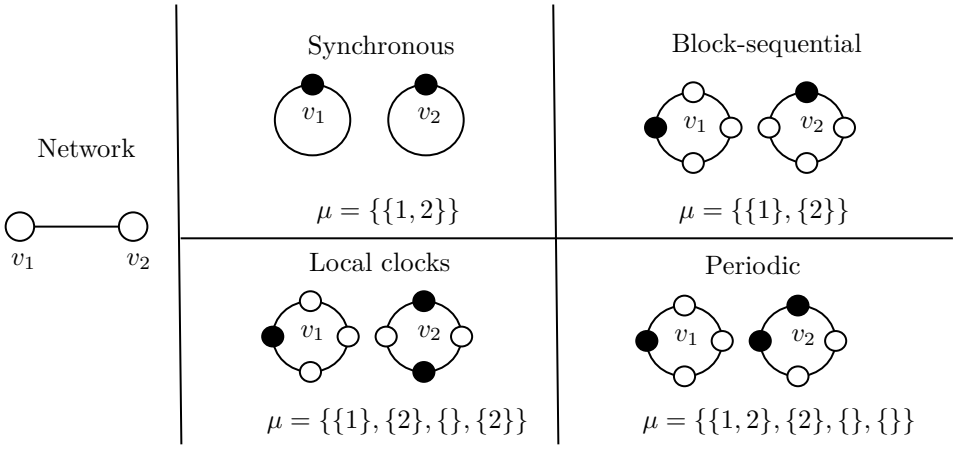


Figure 6: Example of a simple dynamics for four update schemes: synchronous, block-sequential, local clocks and periodic. Big circles surrounding node names represent clocks where time passes by turning clockwise (starting at noon), small black circles on it represent times step (modulo the period) at which the node is updated.

Remark 3. *A so-called* block-parallel *scheme has also been considered more recently [5] which is defined by a set of list of nodes $L_i = (v_{i,j})_{0 \leq j < p_i}$ (for $1 \leq i \leq k$) forming a partition (i.e. such that $v_{i,j}$ are all distinct and $\cup_{i,j} v_{i,j} = V$) to which is associated the map μ such that $v_{i,j} \in \mu(n) \iff j = n \bmod p_i$. It is a particular case of our definition of local clocks scheme above with the additional constraints that the size of the set $\mu(n)$ of updated nodes is constant (related to the size of the network n). We note that, conversely, any local clocks scheme on a given network can be simulated by a block-parallel scheme by artificially adding disconnected nodes that do nothing but satisfy the constraint of $\mu(n)$ being of constant size.*

We present in Figure 6 a summary of the previously introduced update schemes.

Now we present a dynamical formalism that allow us to include all the update schemes presented above and possibly other ones into one formalism. We remark that in all periodic schemes, a given node can take the decision to update or not by simply keeping track of the current value of time modulo the period. The key observation is that, when we add the knowledge of time modulo the period at each node as a new component of states, the whole system becomes deterministic. In fact, we are recovering the original dynamics of some automata network with alphabet Q under a periodic update scheme by projecting a specific deterministic automata network with alphabet $Q' \times Q$ onto Q.

Let $F : Q^V \to Q^V$ be an abstract automata network. We define its *asynchronous* version as the automata network that in every time-step (non-deterministically) choose if a node i should be updated or if it will be stay in the same state. More precisely the *asynchronous* version of F is the non-deterministic function $F^* : Q^V \to (\mathcal{P}(Q))^V$ such that $F^*(x)_i = \{x_i, F(x)_i\}$. Note that, analogously to the deterministic case one can define an orbit starting from x of F^* as a sequence of states $\mathcal{O}_{F^*}(x) = x^0 = x, x^1, x^2, \ldots, x^t, \ldots, \in Q^n$ such that $x_i^s \in F(x^{s-1})_i$ for $i \in V$ and $s \geq 1$. Note also that, given $x \in Q^V$ and an orbit $\mathcal{O}_{F^*}(x)$ we can see $\mathcal{O}_{F^*}(x)$ as a particular realization of certain update scheme μ. More precisely, there exist an update scheme μ (which is defined in the obvious way i.e. by updating the

corresponding nodes in every time step according to points in $\mathcal{O}_{F^*}(x)$) such that for every $x^s \in \mathcal{O}_{F^*}(x)$ we have $x^s = (\mathcal{O}_{F,\mu}(x))^s$. In addition, we have that for each update scheme μ there exist an orbit of F^* which coincides with its dynamics in every time step. Thus, we could work with F^* in order to globally study all possible update schemes. However, we are interested in specific update schemes and we would like to continue working in a deterministic framework in order to keep things simple (in particular the notion of simulation that we define later).

In order to achieve this task, we introduce the following notion of asynchronous extension which is a way to produce the dynamics of different update schemes through projection.

Definition 12. *Let Q be a finite alphabet and $Q' = Q \times R$ where R is finite. Let $F : Q^V \to Q^V$ and $F' : Q'^V \to Q'^V$ two abstract automata networks. We say that F is a projection system of F' if for all $x \in Q'^V$ it holds*

$$\overline{\pi}(F'(x)) \in F^*(\overline{\pi}(x))$$

where $\overline{\pi}$ is the node-wise extension of the projection $\pi : Q' \to Q$ such that $\pi(q, r) = q$ for all $(q, r) \in Q'$.

We show hereunder that the dynamics associated to any of the previously presented periodic update schemes can be described as an asynchronous extension over a product alphabet. To simplify notations we sometimes identify $(A \times B \times \cdots)^V$ with $A^V \times B^V \times \cdots$.

Definition 13 (block sequential extension)**.** *Let $F : Q^V \to Q^V$ be an abstract automata network. Let $b \leq n$ and let $Q' = Q \times \{0, \ldots, b-1\}$. We define the block sequential extension of F with b blocks as the automata network $F' : (Q')^V \to (Q')^V$ such that for all $x = (x_Q, x_b) \in Q'^V$ and all $v \in V$:*

$$F'(x)_v = \begin{cases} (F(x_Q)_v, (x_b)_v - 1 \mod b) & \text{if } (x_b)_v = 0, \\ ((x_Q)_v, (x_b)_v - 1 \mod b) & \text{else.} \end{cases}$$

Definition 14 (local clocks extension)**.** *Let $F : Q^V \to Q^V$ be an abstract automata network. Let $c \in \mathbb{N}$ and let $Q' = Q \times \{0, \ldots, c-1\} \times \{1, \ldots, c\}$. We define the local clocks extension of F with clock length c as the automata network $F' : (Q')^V \to (Q')^V$ such that for all $x = (x_Q, x_c, x_m) \in Q'^V$ and all $v \in V$:*

$$F'(x)_v = \begin{cases} (F(x_Q)_v, (\psi_{(x_m)_v}[(x_c)_v]) + 1 \mod (x_m)_v, (x_m)_v) & \text{if } (x_c)_v = 0, \\ ((x_Q)_v, (\psi_{(x_m)_v}[(x_c)_v]) + 1 \mod (x_m)_v, (x_m)_v) & \text{else.} \end{cases}$$

$\psi_m(r) : \{0, \ldots, c-1\} \to \{0, \ldots, c-1\}$ is such that $\psi_m(r) = \begin{cases} r & \text{if } r \leq m-1, \\ m-1 & \text{else.} \end{cases}$

Definition 15 (periodic extension)**.** *Let $F : Q^V \to Q^V$ be an abstract automata network. Let $p \in \mathbb{N}$ and let $Q' = Q \times \{0, \ldots, p-1\} \times 2^{\{0,\ldots,p-1\}}$. We define the periodic extension of F with period length p as the automata network $F' : (Q')^V \to (Q')^V$ such that for all $x = (x_Q, x_p, x_s) \in Q'^V$ and all $v \in V$:*

$$F'(x)_v = \begin{cases} (F(x_Q)_v, (x_p)_v + 1 \mod p, (x_s)_v) & \text{if } (x_p)_v \in (x_s)_v, \\ ((x_Q)_v, (x_p)_v + 1 \mod p, (x_s)_v) & \text{else.} \end{cases}$$

Remark 4. *Observe that, given an abstract automata network $F : Q^V \to Q^V$ and an asynchronous extension $F' : (Q \times R)^V \mapsto (Q \times R)^V$ of the type previously defined i.e. a block sequential extension, a local clocks extension or a periodic extension, both F' and F^μ (where μ is some of the latter update schemes) describe the same dynamics. In fact, let us illustrate this fact by analyzing the case of the block sequential extension (the other cases are analogous). Let $b \geq 1$ and $\mu_b = (I_0, \ldots, I_{b-1})$. Note that $(I_0, \ldots, I_{b-1})$ is an ordered partition of V. On one hand, we consider an arbitrary initial condition $x \in Q^V$ and the correspondent block sequential orbit $\mathcal{O}_{\mu,F}(x)$. On the other hand, we consider a block extenstion $F^{'} : (Q \times \{0, \ldots, b-1\})^V \mapsto (Q \times \{0, \ldots, b-1\})^V$ and an initial condition $z \in (Q \times \{0, \ldots, b-1\})^V$ given by $z_v = (x_v, y_v)$ where $y_v = k$ if and only if $v \in I_k$ for $1 \leq k \leq b$. By the definition of $F^{'}$, we have that $\overline{\pi}(F'(z)) = (\mathcal{O}_{\mu,F}(x))^1$ since the only nodes v in which F is applied (in the first coordinate) are the ones such that $v \in I_0$. In addition, we have that for each v in V, $\overline{\pi_2}(F^{'}(x))_v = y_v - 1 \mod b$, where π_2 is the node-wise extension of the proyection $\pi_2 : Q \times \{0, \ldots, b-1\} \mapsto \{0, \ldots, b-1\}$. Thus, we have $\overline{\pi}(F'(F'(x))) = (\mathcal{O}_{\mu,F}(x))^2$. Iteratively, we deduce $\overline{\pi}(F'^t(x)) = (\mathcal{O}_{\mu,F}(x))^t$ for each $t \geq 1$.*

Conversely, let us choose an arbitrary initial condition $x = (x_Q, x_b) \in (Q \times \{0, \ldots, b-1\})^V$. We define the ordered partition $I_0 \ldots, I_{b-1}$ given by $v \in I_k$ if and only if $(x_b)_v = k$ for $0 \leq k \leq b-1$. Then, the second coordinate of the initial condition x_b induces a block sequential update scheme μ_{x_b} which is defined by the latter ordered partition.

The previous definitions are formalized for every abstract automata network. We now focus on CSAN families where the extensions are also CSAN as show in the following lemma.

Lemma 16. *Let F be a CSAN, then any block sequential extension (resp. local clocks extension, resp. periodic extension) of F is a CSAN. Moreover, for any CSAN family $\mathcal{F}$ and any fixed b, the set of block sequential extensions with b blocs of networks of $\mathcal{F}$ is again a CSAN family. The same holds for local clocks and periodic extensions.*

Proof. In each case, the definition of the extension F' with alphabet $Q' = Q \times R$ is such that the action of F' on the R component is purely local (the new value of the R component of a node evolves as a function of the old value of this R component) and the value of the R component at a node determines alone if the Q component should be updated according to F or left unchanged. Therefore clearly F' is a CSAN if F is.

In the context of a CSAN family $\mathcal{F}$, the CSAN definition of F' involves only local constraints coming from $F \in \mathcal{F}$ and the action on the R-component is the same at each node. So the second part of the lemma is clear. □

Remark 5. *This approach by asynchronous extensions can also capture non-periodic update schemes. For instance [11] studies an update scheme for Boolean networks called firing memory which uses local delays at each node and, in addition, makes the delay mechanism depend on the state of the current configuration at the node. Firing memory schemes can be captured as an asynchronous extension in such a way that the above lemma for the CSAN case still works.*

To sum up, our formalism allows to treat variations in the update scheme as a change in the CSAN family considered. Given a CSAN family $\mathcal{F}$ and integers b, c, p, we introduce the following notations:

- $\mathcal{F}^{\text{BLOCK},b}$ is the CSAN family of all block sequential extensions of networks from $\mathcal{F}$ with b blocks,
- $\mathcal{F}^{\text{CLOCK},c}$ is the CSAN family of all local clocks sequential extensions of networks from $\mathcal{F}$ with clock length c,
- $\mathcal{F}^{\text{PER},p}$ is the CSAN family of all periodic extensions of networks from $\mathcal{F}$ with period p.

We now introduce the idea of desymmetrization of a CSAN family induced by a particular update scheme. Essentially, we say a that a set of periodic update schemes induces a desymmetrization if the family equipped with this particular update scheme is capable of efficiently simulating its non-symmetric version. Of course, generally speaking, a universal family will be capable of efficiently simulating any other family, in particular its non-symmetric version. However, the inverse is not necessarily true. In fact, we show in Section 4, that this will be the case of all-positive conjunctive networks.

