Intrinsic Universality in Automata Networks II: Glueing and Gadgets

Martín Ríos Wilson, Guillaume Theyssier

Intrinsic Universality in Automata Networks II: Glueing and Gadgets

Martín Ríos-Wilson[a,b], Guillaume Theyssier[b]

[a] *Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez.*

[b] *Aix-Marseille Université, CNRS, I2M (UMR 7373), Marseille, France.*

Abstract

An automata network (AN) is a finite graph where each node holds a state from a finite alphabet and is equipped with a local map defining the evolution of the state of the node depending on its neighbors. This paper is the second of a series about intrinsic universality, i.e. the ability for a family of AN to simulate arbitrary AN. We develop a proof technique to establish intrinsic simulation and universality results which is suitable to deal with families of symmetric networks where connections are non-oriented. It is based on an operation of glueing of networks, which allows to produce complex orbits in large networks from compatible pseudo-orbits in small networks. As an illustration, we give a short proof that the family of networks where each node obeys the rule of the 'game of life' cellular automaton is strongly universal. In the third paper of the series, we heavily rely on this proof technique to show intrinsic universality results of various families with particular update schedules.

Keywords: automata networks, intrinsic universality, glueing, game of life

1. Introduction

Automata networks (AN) are finite dynamical systems introduced in the 40s [1] which are commonly used as models of biological networks [2, 3] or computational models [4, 5, 6, 7, 8]. Although they can be defined as a global map of the form $F : Q^n \to Q^n$ for some alphabet Q, many natural families are better described in a succinct way as a graph where each vertex holds a state and is equipped with a local transition map that determines how the state of the node evolves depending on the states of neighboring nodes. This way, they can be seen as non-uniform cellular automata on arbitrary graphs. Beyond applications, AN theory has been developed in several directions, two of them being the combinatorics of their orbits and computational complexity of various decision problems like prediction or reachability [9, 10, 11, 5, 12, 13, 14, 15, 16, 17, 18]. It turns out that most results focus on a particular problem or dynamical parameter.

This paper is the second of a series of three about intrinsic universality in AN, *i.e.* the ability for a specific family of AN to simulate arbitrary AN. The main interest of this notion is that it has consequences both in terms of dynamics and computational complexity. In the first paper [19], we established a precise formalism of families of AN, intrinsic simulations and (strong) intrinsic universality. The present paper focuses on a proof technique to establish

[1] 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.) and ANID FONDECYT Postdoctorado 3220205 (M.R-W).

intrinsic simulation and universality. At the core of our approach is a concept of *glueing* which is a way to merge two or more ANs sharing a common sub-network. We show how glueing can be useful to build a large network with a prescribed behaviors from compatible pseudo-orbits of small networks. We establish a general composition result that allows to derive intrinsic universality of a family from the existence of a finite set of compatible pseudo-orbits of a finite set of small networks.

A common aspect to most hardness results for automata networks is the formulation of gadgets simulating logic gates in order to perform a reduction to classic computational complexity problems such as the circuit value problem or the boolean satisfiability problem. Since there are similarities in the way these gadgets are constructed, it is usually accepted that the mere existence of gadgets imply that the reductions are correct because a 'simulation of Boolean circuit' is achieved. To our knowledge, no formalism for gadgets and their composition has been proposed so far, and we see two problems in this situation:

- First, it is not clear what 'simulating a Boolean circuit with gadgets' means since the asynchronous nature of Boolean circuit evaluation does not match the extreme sensitivity to synchronism of automata networks; moreover, in a symmetric (non-oriented) network family like in the examples above, gadgets have no clear notion of input and output nodes and, due to potential unwanted feedback behaviors, it is not as simple to correctly connect them as it is to connect logic gates in a Boolean circuit.
- Second, the lack of gadget formalism pushes the authors to state results about hardness of a particular decision problem or lower bound of a particular dynamical parameter: this approach is not modular since, as illustrated in the first paper of this series [19], complexity of one particular aspect does not generally imply complexity of another. We believe for instance that proving intrinsic universality is a much more far reaching objective than proving the hardness of a particular decision problem.

The present paper aims at establishing precise sufficient conditions for intrinsic universality of a family that boils down to the existence a finite set of objects. Together with the dynamical and computational complexity consequences of universality established in [19], this provides a powerful tool to prove many lower bounds or hardness results at once on a family of AN.

The detailed contributions of the present paper are as follows:

1. We consider several families of automata networks based on a finite set of local update rules called $\mathcal{G}$-networks (Definition 6), and show various simulations or (non-)universality results on standard examples (Section 3.4 and Theorems 21 and 22 and 24).
2. We develop a proof tool for intrinsic simulation results in the context of automata networks on non-directed graphs: it is based on a way to compose small networks into a larger one called *glueing* (Definition 7) that preserves pseudo-orbits (Lemma 8) and can be directly used to prove simulation of $\mathcal{G}$-network families through a concept of gadgets (Lemma 13).
3. Based on previously analyzed families of $\mathcal{G}$-networks, we obtain a sufficient condition for intrinsic universality

that boils down to the existence of a coherent finite set of pseudo-orbits of a finite set of networks from the considered family (Corollary 4).