4 Effect of asynchronism: a case study of symmetric networks

In this section, we focus on studying concrete symmetric automata network (CSAN) families. We use previous theoretical framework on complexity of automata networks families in order to classify different CSAN families according to their dynamical behavior under different update schemes. More precisely, we focus on the families presented in the Section 2.2

In addition, we consider the latter presented three update schemes: block sequential, local clocks and general periodic update scheme. We classify previous families according to their dynamical behavior and simulation capabilities by using the framework presented in previous sections Before we enter into the detail, we present in Table 2 a summary of the main results obtained.

Observe that in each row of Table 2, we show how the dynamical behavior of some CSAN families changes as we change the update scheme. In particular, the most simple ones, such as conjunctive and locally positive networks exhibit a relatively simple dynamical behavior (they have bounded period attractors). Contrarily, the last two families have strong universality even for block sequential update schemes. In addition, we would like to remark that there is not only a hierarchy for update schemes (block sequential update schemes are a particular case of local clocks and both are a particular cases of periodic update schemes) but that network families are also somehow related as conjunctive networks are a sub-family of all the other ones. Additionally, one can also observe that there is some sort of "diagonal emergence" of strong universality in Table 2 consisting in the fact that it seems to exists a trade-off

	Synchronous	Block sequential	Local clocks	Periodic
All-negative 	Transient time: $\mathcal{O}(1)$ Maximum period: $\mathcal{O}(1)$ B-PRED $\in \mathcal{O}(1)$	**universal**	**universal**	**universal**
Locally positive 	Transient time: $n^{\mathcal{O}(1)}$ Maximum period: $\mathcal{O}(1)$ B-PRED $\in$ **P**	Transient time: $n^{\mathcal{O}(1)}$ Maximum period: $\mathcal{O}(1)$ B-PRED $\in$ **P**	**universal**	**universal**
All-positive 	Transient time: $n^{\mathcal{O}(1)}$ Maximum period: $\mathcal{O}(1)$ PRED $\in$ **NC**	Transient time: $n^{\mathcal{O}(1)}$ Maximum period: $\mathcal{O}(1)$ PRED $\in$ **NC**	Transient time: $n^{\mathcal{O}(1)}$ Maximum period: $\mathcal{O}(1)$ PRED $\in$ **NC**	**desymm** Maximum period: non-polynomial PRED $\in$ **NC**

Table 2: Summary of the main results on complexity of the dynamics of the network families studied in the current chapter, depending on different update schemes.

between the complexity in the definition of some network families and the complexity of the corresponding update schemes. In other words, simple families seem to need more complex update schemes in order to be universal and as we pass to more complex rules one can observe this property for simpler update schemes.

4.1 All negative networks

In this section we study conjunctive networks with negative edges without any local constraint in the number of positive edges. Formally the symmetric signed conjunctive networks family is a CSAN family in $\{0,1\}$ in which $\lambda_v : Q \times 2^Q \to Q$ is given by $\lambda(q,S) = 0$ if $0 \in S$ and $\lambda(q,S) = 1$ if $0 \notin S$ and for any $e \in E$ we have $\rho_e \in \{\mathrm{Id}, \mathrm{Switch}\}$. where $\mathrm{Switch}(x) = 1 - x$. We denote previous family as $\mathcal{F}_{\mathrm{neg}}$ and for a different update schemes we consider the notation $\mathcal{F}_{\mathrm{neg}}^{\mathrm{BLOCK},b}, \mathcal{F}_{\mathrm{neg}}^{\mathrm{CLOCK},c}$ and $\mathcal{F}_{\mathrm{neg}}^{\mathrm{PER},p}$ for block sequential, local clocks and periodic versions of this family respectively.

We start by remarking that for the parallel update scheme, $\mathcal{F}_{\mathrm{neg}}$ family is not universal as it is a type of threshold family and thus it have bounded period transient and attractors (see [16, 10]). Then, a natural question is whether this remains true for other update schemes. In this sense, we are going to show that, when we consider the next update scheme in our hierarchy, the block sequential update scheme then, $\mathcal{F}_{\mathrm{neg}}$ family is strongly universal.

4.1.1 Parallel case

We have that symmetric conjunctive networks are a particular case of symmetric threshold networks and thus, the maximum period of the attractors is at most 2, i.e. there are only fixed points and attractors of period 2 [16, 10]. Although this result is quite strong for the general case (there is no constraint on the assignation of the signs) there is one particular case that is interesting since it exhibits an extremely simple dynamical behavior. This is the case in which all the signs of the edges are negative, i.e. all the edges are labeled by the function $\mathrm{Switch}(x)$. Observe that, equivalently, by using De Morgan laws, we have that in that case the local rule of each node is a NOR of the states of its neighbors. We are interested in this particular case because, as we are going to see in the next sections, this family of networks has a very limited behavior for parallel update schemes and at the same time, is strongly universal for the block sequential case.

Lemma 17. *Let us assume that for every $(G,\lambda,\rho) \in \mathcal{F}_{neg}$ (i.e, we have $\rho_e = Switch$ for each $e \in E$). Then, the maximum period of the attractors is constant. In addition, its maximum transient is also constant. In particular, this family of automata networks is not universal.*

Proof. Let $v \in V$ be a node and x be an arbitrary initial condition. It suffices to observe two time steps to completely determine the dynamics of v. In fact, if v is in state 1 then, its neighbors will change to 0 the next time step. In addition, if v remains in state 1 then all their neighbors are in state 0 and then, it cannot change. Thus, we can only have fixed points and two-cycles. In addition, the dynamics converges in constant time (2 time steps). □

4.1.2 Block sequential case

In this section, we will show that $\mathcal{F}_{\text{neg}}^{\text{BLOCK},b}$ is strongly universal as a consequence of its capability to implement coherent monotone gadgets. In addition, we conclude that, as a direct consequence of the latter property, the previous family is both dynamically complex and computationally complex. This means that for this family, complex behavior is exhibited under block sequential update schemes.

As we will be using the same structures to show the main result, we start by showing a less powerful result. We show that $\mathcal{F}_{\text{neg}}^{\text{BLOCK},b}$ is able to simulate a wire module. A wire module is a network that transport information emulating a wire. This is only a way to motivate the main result by showing how key structures work in a particular simple context. As we will see when we present the main result, we use these structures in the actual proof of main theorem. In fact, both modules are part of the monotone gadgets we construct. We remark that all involved networks, both the wire modules and the gadgets have only negative labels, i.e. each edge is labeled by the Switch function which takes a bit x and produces $1 - x$.

Wire modules We define a wire module as an automata network constructed with two copies of the NOT module presented in Figures 7 and 8. Observe that this gadget has 3 central nodes (marked inside a thick dotted rectangle in Figures 7 and 8) together with 2 copies of a 4 nodes cycle graph. Generally speaking, the dynamics on these cycle graphs works as a clock which allows information to flow through the central part in only one direction (from left to right). In addition, they allow the gadget to erase information once it has been transmitted. This latter property allows the gadget to clean itself in order to receive new information. The wire module is composed by two copies of the NOT module as is presented in Figure 9. Since this gadget is composed by two copies of the NOT module, this gadget is defined by a path graph with 6 central nodes together with $2 \times 2 \times 3 = 12$ cycle graphs (two copies for each node). We enumerate nodes in the central part from left to write by the following ordering: $\{0, 1, 2, 3, 4, 5\}$. Additionally, since the functioning of each copy of the NOT module is based in an ordered partition of 3 blocks, we take the union of corresponding blocks in each partition in order to define 3 larger blocks for the wire module. For one of this larger blocks we use the notation $\{0, 1, 2\}$. Thus, observe that each wire takes $T = 6 \times 3 = 18$ time steps in order transport the information. This is because for each round of 3 time steps in which we update each block, we make the signal pass through exactly one node.

We are now in conditions to introduce the main result:

Lemma 18. *The family $\mathcal{F}_{neg}^{BLOCK,3}$ of all signed symmetric conjunctive networks under block sequential update schemes of at most 3 blocks has coherent monotone gadgets.*

Proof. We will show that the gadgets in the Figures 13 and 14 are coherent monotone gadgets. Note that both these gadgets are made of several wires of NOT modules (we call the two in the left hand side of the figure input wires and the other two at the right hand side output wires) and a computation gadget. Observe that, each of these structures (wires and computation gadget) needs exactly 3 blocks. In addition, in Table 5 the dynamics of the 4-cycles that are attached to each node is shown. We recall that the function of these clocks is to allow information to flow in one direction only (all the interactions are symmetric so this is not straightforward) and to erase information once it has been copied or processed by the nodes in the gadget. We use the following notation in order to represent nodes in these structures: $c_{s,i,j,p}$ where s is the number of the NOT module (a 3-node-path together with two 4-cycles for each node, see Figure 10) to which the cycle is attached, so $s \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ where the order is taken from left to right (for example in Figure 13 the copy associated to (v_1, v_2, v_3) comes first then, (v_1', v_2', v_3'), then (v_4, v_5, v_6) and so on). Additionally, i is in the position of the cycle in the NOT block, so $i \in \{1, 2, 3\}$, j is the position of the node relative to the 4-cycle graph considered in counter clockwise order (see Figure 10), so $j \in \{1, 2, 3, 4\}$ and p is the position of the cycle in the central structure of the gadget. Observe that, since there are two copies for each node, we denote them by *upper* and *lower* so $p \in \{u, l\}$. Since input wires and output wires are needed to be updated at the same time and are independent (we have two copies for each one) we combine their blocks in the obvious way (we take the union of pairs of blocks that have the same update order). More precisely, we combine 3×9 blocks of each particular part (there are 8 copies of NOT and one computation gadget having 3 blocks each one) in order to define again only 3 blocks. Precise definition, using notation shown in Figures 13 and 14, is the following:

- $B_0 = \bigcup_{i=1}^{12} \{v_i\} \cup \{v_i'\} \cup \bigcup_{s,i} \{c_{s,i,1,l}, c_{s,i,2,l}\}$;
- $B_1 = \bigcup_{s} \{c_{s,1,3,u}, c_{s,1,4,u}, c_{s,2,1,u}, c_{s,2,2,u}, c_{s,3,1,u}, c_{s,3,2,u}\}$; and

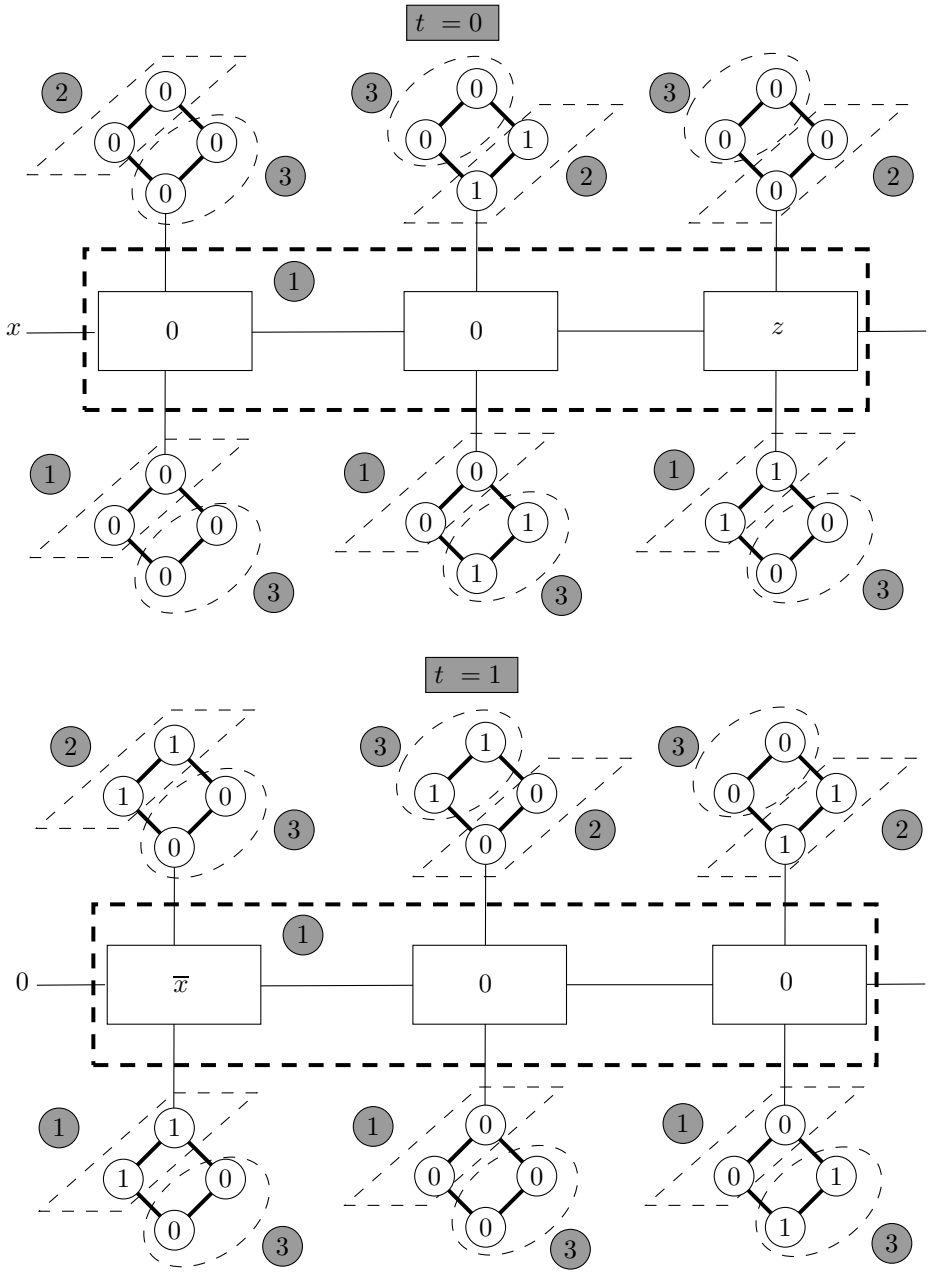


Figure 7: One step of the dynamics of the NOT module implemented by a signed symmetric conjunctive network. Dotted figures represent blocks. Numbers in gray represent the updating order of each block. Total simulation time is $T = 9$.

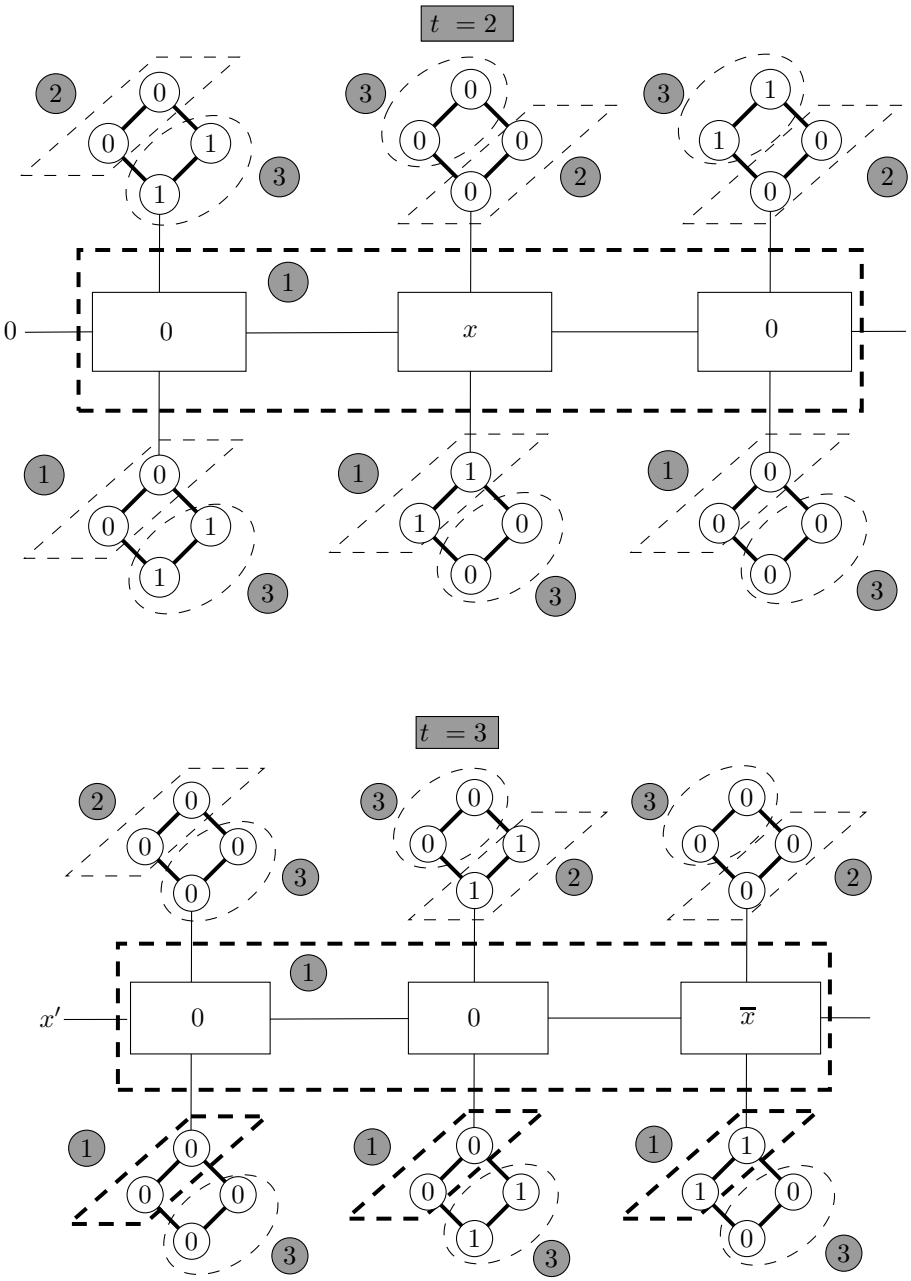


Figure 8: Two last steps of the dynamics described by the NOT module implemented by a symmetric signed conjunctive network. Dotted figures represent blocks. Numbers in gray represent the updating order of each block. Total simulation time is $T = 9$

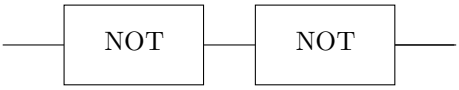


Figure 9: Wire module implemented on a signed symmetric conjuntive network. 2 copies of NOT module are combined in order to form a wire. Simulation time is $T = 18$.

- $B_2 = \bigcup_{s,i}\{c_{s,i,3,l}, c_{s,i,4,l}\}$.

Now we are going to use information in Tables 3, 4 and 5 in order to show that these gadgets satisfy the conditions of Definition 7. Observe that, on the one hand, for the OR gadget (see Figure 13), input wires compute the result in 3×3 (it needs to carry the signal through the three nodes in the wire and each of this intermediate steps takes three steps, one for each block) time steps and computation gadget takes 3×3 as well. On the other hand, for the AND gadget (see Figure 14) input wires compute desired in 3×6 time steps. We define the associated network as $F_{\text{AND}} : (\{0,1\} \times \{0,1,2\})^{15\times 2\times 4} \to (\{0,1\} \times \{0,1,2\})^{15\times 2\times 4}$ and $F_{\text{OR}} : (\{0,1\} \times \{0,1,2\})^{15\times 2\times 4} \to (\{0,1\} \times \{0,1,2\})^{15\times 2\times 4}$. In fact we have that:

1. There is a unique glueing interface given by $C = C_i \cup C_o$ where:
   - $C_i = \{i\} \cup \{a(i,1), a(i,2), a(i,3), a(i,4)\} \cup \{a'(i,1), a'(i,2), a'(i,3), a'(i,4)\}$; and
   - $C_o = \{o', o\} \cup \{a(o',1), a(o',2), a(o',3), a(o',4)\} \cup \{a(o,1), a(o,2), a(o,3), a(o,4)\} \cup \{a'(o',1), a'(o',2), a'(o',3), a'(o',4)\} \cup \{a'(o,1), a'(o,2), a'(o,3), a'(o,4)\}$

   We define labelling functions $\phi^i_{\text{AND},k}$, $\phi^o_{\text{AND},k}$, $\phi^i_{\text{OR},k}$, $\phi^o_{\text{OR},k}$ (for the sake of simplicity we show the definition for the AND gadget since the one for the OR gadget is completely analogous) for $k = 1, 2$ as:
   - $\phi^i_{\text{AND},1}(i) = v_1$, $\phi^i_{\text{AND},1}(o') = v_2$ and $\phi^i_{\text{AND},1}(o) = v_3$;
   - $\phi^i_{\text{AND},1}(a(i,r)) = c_{1,1,r,u}$, $\phi^i_{\text{AND},1}(a'(i,r)) = c_{1,1,r,l}$ for $r = 1,2,3,4$, where 1 corresponds to the NOT module which starts with v_1 in Figure 14.
   - $\phi^i_{\text{AND},1}(a(o',r)) = c_{1,2,r,u}$, $\phi^i_{\text{AND},1}(a'(o',r)) = c_{1,2,r,l}$ for $r = 1,2,3,4$, where 1 corresponds to the NOT module which starts with v_1 in Figure 14;
   - $\phi^i_{\text{AND},1}(a(o,r)) = c_{1,3,r,u}$, $\phi^i_{\text{AND},1}(a'(o,r)) = c_{1,3,r,l}$ for $r = 1,2,3,4$, where 1 corresponds to the NOT module which starts with v'_1 in Figure 14;
   - $\phi^i_{\text{AND},2}(i) = v'_1$, $\phi^i_{\text{AND},2}(o') = v'_2$ and $\phi^i_{\text{AND},2}(o) = v'_3$;
   - $\phi^i_{\text{AND},2}(a(i,r)) = c_{1',1,r,u}$, $\phi^i_{\text{AND},1}(a'(i,r)) = c_{1',1,r,l}$ for $r = 1,2,3,4$, where 2 correspond to the NOT module which starts with v'_1 in Figure 14;
   - $\phi^i_{\text{AND},2}(a(o',r)) = c_{1',2,r,u}$, $\phi^i_{\text{AND},1}(a'(o',r)) = c_{1',2,r,l}$ for $r = 1,2,3,4$, where 2 corresponds to the NOT module which starts with v'_1 in Figure 14;
   - $\phi^i_{\text{AND},2}(a(o,r)) = c_{1',3,r,u}$, $\phi^i_{\text{AND},1}(a'(o,r)) = c_{1',3,r,l}$ for $r = 1,2,3,4$, where 2 corresponds to the NOT module which starts with v'_1 in Figure 14.
   - $\phi^o_{\text{AND},1}(i) = v_{10}$, $\phi^o_{\text{AND},1}(o') = v_{11}$ and $\phi^o_{\text{AND},1}(o) = v_{12}$;
   - $\phi^o_{\text{AND},1}(a(i,r)) = c_{5,1,r,u}$, $\phi^o_{\text{AND},1}(a'(i,r)) = c_{5,1,r,l}$ for $r = 1,2,3,4$, where 8 corresponds to the NOT module which starts with v_{10} in Figure 14.
   - $\phi^o_{\text{AND},1}(a(o',r)) = c_{5,2,r,u}$, $\phi^o_{\text{AND},1}(a'(o',r)) = c_{5,2,r,l}$ for $r = 1,2,3,4$, where 8 corresponds to the NOT module which starts with v_{10} in Figure 14;
   - $\phi^o_{\text{AND},1}(a(o,r)) = c_{5,3,r,u}$, $\phi^o_{\text{AND},1}(a'(o,r)) = c_{5,3,r,l}$ for $r = 1,2,3,4$, where 8 corresponds to the NOT module which starts with v_{10} in Figure 14;
   - $\phi^o_{\text{AND},2}(i) = v'_{10}$, $\phi^o_{\text{AND},2}(o') = v'_{11}$ and $\phi^o_{\text{AND},2}(o) = v'_{12}$;
   - $\phi^i_{\text{AND},2}(a(i,r)) = c_{5',1,r,u}$, $\phi^i_{\text{AND},2}(a'(i,r)) = c_{5',1,r,l}$ for $r = 1,2,3,4$, where 9 correspond to the NOT module which starts with v'_{10} in Figure 14;
   - $\phi^o_{\text{AND},2}(a(o',r)) = c_{5',2,r,u}$, $\phi^o_{\text{AND},2}(a'(o',r)) = c_{5',2,r,l}$ for $r = 1,2,3,4$, where 9 corresponds to the NOT module which starts with v_{10} in Figure 14;
   - $\phi^o_{\text{AND},2}(a(o,r)) = c_{5',3,r,u}$, $\phi^o_{\text{AND},2}(a'(o,r)) = c_{5',3,r,l}$ for $r = 1,2,3,4$, where 9 corresponds to the NOT module which starts with v'_{10} in Figure 14.

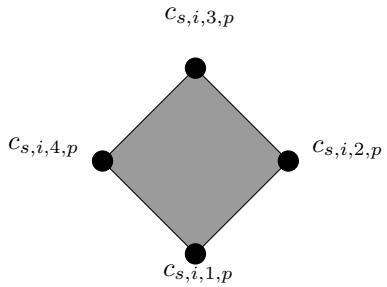


Figure 10: Scheme of labelling for 4-cycles in AND/OR gadgets. Notation is given by the following guidelines: s represent the associated group of three nodes, second two coordinates indicate its position relative to the original gadget (there are 3 clocks) and its position in the 4-cycle graph (considering counter clock-wise order), and u, l stands for upper or lower according to its position in the gadget.

Node/Time	v_1	v_2	v_3	v_4	v_5	v_6	v'_1	v'_2	v'_3	v'_4	v'_5	v'_6	w_1	w_2	w_3	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v'_7	v'_8	v'_9	v'_{10}	v'_{11}	v'_{12}
0	x	0	0	0	0	0	y	0	0	0	0	0	0	0	0	0	0	0	z	0	0	0	0	0	z	0	0
3	0	$\overline{x}$	0	0	1	0	0	$\overline{y}$	0	0	1	0	0	1	0	0	1	0	0	$\overline{z}$	0	0	1	0	0	$\overline{z}$	0
6	0	0	x	0	0	0	0	0	y	0	0	0	0	0	0	0	0	0	0	0	z	0	0	0	0	0	z
9	1	0	0	$\overline{x}$	0	0	1	0	0	$\overline{y}$	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
12	0	0	0	0	x	0	0	0	0	0	y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	1	0	0	$\overline{x}$	0	0	1	0	0	$\overline{y}$	0	0	1	0	0	1	0	0	1	0	1	0	0	0	1
18	0	0	0	0	0	0	0	0	0	0	0	0	$x \wedge y$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	1	0	0	1	0	0	1	0	0	1	0	0	$\overline{x \wedge y}$	0	0	1	0	0	1	0	0	1	0	0	1	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	$x \wedge y$	0	0	0	0	0	0	0	0	0	0	0	0	0
27	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	$\overline{x \wedge y}$	0	0	1	0	0	$\overline{x \wedge y}$	0	0	1	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$x \wedge y$	0	0	0	0	0	$x \wedge y$	0	0	0	0
33	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	$\overline{x \wedge y}$	0	0	1	0	0	$\overline{x \wedge y}$	0	0	1
36	x'	0	0	0	0	0	y'	0	0	0	0	0	0	0	0	0	0	0	$x \wedge y$	0	0	0	0	0	$x \wedge y$	0	0

Table 3: Dynamics for central gadgets in AND gadget implemented over a symmetric signed conjunctive network. Notation is the same of the one shown in Figure 14

2. State configurations are defined for each $q \in \{0,1\}$ as $s_q(i) = (q,1)$ and $s_q(o') = s_q(o) = (0,1)$ and the state configuration of the nodes in the clocks i.e. the ones labeled by a are constant and shown in Table 5. The block number for each of these nodes can is the same as the original NOT module 7.

3. Context configurations are described in Tables 3, 4 and 5 as the ones related to the cycles of length 4 connected to central path of the gadgets and nodes in the path which are not part of the glueing interface.

4. Standard trace is defined in Tables 3 and 4 (which contain the information related to the dynamics of nodes $v_1, v'_1, v_2, v'_2, v_3, v'_3, v_{10}, v'_{10}, v_{11}, v'_{11}, v_{12}, v'_{12}$) and in Table 5 (which contains the dynamics of the nodes in the 4-cycles).

5. Simulation constant is $T = 3 \times 12$ as it is shown in Tables 3,4 and

6. Pseudo-orbit is given by the dynamics shown in in Tables 3, 4, and 5 where x, y, x', y', z are variables.

□

By Theorem 8 we have the following corollary:

Corollary 1. *The family $\mathcal{F}_{neg}^{BLOCK,2}$ of all signed symmetric conjunctive networks under block sequential update schemes with at most 3 blocks is strongly universal.*

Node/Time	v_1	v_2	v_3	v'_1	v'_2	v'_3	w_1	w_2	w_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v'_4	v'_5	v'_6	v'_7	v'_8	v'_9	v'_{10}	v'_{11}	v'_{12}
0	x	0	0	y	0	0	0	0	0	0	0	0	0	0	0	z	0	0	0	0	0	0	0	0	z	0	0
3	0	$\overline{x}$	0	0	$\overline{y}$	0	0	1	0	0	1	0	0	1	0	0	$\overline{z}$	0	0	1	0	0	1	0	0	$\overline{z}$	0
6	0	0	x	0	0	y	0	0	0	0	0	0	0	0	0	0	0	z	0	0	0	0	0	0	0	0	z
9	1	0	0	1	0	0	$\overline{x \vee y}$	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
12	0	0	0	0	0	0	0	$x \vee y$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	1	0	0	1	0	0	$\overline{x \vee y}$	0	0	1	0	0	1	0	0	1	0	1	0	1	0	0	1	0	0
18	0	0	0	0	0	0	0	0	0	$x \vee y$	0	0	0	0	0	0	0	0	$x \vee y$	0	0	0	0	0	0	0	0
21	0	1	0	0	1	0	0	1	0	0	$\overline{x \vee y}$	0	0	1	0	0	1	0	0	$\overline{x \vee y}$	0	0	1	0	0	1	0
24	0	0	0	0	0	0	0	0	0	0	$x \vee y$	0	0	0	0	0	0	0	0	$x \vee y$	0	0	0	0	0	0	0
27	1	0	0	1	0	0	1	0	0	1	0	0	$\overline{x \vee y}$	0	0	1	0	0	1	0	0	$\overline{x \vee y}$	0	0	1	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	$x \vee y$	0	0	0	0	0	0	0	0	$x \vee y$	0	0	0	0
33	0	0	1	0	0	1	0	0	1	0	0	1	0	0	$\overline{x \vee y}$	0	0	1	0	0	1	0	0	$\overline{x \vee y}$	0	0	1
36	x'	0	0	y'	0	0	0	0	0	0	0	0	0	0	0	$x \vee y$	0	0	0	0	0	0	0	0	$x \vee y$	0	0

Table 4: Dynamics for central gadgets in OR gadget implemented over a symmetric signed conjunctive network. Notation is the same of the one shown in Figure 13

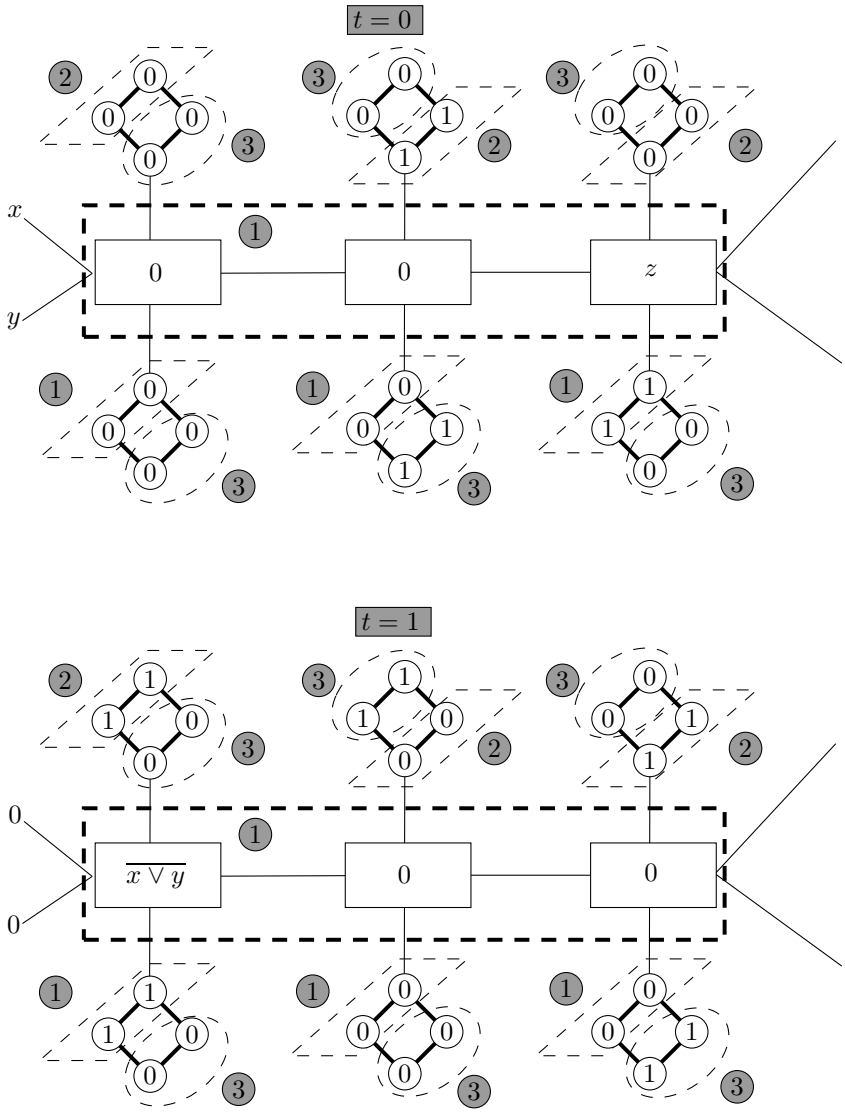


Figure 11: One step of the dynamics of the computation gadget inside NOR gadget implemented by a signed symmetric conjunctive network. Dotted figures represent blocks. Numbers in gray represent the updating order of each block. Total simulation time is $T = 9$.

Node/Time	$c_{s,1,1,u}$	$c_{s,1,2,u}$	$c_{s,1,3,u}$	$c_{s,1,4,u}$	$c_{s,2,1,u}$	$c_{s,2,2,u}$	$c_{s,2,3,u}$	$c_{s,2,4,u}$	$c_{s,3,1,u}$	$c_{s,3,2,u}$	$c_{s,3,3,u}$	$c_{s,3,4,u}$
0	0	0	1	1	0	0	1	1	1	1	0	0
3	1	1	0	0	0	0	0	0	0	0	1	1
6	0	0	0	0	1	1	0	0	0	0	0	0
9	0	0	1	1	0	0	1	1	1	1	0	0
Node/Time	$c_{s,1,1,l}$	$c_{s,1,2,l}$	$c_{s,1,3,l}$	$c_{s,1,4,l}$	$c_{s,2,1,l}$	$c_{s,2,2,l}$	$c_{s,2,3,l}$	$c_{s,2,4,l}$	$c_{s,3,1,l}$	$c_{s,3,2,l}$	$c_{s,3,3,b}$	$c_{s,3,4,b}$
0	1	1	0	0	0	0	0	0	0	0	1	1
3	0	0	1	1	1	1	0	0	0	0	0	0
6	0	0	0	0	0	0	1	1	1	1	0	0
9	1	1	0	0	0	0	0	0	0	0	1	1

Table 5: Dynamics for context in AND/OR gadgets implemented on symmetric signed conjunctive networks. Notation is given by the following guidelines: s represent the associated group of three nodes, second two coordinates indicate its position relative to the original gadget (there are 3 clocks) and its position in the 4-cycle graph (considering counter clock-wise order), and u, l stands for upper or lower according to its position in the gadget.

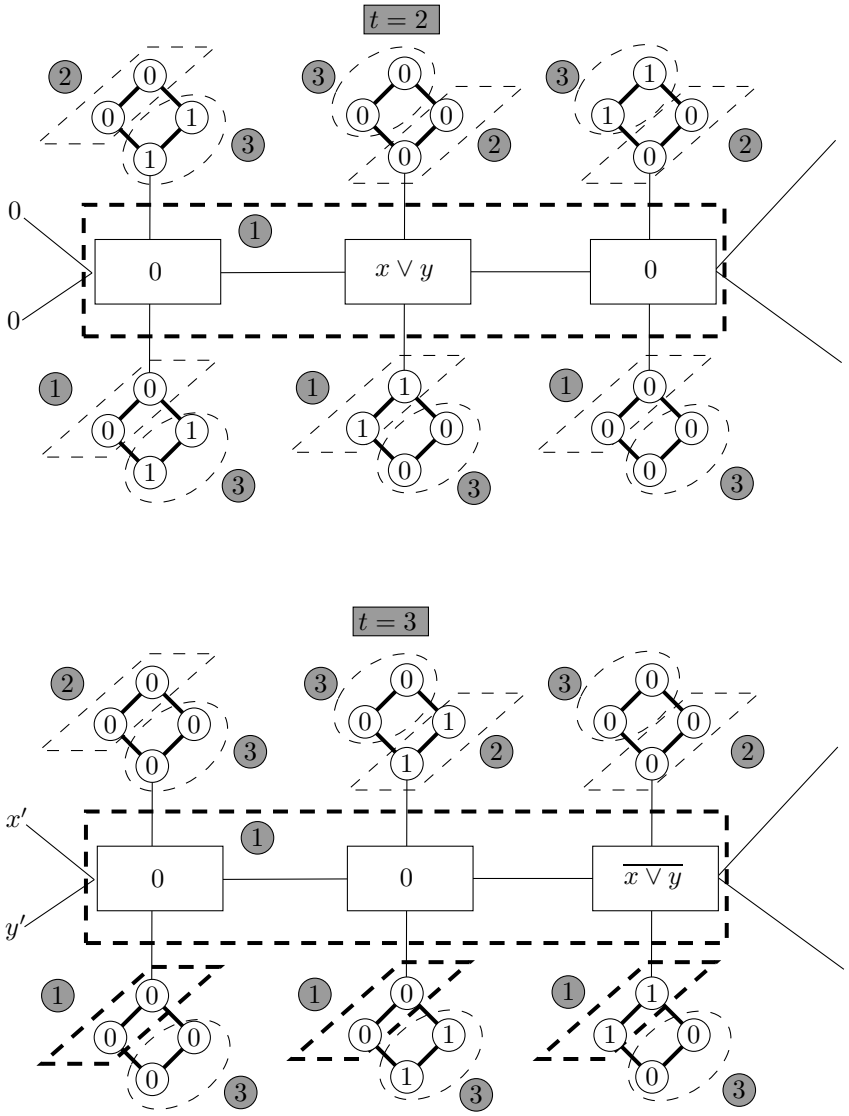


Figure 12: Last two steps of the dynamics of the computation gadget inside NOR gadget implemented by an AND-not network. Dotted figures represent blocks. Numbers in gray represent the updating order of each block. Total simulation time is $T = 9$.

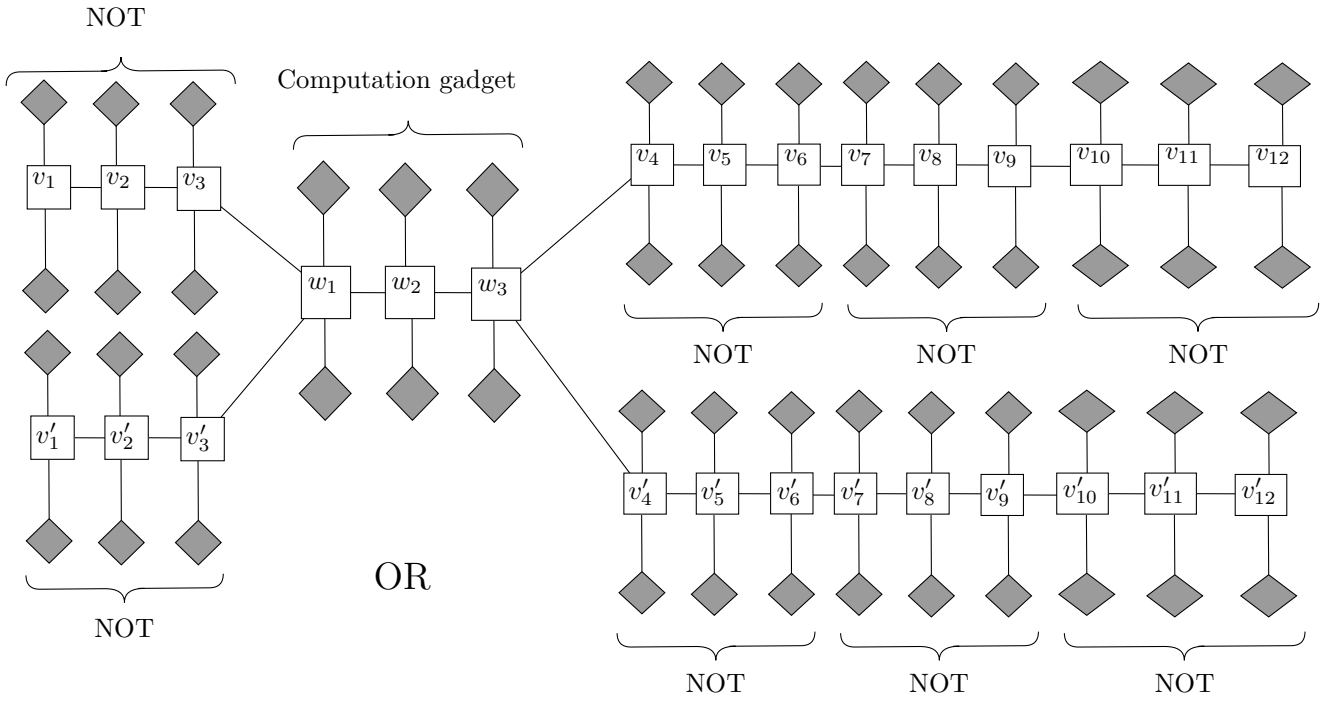


Figure 13: OR gadget structure. In order to produce a OR gadget, wire module and NOT module are combined with computation part showed in Figures 11 and 12.

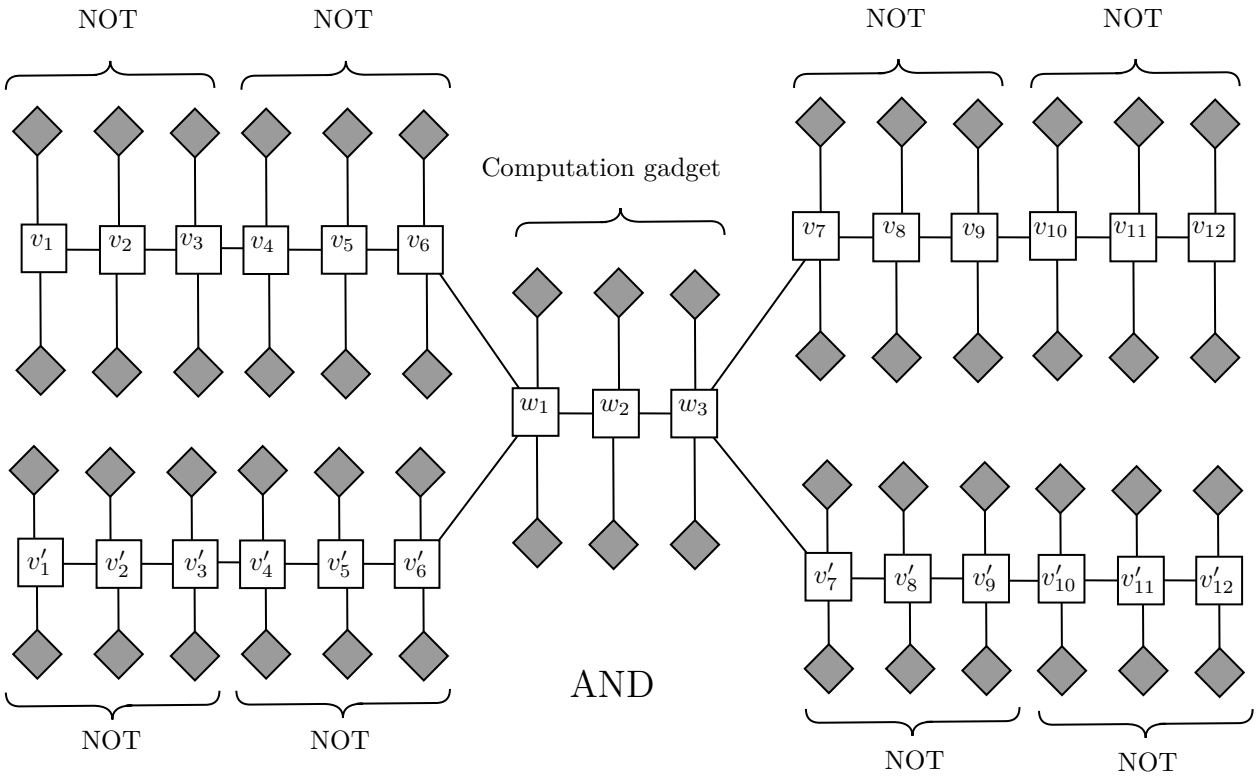


Figure 14: AND gadget structure. In order to implement an AND gadget, wire module and NOT module are combined with computation part showed in Figures 11 and 12.

4.2 Locally positive symmetric conjunctive networks

In this section, we study a generalization of conjunctive networks that we call *locally positive* symmetric conjunctive networks. Using latter notation for CSAN families we denote this family by $\mathcal{F}_{\text{locally-pos}}$. In this particular case, we allow edges to have negative signs (which will switch the state of the corresponding neighbor) but with a local constraint: no neighborhood in which all the connections are negative is allowed. More precisely, a locally positive symmetric conjunctive network is a CSAN (G, λ, ρ) for any $v \in V(G)$ we have $\lambda_v(q, S) = \bigwedge_{q \in S} q$ and there exists $w \in N(v) : \rho(vw) = \mathrm{Id}$.

We will show for this family that the *threshold of universality* when changing update modes is between block sequential update schemes and local clocks update schemes. More precisely, we show, on one hand, that the family remains dynamically constrained under block sequential schedule, and, on the other hand, we show that a local clocks version of this family is strongly universal because it admits coherent monotone gadgets. A similar construction will be used also for the general case in which we allow any kind of sign label over the edges of the network as we will show in next sections.

Theorem 19. *Fix any $b \geq 1$ and consider $\mathcal{F}^{BLOCK,b}_{locally\text{-}pos}$ the family of all locally positive symmetric conjunctive networks under block sequential schedule with at most b blocks. Any orbit of any $F \in \mathcal{F}^{BLOCK,b}_{locally\text{-}pos}$ reaches a cycle of length 1 or $2b$ in linear time (linear in the number of nodes). In particular, B-PRED and REACH problems for $\mathcal{F}^{BLOCK,b}_{locally\text{-}pos}$ can be solved in polynomial time and the family is not universal.*

Proof. Fix such an F and denote by E^+ the set of its edges with positive sign. First, we observe that for a conjunctive network an edge (u, v) such that u and v are in state 0 is a stable configuration meaning that they will remain (both u and v) in state 0 no matter what state is assigned for the rest or the nodes nor the update scheme that is considered. Recall that F is a block sequential extension using alphabet $Q = \{0, 1\} \times \{0, \dots, b-1\}$. To simplify notation, for any configuration x, we denote by $\alpha(x)$ its projection on the state component $\{0, 1\}$ and by $\beta(x)$ its projection on the block component $\{0, \dots, b-1\}$. We say a configuration x is *clean* if for any edge $(v_1, v_2) \in E^+$ it holds: if node v_1 updates strictly before node v_2 starting from x (*i.e.* $\beta(x)_{v_1} < \beta(x)_{v_2}$) then $\alpha(x)_{v_1} \geq \alpha(x)_{v_2}$. First, one can remark that for any configuration x, the configuration $F^b(x)$ is clean. Indeed, the condition is trivial for edges of E^+ where both nodes update synchronously. Moreover, each time a node v_1 from an edge $(v_1, v_2) \in E^+$ updates strictly before v_2 then its state becomes no larger than $\alpha(x)_{v_2}$. Therefore, after a first update of v_1, v_2 becomes the next to be updated and has a no smaller state and the edge (v_1, v_2) maintains the 'cleanness' property for the remaining of the orbit.

Let us now consider an "energy" map giving to each edge $(v_1, v_2) \in E^+$ the energy $\mathcal{E}(x, v_1, v_2) = \alpha(x)_{v_1} + \alpha(x)_{v_2}$.

Claim 1: if x is clean, then for any edge $(v_1, v_2) \in E^+$ it holds $\mathcal{E}(F(x), v_1, v_2) \leq \mathcal{E}(x, v_1, v_2)$ If v_1 and v_2 update synchronously in one step, then $\alpha(F(x))_{v_i} \leq \alpha(x)_{v_j}$ for $i \neq j \in \{1, 2\}$ so $\alpha(F(x))_{v_1} + \alpha(F(x))_{v_2} \leq \alpha(x)_{v_1} + \alpha(x)_{v_2}$. If $\beta(v_1) < \beta(v_2)$ then,

$$\alpha(F(x))_{v_1} \leq \alpha(x)_{v_2} \leq \alpha(x)_{v_1},$$

the last inequality being because x is supposed clean. Thus $\mathcal{E}(F(x), v_1, v_2) \leq \mathcal{E}(x, v_1, v_2)$ because $\alpha(F(x))_{v_2} = \alpha(x)_{v_2}$ in this case. The case where v_2 updates and v_1 doesn't is symmetric. If none of v_1 and v_2 update, their states don't change so the energy remains constant.

Claim 2: if x is clean and $\mathcal{E}(F^{2b}(x), v_1, v_2) = \mathcal{E}(x, v_1, v_2)$ for each $(v_1, v_2) \in E^+$ then $F^{2b}(x) = x$ By Claim 1, considering any edge $(v_1, v_2) \in E^+$, the sum of their states is constant during $2b$ steps. If $\alpha(x)_{v_1} = \alpha(x)_{v_2}$ then necessarily $F^{2b}(x)_{v_i} = x_{v_i}$ for $i = 1, 2$ because energy 0 (resp. 2) can only be realized by the pair of states $(0, 0)$ (resp. $(1, 1)$). If $\alpha(x)_{v_1} \neq \alpha(x)_{v_2}$ then necessarily v_1 and v_2 must swap their state synchronously to maintain the sum constant. This means that after $2b$ steps, they have done exactly two synchronous updates and they are back to the same state, thus we again have $F^{2b}(x)_{v_i} = x_{v_i}$. Since E^+ covers all vertices (property of the locally positive family $\mathcal{F}_{\text{locally-pos}}$), the property $F^{2b}(x)_v = x_v$ holds for all vertices. We conclude that $F^{2b}(x) = x$.

Claim 3: if x is clean and $(v, v_1) \in E^+$ and $(v, v_2) \in E^+$ then $\mathcal{E}(F^{2b}(x), v, v_2) \leq \mathcal{E}(x, v, v_1)$

- if $\mathcal{E}(x, v, v_1) = 2$ the conclusion of the claim is obvious by definition;
- if $\mathcal{E}(x, v, v_1) = 1$ then
  - either $\alpha(x)_v = 0$ and therefore $\alpha(F^b(x))_{v_2} = 0$ since x is clean, so that $\mathcal{E}(F^b(x), v, v_2) \leq 1$ and therefore $\mathcal{E}(F^{2b}(x), v, v_2) \leq 1$ by Claim 1;
  - or $\alpha(x)_{v_1} = 0$ and therefore $\alpha(F^{2b}(x))_{v_2} = 0$ because this time $\alpha(F^b(x))_v = 0$ (because x is clean), so that $\mathcal{E}(F^{2b}(x), v, v_2) \leq 1$;
- if $\mathcal{E}(x, v, v_1) = 0$ then $\alpha(x)_v = \alpha(x)_{v_1} = 0$ hence they will remain in state 0 forever, which implies that $\alpha(F^{2b}(x))_{v_2} = \alpha(F^{2b}(x))_v = 0$ so that $\mathcal{E}(F^{2b}(x), v, v_2) = 0$.

To conclude from the above claims, let $N^+(v)$ be the positive neighborhood of v, *i.e.* the set $N^+(v) = \{v' : (v, v') \in E^+\}$, and consider the following quantity on each vertex for a clean configuration x:

$$\Phi(x, v) = \min_{v' \in N^+(v)} \{\mathcal{E}(x, v, v')\}.$$

Note that from Claim 1, $\Phi(F(x), v) \leq \Phi(x, v)$ for any clean x. From Claim 3, for a clean x and $(v_1, v_2) \in E^+$, it holds that $\mathcal{E}(F^{2b}(x), v_1, v_2) \leq \Phi(x, v_1)$. Hence, if $\Phi(F^{4b}(x), v) = \Phi(x, v)$ for all vertices v, then $\mathcal{E}(F^{4b}(x), v_1, v_2) = \mathcal{E}(F^{2b}(x), v_1, v_2)$ for all $(v_1, v_2) \in E^+$ because

$$\Phi(F^{4b}(x), v_1) \leq \mathcal{E}(F^{4b}(x), v_1, v_2) \leq \mathcal{E}(F^{2b}(x), v_1, v_2) \leq \Phi(x, v_1).$$

In particular, from Claim 2, it implies $F^{4b}(x) = F^{2b}(x)$. Consider now the map $\mathcal{E} : Q^V \to \mathbb{N}$ giving to any configuration the "energy"

$$\mathcal{E}(x) = \sum_{v \in V} \Phi(x, v)$$

which can only decrease along an orbit starting from a clean configuration, and whose maximum value is at most 2 times the number of nodes. We deduce from the above that if x is clean and such that $\mathcal{E}(F^{4b}(x) = \mathcal{E}(x)$, then $F^{2b}(x)$ belongs to a periodic orbit of period at most $2b$ (and in fact either 1 or $2b$ since each node updates once every b steps). On the other hand, the sequence $e_t = \mathcal{E}(F^{bt+1}(x))$ is decreasing for any configuration x because $F^k(x)$ is clean for any k. The theorem follows (the consequence on B-PRED and REACH are straightforward and the non-universality follows from Theorem 6).

□

Now, we want to show that coherent AND/OR gadgets can be implemented in $\mathcal{F}^{\text{CLOCK},c}_{\text{locally-pos}}$, i.e. we want to show that this family has coherent monotone gadgets. However, in order to accomplish this task we need a construction that we will be useful for the next subsection. Particularly, we need to implement the gadgets that are shown in Figures 13 and 14. Then, we will adapt latter gadgets in order to make it work for locally positive symmetric conjunctive networks. Thus, as a consequence, we will have that $\mathcal{F}^{\text{CLOCK},c}_{\text{locally-pos}}$ is strongly universal.

Theorem 20. *There exist $c > 0$ such that the family $\mathcal{F}^{CLOCK,c}_{locally\text{-}pos}$ of all locally positive symmetric conjunctive networks under local clocks update scheme with clock parameter c has coherent monotone gadgets.*

Proof. We start by observing that the gadgets in Figures 13 and 14 can be implemented in $\mathcal{F}^{\text{CLOCK},c}_{\text{locally-pos}}$ for some c. This can be easily done by adding a positive node to each node in the gadget. The main idea here is that each of these artificial positive nodes will play no role in calculations and will stay in state 1 most of the time. In fact, it suffices that these positive neighbors reach state 1 before critical steps of computation are performed inside the gadget.

In order to illustrate this idea, let us consider two different cases and analyze why computation gadget still works in this case:

1. **Nodes in 4-cycles**: observe that these nodes have a fixed trajectory that is independent on the input that computation part is handling. Thus, it suffices to note that each node in the context effectively changes its state (they are in an attractor of period 3×3 as it is shown in Figure 5). As a consequence of this latter observation, we can set the local period of each positive neighbor so it is updated when its neighbor in the clock is in state 1. More precisely, we fix the corresponding local period value to 9 and correctly initialize them so each positive neighbor is updated exactly when their correspondent node is in state 1.

2. **Central nodes**: Observe that, in this case, we have that in the pseudo-orbit given in Table 3 and Table 4 each node eventually reaches the state 1 independently from the value of x, y, z, x' and y'. Thus, as same as the nodes that are in the 4-cycles, it suffices to set up the local clock of each positive neighbor in order to be updated while its neighbor in the gadget is in state 1. More precisely, it suffices to set up clocks following values in Table 3 and Table 4 and set clocks to be updated every 18 time-steps. Note that this works since the nodes in the central part are in the first block so positive neighbors are updated at the same time as its neighbors but only when nodes in the gadget are in state 1.

Thus, gadgets in Figures 13 and 14 can be implemented as same as we did for general symmetric signed conjunctive networks and desired result holds.

□

Finally, as a direct consequence of latter theorem we have the following corollary:

Corollary 2. *There exist $c > 0$ such that the family $\mathcal{F}^{CLOCK,c}_{locally\text{-}pos}$ of all locally positive symmetric conjunctive networks under local clocks update scheme with clock parameter c is strongly universal.*

4.3 All-positive networks

As same as we did in the previous sections, we use the notation: $\mathcal{F}^{\text{PER},p}_{\text{pos}}, \mathcal{F}^{\text{CLOCK},c}_{\text{pos}}, \mathcal{F}^{\text{BLOCK},b}_{\text{pos}}$. First, we observe that positive conjunctive networks being a particular case of symmetric threshold networks it follows from the classical results of [16, 10], that they always converge to some fixed point or cycle of length two under the parallel update scheme. Therefore they are not dynamically complex with parallel update schedule. It turns out that with a periodic update schedule of period 3 they can break this limitation and produce super-polynomial cycles.

Particularly, we show that positive conjunctive networks with a periodic update schedule of period 3 can break the previous limitation on attractor period and produce superpolynomial cycles. Observe that all the graphs in this family (and also in the other concrete examples we will be exploring in next sections) are non-directed (symmetric). If we observe carefully the effect of periodic update schemes, we note that we are actually changing the interaction graph of the network by considering different orders for updating nodes and thus, breaking the symmetry in the different connections that nodes have in the network. Moreover, we corroborate this remark by showing that actually, we can simulate arbitrary conjunctive networks (defined over directed graphs) by using a periodic update scheme. In fact, we show that the family of symmetric conjunctive networks admits coherent conjunctive gadgets and thus it is capable of simulating the family of directed (non-symmetric) networks.

Proposition 1. *Let $p \geq 1$ and denote by $\mathcal{F}^{PER,p}_{pos}$ be the family of positive conjunctive networks under periodic update schedules of period p. Then, $\mathcal{F}^{PER,p}_{pos}$ is not universal and therefore it does not admit coherent monotone gadgets.*

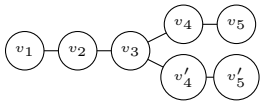
Proof. We actually show that the longest transient of any $F \in \mathcal{F}$ with n nodes is $O(n^2)$, then the conclusions follow by Theorems 6 and 8. Recall that the alphabet is $\{0,1\} \times \{0, \cdots, p\} \times 2^{\{0,\cdots,p\}}$ and, by definition, on any given configuration x the component $\{0, \cdots, p\} \times 2^{\{0,\cdots,p\}}$ is periodic of period p independently of the behavior on the first $\{0,1\}$ component. Moreover, the behavior on the $\{0,1\}$ component is that of a fixed (non-symmetric) conjunctive network F' (depending on F and the update scheme) in the following sense:

$$\forall t \geq 0, F^{t+p}(x) = F'(F^t(x)).$$

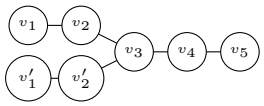
By [23, Theorem 3.20], the transient of any orbit of F' is $O(n^2)$. We deduce that the transient of the orbit of x under F is also $O(n^2)$. □

Theorem 21. *Let $\mathcal{F}^{PER,3}_{pos}$ be the family of positive conjunctive networks under periodic update schedule of period 3. $\mathcal{F}^{PER,3}_{pos}$ has coherent conjunctive gadgets and therefore simulates the family of directed conjunctive networks and for instance can produce superpolynomial cycles.*

Proof. The conclusion about simulations of directed conjunctive networks and superpolynomial cycles follows from 10. We now describe the coherent conjunctive gadgets within family $\mathcal{F}^{\text{PER},3}_{\text{pos}}$ using notations from Definition 9. Let $F_{\text{COPY}} \in \mathcal{F}^{\text{PER},3}_{\text{pos}}$ be defined on the following graph with nodes $V_{\text{COPY}} = \{v_1, v_2, v_3, v_4, v_5, v'_4, v'_5\}$:



Let also $F_{\text{AND}} \in \mathcal{F}$ be defined on the following graph with nodes $V_{\text{AND}} = \{v_1, v_2, v'_1, v'_2, v_3, v_4, v_5\}$:



Recall that both F_{COPY} and F_{AND} have alphabet $Q = \{0,1\} \times \{0,1,2\} \times 2^{\{0,1,2\}}$. Now let $C = C_i \cup C_o$ be the glueing interface with $C_i = \{i\}$ and $C_o = \{o\}$. F_{COPY} is seen as a gadget with one input and two outputs for the gate $\text{COPY} \in \mathcal{G}_{conj}$ as follows:

- $\phi^i_{\text{COPY},1}(i) = v_2$ and $\phi^i_{\text{COPY},1}(o) = v_1$;
- $\phi^o_{\text{COPY},1}(i) = v_5$ and $\phi^o_{\text{COPY},1}(o) = v_4$.
- $\phi^o_{\text{COPY},2}(i) = v'_5$ and $\phi^o_{\text{COPY},2}(o) = v'_4$.

F_{AND} is seen as a gadget with two inputs and one output for the gate $\text{AND} \in \mathcal{G}_{conj}$ as follows:

- $\phi^i_{\text{AND},1}(i) = v_2$ and $\phi^i_{\text{AND},1}(o) = v_1$;
- $\phi^i_{\text{AND},2}(i) = v'_2$ and $\phi^i_{\text{AND},2}(o) = v'_1$.
- $\phi^o_{\text{AND},1}(i) = v_5$ and $\phi^o_{\text{AND},1}(o) = v_4$.

We now define the following elements required by Definition 9:

- the two state configurations s_q for $q \in \{0,1\}$ are defined by $s_q(i) = (q, 0, \{0,2\})$ and $s_q(o) = (1, 0, \{0,1\})$;
- the context configuration is defined by $c(v_3) = (1, 0, \{1,2\})$;
- the time constant is $T = 3$
- the standard trace τ_{q,q^T} over the glueing interface from $q \in \{0,1\}$ to $q^T \in \{0,1\}$ is given by:

time	i	o
0	$(q, 0, \{0,2\})$	$(1, 0, \{0,1\})$
1	$(1, 1, \{0,2\})$	$(q, 1, \{0,1\})$
2	$(1, 2, \{0,2\})$	$(1, 2, \{0,1\})$
3	$(q^T, 0, \{0,2\})$	$(1, 0, \{0,1\})$

- for any $q_i, q_i^T, q_o, q_{o'} \in \{0,1\}$ we have the following $\{v_1, v_5, v'_5\}$-pseudo orbit for F_{COPY}:

time	v_1	v_2	v_3	v_4	v_5	v'_4	v'_5
0	$(q_i, 0, \{0,2\})$	$(1, 0, \{0,1\})$	$(1, 0, \{1,2\})$	$(q_o, 0, \{0,2\})$	$(1, 0, \{0,1\})$	$(q_{o'}, 0, \{0,2\})$	$(1, 0, \{0,1\})$
1	$(1, 1, \{0,2\})$	$(q_i, 1, \{0,1\})$	$(1, 1, \{1,2\})$	$(1, 1, \{0,2\})$	$(q_o, 1, \{0,1\})$	$(1, 1, \{0,2\})$	$(q_{o'}, 1, \{0,1\})$
2	$(1, 2, \{0,2\})$	$(1, 2, \{0,1\})$	$(q_i, 2, \{1,2\})$	$(1, 2, \{0,2\})$	$(1, 2, \{0,1\})$	$(1, 2, \{0,2\})$	$(1, 2, \{0,1\})$
3	$(q_i^T, 0, \{0,2\})$	$(1, 0, \{0,1\})$	$(1, 0, \{1,2\})$	$(q_i, 0, \{0,2\})$	$(1, 0, \{0,1\})$	$(q_i, 0, \{0,2\})$	$(1, 0, \{0,1\})$

- for any $q_i, q_i^T, q_{i'}, q_{i'}^T, q_o \in \{0,1\}$ we have the following $\{v_1, v'_1, v_5\}$-pseudo orbit for F_{AND}:

time	v_1	v_2	v'_1	v'_2	v_3	v_4	v_5
0	$(q_i, 0, \{0,2\})$	$(1, 0, \{0,1\})$	$(q_{i'}, 0, \{0,2\})$	$(1, 0, \{0,1\})$	$(1, 0, \{1,2\})$	$(q_o, 0, \{0,2\})$	$(1, 0, \{0,1\})$
1	$(1, 1, \{0,2\})$	$(q_i, 1, \{0,1\})$	$(1, 1, \{0,2\})$	$(q_{i'}, 1, \{0,1\})$	$(1, 1, \{1,2\})$	$(1, 1, \{0,2\})$	$(q_o, 1, \{0,1\})$
2	$(1, 2, \{0,2\})$	$(1, 2, \{0,1\})$	$(1, 2, \{0,2\})$	$(1, 2, \{0,1\})$	$(q_i \wedge q_{i'}, 2, \{1,2\})$	$(1, 2, \{0,2\})$	$(1, 2, \{0,1\})$
3	$(q_i^T, 0, \{0,2\})$	$(1, 0, \{0,1\})$	$(q_{i'}^T, 0, \{0,2\})$	$(1, 0, \{0,1\})$	$(1, 0, \{1,2\})$	$(q_i \wedge q_{i'}, 0, \{0,2\})$	$(1, 0, \{0,1\})$

□

Interestingly, local clock update schedules on conjunctive networks are not able to produce superpolynomial cycles. To avoid to much conflicts in indices in notations, we denote by x^Q, x^c and x^m the three components of a configuration x in some local clocks extension network.

Lemma 22. *Let us fix any $c > 0$ and consider the family $\mathcal{F}_{pos}^{CLOCK,c}$ of all positive conjunctive networks under local clocks update scheme with clock period c. Fix $n > 0$ and let $F : Q_c^n \to Q_c^n \in \mathcal{F}_{pos}^{CLOCK,c}$. For any configuration $(x^Q, x^c, x^m) \in Q_c^n$ we have that the period of the attractor reached from (x^Q, x^c, x^m) is at most $2\,\mathrm{lcm}\{x_v^m : v \in \{1, \dots, n\}\}$, where* lcm *denotes the least common multiple. Moreover, for each attractor $\overline{x} \in Att(F)$, the set of nodes whose Q component is not constant in $\overline{x}$ induces a bipartite subgraph.*

Proof. Let $(x^Q, x^c, x^m) \in Q_c^n$ be a configuration and $\overline{x}^t = (x^t, c^t, m) \in \mathrm{Att}(F)$ with $t = 1, \dots, p$ be an attractor that is reachable from (x^Q, x^c, x^m) and that is not a fixed point, i.e. $p \geq 2$. Note that by definition $F(\overline{x}^p) = (x^1, c^1, m)$ and that the third componente (the m-component) does not change. We will often use the fact that if two neighbors have Q component 0 in some configuration, then their Q components remain 0 forever. So in particular, it should never happen to a node whose Q component is not constant in the attractor. We'll refer to this as the "double 0 argument". Let us assume that the nodes in the graph are enumerated from 1 to n and let $i \in \{1, \dots, n\}$ be a node such that changes its state, i.e there exists some $t \in [0, \dots, p-1]$ such that $x_i^p = x_i^0$. Without loss of generality we assume that $x_i^0 = 1$ and $c_i^1 = 0$ and $x_i^1 = 0$.

Consider t_1 as the first time step such that the node i changes its state from 0 to 1, which means that, t_1 is the first time step such that $x^{t_1} = 0$ and $x^{t+1} = 1$. Observe that by definition of local clocks update scheme we have $t_1 = s \cdot m_i$ for some $s \geq 1$. In addition, note that, for all $j \in N(i)$ we have $x_j^{t_1} = 1$. Moreover, we have that $x_j^s = 1$ for all $j \in N(i)$ and for all $s \in [1, t_1]$ (see Figure 15). This is because, since the interaction graph is symmetric, j cannot be in state 0 during interval $[1, t_1]$ otherwise both i and j would stay in state 0 forever, contradicting the hypothesis on i (double 0 argument). For this reason, we must also have that $t_1 = m_i$ (*i.e.* $s = 1$) because, otherwise, there must exists some neighbor ℓ of i such that $x_\ell^t = 0$ for $t \in [1, t_1 - 1]$ (to maintain i in state 0 at the update). But then, $x_k^t = x_\ell^t = 0$, which is not possible (double 0 argument).

Now consider t_0 to be the first time step in which the node i changes its state from 1 to 0, i.e. $x^{t_0} = 1$ and $x^{t_0+1} = 0$. So there is some $s^* \geq 1$ such that $t_1 < t_0 = s^* \cdot m_i$. There must exist some neighbor $k \in N(i)$ satisfying that $x_k^{t_0} = 0$, otherwise i cannot change to 0. We are going to show that nodes i and k are synchronized (*i.e.* update exactly at the same time steps) and are always in two distinct states at any given time. First, since we have that node i is in state 0 in the interval $[0, m_i]$ then, $m_k \geq m_i$ (double 0 argument). Consider the maximal time interval I with $t_0 \in I$ during which node k is in state 0. The length of I is exactly m_k because if it was longer, it would mean that an update of node k in state 0 maintain it in 0, which is possible only if one of its neighbors is 0 at the time of the update, which would give a contradiction by double 0 argument. Using double 0 argument again, we must have $I \subseteq]t_1, t_0]$. Since $t_0 \in I$ and $t_0 + 1 \notin I$, node k is actually updating its state at time t_0 like node i. Supposing $m_k > m_i$ would mean that node i updates also at some time step $t \in T \setminus \{t_0\}$ which would imply that it turns in state 0 before step t_0 (because at time t its neighbor k is in state 0) contradicting the choice of t_0. We therefore have $m_k = m_i$ and nodes i and k update exactly at the same time steps (because they do update both at time t_0) and therefore k is in state 0 at time $t_1 + 1$ and therefore $I = [t_0 + 1, t_1]$ and $s^* = 1$ by double 0 argument. From their it is clear that nodes i and k both alternate between 0 and 1 every $m_i = m_k$ steps and that they are never in the same state at the same time.

We have shown that any node i in an attractor as a periodic behavior of period is at most $2m_i$, so we deduce $p(\overline{x}) \leq 2\,\mathrm{lcm}\{m_i : i \in \{1, \dots, n\}\}$, where lcm is the least common multiple of a set of numbers.

Let us go further and show that any neighbor of i whose sate is non constant in the periodic orbit of x^1 is synchronized with i (*i.e.* it changes its state exactly when i does) and is never in the same state as i. Consider a non constant neighbor ℓ of i. As said above it must satisfy $m_\ell \geq m_i$ and must be in state 1 on the interval $[0, t_1 = m_i)$ but also in any interval $]t_0 + rm_i, t_0 + (r+1)m_i]$ with r even by double 0 argument (see Figure 15). Since ℓ is non constant, it must be in state 0 at some time step $t \in]t_0 + rm_i, t_0 + (r+1)m_i]$ for r odd. Supposing $m_\ell > m_i$ would then imply that ℓ is still in state 0 at time $t_0 + (r+1)m_i + 1$ which is impossible as said before. The only possibility is that ℓ is synchronized with i and never in the same state as i. Since ℓ was an arbitrary non constant neighbor of an arbitrary non constant node i, we deduce that the set of nodes whose Q component does not constant during the periodic orbit of an attractor induces a two colorable subgraph. The lemma follows. □

At the light of Lemma 22 and Theorem 21 we observe that there is a big difference between local clocks and periodic update schedules for conjunctive networks in terms of the maximum period of the attractors (constant

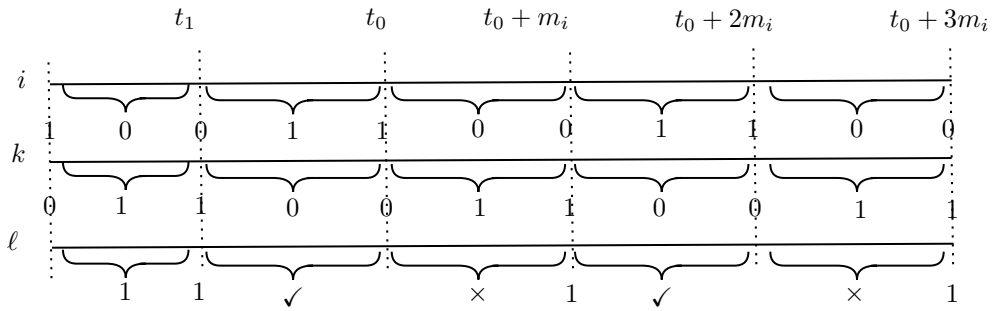


Figure 15: Scheme of the dynamics of nodes i, k and ℓ defined in the proof of Lemma 22. The checkmarks indicate where it is feasible for ℓ to be updated and the crosses mark the intervals on which ℓ cannot be updated.

period v/s superpolynomial). A natural question is whether there is also a difference in terms of the complexity of prediction problems or the transient length. However, the next theorem states that even general periodic update schedules fail to produce maximal complexity (under standard complexity classes separations assumptions).

Theorem 23. *Let $p \in \mathbb{N}$ and consider the family $\mathcal{F}_{pos}^{PER,p}$ of positive conjunctive networks under periodic update schedules of period p. $\mathcal{F}_{pos}^{PER,p}$ is neither dynamically nor computationally complex: more precisely, the transients of any network in $\mathcal{F}_{pos}^{PER,p}$ with n nodes are of length at most $O(n^2)$, the problem U-PRED$_{\mathcal{F}_{pos}^{PER,p}}$ can be solved by a $\boldsymbol{NC}^2$ algorithm and B-PRED$_{\mathcal{F}_{pos}^{PER,p}}$ can be solved in polynomial time.*

Proof. Let $F \in \mathcal{F}_{\text{pos}}^{\text{PER},p}$ with n nodes and consider any initial configuration x. By definition, the orbit of x under F^p is constant on the second and third component of sates, and the action of F^p on the first component when starting from x is a particular non-symmetric conjunctive network F_x that can be seen as an arbitrary Boolean matrix M_x. First, by [23, Theorem 3.20], the transient of the orbit of x under F_x is of length at most $2n^2 - 3n + 2$. We deduce that the transient of x under F is in $O(n^2)$.

Second, it is easy to compute M_x from F and x in $\mathbf{NC}^1$. Moreover, matrix multiplication can be done in $\mathbf{NC}^1$ and by fast exponentiation circuits we can compute M_x^t with polynomial circuits of depth $O(\log(t)\log(n))$. With a constant computational overhead, we can therefore efficiently compute $F^t(x)_v$ and the complexity upper bounds on U-PRED$_{\mathcal{F}_{\text{pos}}^{\text{PER},p}}$ and B-PRED$_{\mathcal{F}_{\text{pos}}^{\text{PER},p}}$ follow. □

Remark 6. *We remark that even when symmetric conjunctive networks under periodic update schemes are not dynamically nor computationally complex, it has been shown that, with a different type of update schemes, it is possible to construct coherent monotone gadgets and thus, show strong universality for this latter family. Particularly, we allude to the case of firing memory update schemes, which can also be studied in the context of our asynchronous extensions framework, as same as periodic update schemes. Just as we have mentioned before, this type of update scheme introduces internal clocks in each node, which will actually depend on the dynamics of the network (contrarily to the case of local clocks, where clocks are fixed and independent from the dynamics of the network). This key aspect gives the network strong dynamical and computational capabilities which can be used to simulate monotone boolean networks. Interested reader is referred to [13] in order to see details.*

5 Counter examples to the desymmetrization phenomena

At the light of the previous results we can identify an interesting phenomenon: a base concrete family over symmetric graphs is able to simulate its non-symmetric (directed) version when general update schemes are considered (periodic updates for conjunctive networks). In that sense, one can wonder how general is the phenomenon of desymmetrization by asynchronism. In this section, we give two examples of CSAN families that have opposite behaviors when one compares their periodic update extension and their non-symmetric version with parallel update. Let us first start by an example where the non-symmetric version is strictly more powerful than the periodic update extension.

Proposition 2. *There exists a CSAN family $\mathcal{F}$ such that for every $p \in \mathbb{N}$ its periodic update extension $\mathcal{F}_p$ admits attractors of period at most 2 and its non-symmetric version $\vec{\mathcal{F}}$ admits attractors of non-constant period. As a consquence, $\mathcal{F}$ is such that none of its periodic update extension $\mathcal{F}_p$ can simulate its non-symmetric version $\vec{\mathcal{F}}$, whatever $p \in \mathbb{N}$.*

Proof. Let $Q = \{0,1\} \times \{0,1\} \cup \{e\}$ and denote by π_1 and π_2 the projections on the first and second components respectively for states in $\{0,1\} \times \{0,1\}$. We construct a family which essentially applies a majority rule on one component of composite states, it does synchronization checks using the second component, and use e as an error state that spreads over the network. More precisely, consider the local map $\lambda : Q \times \mathbb{N}^Q \to Q$ that does the following at each node v:

- if e appears in the neighborhood, then the image by λ at v is e;
- otherwise, let M_1 be the multiset of states of the π_1 components of states of neighbors and the the node itself and denote by m_1 the state with a majority of occurrences in M_1 (π_1 of the state of v in case of tie); then
  - if all neighbors of node v (including v) have the same π_2 component b, then the image by λ at v is $(m_1, 1-b)$;
  - otherwise, the image by λ at v is e.

The CSAN family $\mathcal{F}$ we consider is the one defined by this local rule λ at each node. Consider any automata network of the periodic update extension of $\mathcal{F}$ of period p, and consider any limit cycle of some of its orbits. We show that this limit cycle has length at most 2. There are two cases:

- if the update scheme is not a synchronous update scheme (all nodes update exactly at the same time steps), then there must be some time step at which a node v is updated but some of its neighbors v' isn't. So either v updates to e at this time step, or v updates to a composite state with a π_2 component different from the π_2 component of v'. In the second case the next node to update among v and v' will update to e. Finally, since e appears at some node in the orbit, this means that the limit cycle is the fixed point configuration everywhere equal to e.
- if the update scheme is the parallel update scheme, then: either e appears in the considered orbit and the limit cycle is a fixed point; or e never appears, which means that the π_2 component of nodes is oscillating between 0 and 1 and the behavior in the π_1 component is that of the classical majority rule with parallel update mode (which can only exhibit attractors of period at most 2 as shown in [16]), hence also of period at most 2.

We conclude that the family $\mathcal{F}_p$ has only limit cycles of length ≤ 2, whatever p.

If we consider now the non-symmetric version $\vec{\mathcal{F}}$, it is straightforward to build limit cycle of length n (size of the network) using synchronized π_2 components. More precisely, if we consider a configuration x such that the π_2 component is defined as 1 for every node in the graph, i.e. $\pi_2(x_v) = 1$ for all node v then, each node will update its state following the classical majority rule in its π_1 component. In addition, the component π_2 will oscillate, meaning that $\pi_2(F(x)_v) = 0$ whenever $\pi_2(x_v) = 1$ and $\pi_2(F(x)_v) = 1$ whenever $\pi_2(x_v) = 0$.

Thus, we have that $\mathcal{F}_p$ cannot simulate $\vec{\mathcal{F}}$ (actually, it is not hard to show that $\vec{\mathcal{F}}$ is strongly universal since it admits coherent monotone gadgets). □

Let us now give an example of CSAN family where the opposite situation occurs: periodic extensions are stronger than the non-symmetric version.

Proposition 3. *There exists a CSAN family $\mathcal{F}$ such that its non-symmetric version $\vec{\mathcal{F}}$ admits only fixed points but for some $p \in \mathbb{N}$ its periodic update extension $\mathcal{F}_p$ admits attractors of non-constant period. As a consequence, there exists a CSAN family $\mathcal{F}$ such that its periodic update extension $\mathcal{F}_p$ for some p cannot be simulated by its non-symmetric version $\vec{\mathcal{F}}$.*

Proof. Consider any CSAN family $\mathcal{F}_0$ over alphabet Q than can produce orbits with limit cycles of non-constant length over symmetric networks that are bipartite: one can for instance choose $\mathcal{F}_0$ to be the family of positive conjunctive networks under periodic update schedule of period 3 (see Theorem 21). Define a new alphabet $Q' = Q \cup Q \times \{0,1\} \cup \{e\}$ where we distinguish three types of states: *pure* states from Q, *composite* states from $Q \times \{0,1\}$ and the error state e. For a non-error state, its *Q-content* is either itself (when it is pure) or the projection on its Q component (when it is composite). Let's define the family $\mathcal{F}$ by the following local behavior on any graph:

- e states spread over the network, *i.e.* any node with an incoming neighbor in state e updates to state e;
- if a node has a pure (resp. composite) state, then all its incoming neighbors must have a composite (resp. pure) state otherwise it updates to e in one step; moreover, any pure state with some incoming neighbor in a state from $Q \times \{0\}$ will also update to e (so when a node in a pure state is updated, all its incoming neighbors must be in a state from $Q \times \{1\}$ to prevent it from updating to e);
- outside the above cases, the type of a state doesn't change and
  - if it is a pure state, its Q-content changes according to the local rules over Q of the chosen CSAN family $\mathcal{F}_0$;
  - if it is a composite state, its $\{0,1\}$ component is inverted (by $b \mapsto 1-b$), and its Q-content is updated according to the local rules over Q as above, but only when the $\{0,1\}$ component is 1, otherwise the Q-content is left unchanged.

Now take any connected automata network from the non-symmetric version $\vec{\mathcal{F}}$ of the family we just defined. We claim that any orbit reaches a fixed-point. First, it is well-known (see [22]) that if the graph is acyclic, then all orbits converge to a fixed-point. So let's consider the case where the graph possesses a directed cycle C. Consider some configuration x, and let us show that e must appear in the network at some step which implies that the orbit will reach a fixed-point. If the type of states does not alternate between pure and composite along cycle C, then e is generated in one step. If the type of states correctly alternates in C between pure and composite, then there must be a first step t where some node of C is in a composite state with value 0 on its $\{0,1\}$ component. Since C is a cycle, this node is the incoming neighbor of some other node in a pure state. This last node will update to e in this situation. This concludes the claim that any orbit reaches a fixed point in any network of $\vec{\mathcal{F}}$.

Let us now consider any bipartite symmetric network and some orbit of an automata network of family $\mathcal{F}_0$ reaching a long limit cycle. We can simulate any orbit starting from some Q-configuration x by applying a periodic update scheme of period 2 to some automata network of family $\mathcal{F}$ on the same network as follows: by bi-partition of the network we can distribute the type of states (pure/composite) in such a way that no node has a neighbor with the same type as itself; then we update composite nodes at each step and pure nodes once every two steps (and start to update them at first step) ; we choose as initial configuration a configuration where the Q-contents correspond to x and all composite node start with 1 in their $\{0,1\}$ component.

Thus, we have shown that the periodic extension of network admits attractors of non-constant period (because $\mathcal{F}_0$ satisfies this property and we have shown that the orbits of $\mathcal{F}_0$ can be simulated by $\mathcal{F}$ equipped with some periodic update scheme) and that the non-symmetric extension $\vec{\mathcal{F}}$ admits only fixed points. This shows that the periodic update extension of period 2 of $\mathcal{F}$ cannot be simulated by its non-symmetric version $\vec{\mathcal{F}}$. □

6 Perspectives

The main results of this paper extend previous results [13, 15, 12] on the strong interplay between symmetry of local interactions and update modes. We showed how universality can be recovered from the symmetry constraint by allowing various degrees of asynchronism. An immediate question following our work is to find a natural example of a CSAN family which is (strongly) universal under periodic update schemes, but not universal under local clocks update schemes.

Besides universality, it would be interesting to better understand in general when a symmetric family can simulate its non-symmetric version under non-parallel update modes (as it was shown in Theorem 21). Here again, we are interested in the level of asynchronism required for this simulation to hold.

Another natural research direction is to try to further extend this analysis to other update modes. As we have pointed out in the update schemes section (see Remark 3), it would be interesting to study other types of update schemes which are not captured by general periodic update schemes such as firing memory schemes [13, 11], or other modes where the decision to apply an update at a node depends on its state (for instance we can interpret reaction-difussion systems [8] as threshold networks with an update mode with memory). We can consider even more general ones inspired by the most permissive semantics studied in [4]. In some cases, Definition 12 of asynchronous extension should be adapted in order to capture latter update schemes.

References

[1] J. Aracena, E. Goles, A. Moreira, and L. Salinas. On the robustness of update schedules in boolean networks. *Biosystems*, 97(1):1–8, jul 2009.

[2] J. Aracena, A. Richard, and L. Salinas. Fixed points in conjunctive networks and maximal independent sets in graph contractions. *J. Comput. System Sci.*, 88:145–163, 2017.

[3] Eric Goles Ch. and Pedro Montealegre. Computational complexity of threshold automata networks under different updating schemes. *Theor. Comput. Sci.*, 559:3–19, 2014.

[4] Thomas Chatain, Stefan Haar, Loïc Paulevé, et al. Most permissive semantics of boolean networks. *arXiv preprint arXiv:1808.10240*, 2018.

[5] Jacques Demongeot and Sylvain Sené. About block-parallel boolean networks: a position paper. *Nat. Comput.*, 19(1):5–13, 2020.

[6] Maximilien Gadouleau. On the influence of the interaction graph on a finite dynamical system. *Natural Computing*, 19(1):15–28, feb 2019.

[7] Maximilien Gadouleau and Søren Riis. Memoryless computation: new results, constructions, and extensions. *Theoretical Computer Science*, 562:129–145, January 2015.

[8] E Goles and Martin Matamala. Reaction-diffusion automata: Three states implies universality. *Theory of Computing Systems*, 30(3):223–229, 1997.

[9] E. Goles and M. Noual. Disjunctive networks and update schedules. *Adv. Appl. Math.*, 48:646–662, 2012.

[10] E. Goles and J. Olivos. Periodic behaviour of generalized threshold functions. *Discrete Mathematics*, 30(2):187 – 189, 1980.

[11] Eric Goles, Fabiola Lobos, Gonzalo A. Ruz, and Sylvain Sené. Attractor landscapes in Boolean networks with firing memory. *Natural Computing*, 19:295–319, 2019.

[12] Eric Goles and Pedro Montealegre. Computational complexity of threshold automata networks under different updating schemes. *Theoretical Computer Science*, 559:3–19, 2014.

[13] Eric Goles, Pedro Montealegre, and Martín Ríos-Wilson. On the effects of firing memory in the dynamics of conjunctive networks. *Discrete & Continuous Dynamical Systems - A*, 40(10):5765–5793, 2020.

[14] Eric Goles, Pedro Montealegre, Ville Salo, and Ilkka Törmä. Pspace-completeness of majority automata networks. *Theor. Comput. Sci.*, 609:118–128, 2016.

[15] Eric Goles, Pedro Montealegre, Ville Salo, and Ilkka Törmä. Pspace-completeness of majority automata networks. *Theoretical Computer Science*, 609:118–128, 2016.

[16] E Goles-Chacc, F Fogelman-Soulie, and D Pellegrin. Decreasing energy functions as a tool for studying threshold networks. *Discrete Applied Mathematics*, 12(3):261–277, 1985.

[17] Stuart Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224(5215):177–178, oct 1969.

[18] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, dec 1943.

[19] Martín Ríos-Wilson and Guillaume Theyssier. Intrinsic universality in automata networks i: Families and simulations. *Theoretical Computer Science*, 997:114511, 2024.

[20] Martín Ríos-Wilson and Guillaume Theyssier. Intrinsic universality in automata networks ii: Glueing and gadgets. *Theoretical Computer Science*, page 114779, 2024.

[21] F. Robert. Blocs-h-matrices et convergence des methodes iteratives classiques par blocs. *Linear Algebra and its Applications*, 2(2):223–265, apr 1969.

[22] François Robert. Iterations sur des ensembles finis et automates cellulaires contractants. *Linear Algebra and its applications*, 29:393–412, 1980.

[23] Bart De Schutter and Bart De Moor. On the sequence of consecutive powers of a matrix in a boolean algebra. *SIAM Journal on Matrix Analysis and Applications*, 21(1):328–354, jan 1999.

[24] René Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42(3):563–585, dec 1973.

[25] Angela Wu and Azriel Rosenfeld. Cellular graph automata. i. basic concepts, graph property measurement, closure properties. *Information and Control*, 42(3):305 – 329, 1979.

[26] Angela Wu and Azriel Rosenfeld. Cellular graph automata. ii. graph and subgraph isomorphism, graph structure recognition. *Information and Control*, 42:330–353, 09 1979.