4. We apply this techniques on the family of automata networks on arbitrary undirected graphs where each node behaves like the cells of the famous 'Game of Life' cellular automaton, and establish strong universality of the family (Theorem 18) by new gadgets that are smaller and more time-efficient than the classical ones used to prove intrinsic universality of the cellular automaton on the grid.

2. Preliminaries

Graph definitions. 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 $\delta^+ = |N^+(v)|$

Automata networks During the most part of of the text, and unless explicitly stated otherwise, every graph G will be assumed to be connected and undirected. We start by stating the following basic definitions, notations and properties that we will be using in the next sections. In general, Q and V will denote finite sets representing the alphabet and the set of nodes respectively. We define $\Sigma(Q)$ as the set of all possible permutations over alphabet Q. We call an *abstract automata network* any function $F : Q^V \to Q^V$. Note that F induces a dynamics in Q^V and thus we can see (Q^V, F) as dynamical system. In this regard, we recall some classical definitions. We call a *configuration* 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.

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 $F^{\tau+p}(x) = F^\tau(x)$ for all $x \in Q^V$. 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. We define the *global period* or simply the *period* of $\overline{x} \in \mathrm{Att}(F)$ 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*.

Given a node v, its behavior $x \mapsto F(x)_v$ might depend or not on another node u. This dependencies can be captured by a graph structure which plays an important role in the theory of automata networks (see [20] for a review of known results on this aspect). This motivates the following definitions.

Definition 1. *Let $F : Q^V \to Q^V$ be an abstract automata network and $G = (V, E)$ a directed graph. We say G is a* communication graph *of F if for all $v \in V$ there exist $D \subseteq N_v^-$ and some function $f_v : Q^D \to Q$ such that $F(x)_v = f_v(x|_D)$. The* interaction graph *of F is its minimal communication graph.*

Note that by minimality, for any node v and any in-neighbor u of v in the interaction graph of some F, then the next state at node v effectively depends on the actual state at node u. More precisely, there is some configuration $c \in Q^V$ and some $q \in Q$ with $q \neq c_u$ such that $F(c)_v \neq F(c')_v$ where c' is the configuration c where the state of node u is changed to q. This notion of effective dependency is sometimes taken as a definition of edges of the interaction graph.

From now on, for an abstract automata network F and some communication graph G of F we use the notation $\mathcal{A} = (G, F)$. In addition, by abuse of notation. we also call $\mathcal{A}$ an abstract automata network. We define a set of automata networks or an *abstract family* of automata networks on some alphabet Q as a set $\mathcal{F} \subseteq \bigcup_{n \in \mathbb{N}} \{F : Q^V \to Q^V : V \subseteq [n]\}$.

A *multiset* over Q is a map $m : Q \to \mathbb{N}$ (recall that $0 \in \mathbb{N}$). 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 2. *Given a non-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 symmetric automata network (CSAN) $\mathcal{A} = (G, \lambda, \rho)$. A* family of concrete symmetric automata networks *(CSAN family) $\mathcal{F}$ is given by an alphabet Q, 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 CSAN (G, λ, ρ) belongs to $\mathcal{F}$ if for any vertex v of G with incident edges E_v it holds $(\lambda(v), \rho(E_v)) \in \mathcal{C}$.*

Representations and simulations.

Definition 3. *Let $\mathcal{F}$ be a set of abstract automata network over alphabet Q. A standard representation $\mathcal{F}^*$ for $\mathcal{F}$ is a language $L_{\mathcal{F}} \subseteq \{0,1\}^*$ together with a **DLOGSPACE** algorithm such that:*

- *the algorithm transforms any $w \in L_{\mathcal{F}}$ into the canonical representation of a circuit encoding $C(w)$ that code an abstract automata network $F_w \in \mathcal{F}$;*
- *for any $F \in \mathcal{F}$ there is $w \in L_{\mathcal{F}}$ with $F = F_w$.*

As it is described in [19], CSAN families are a collection of labeled graphs and thus they are naturally represented as a graph G together with some representations of local functions (i.e. λ and ρ).

Definition 4. *Let $(\mathcal{F}, \mathcal{F}^*)$ and $(\mathcal{H}, \mathcal{H}^*)$ be two families with standard representations on alphabets Q_F and Q_H respectively. Let $T, S : \mathbb{N} \to \mathbb{N}$ be two functions. We say that $\mathcal{F}^*$ simulates $\mathcal{H}^*$ in time T and space S if there exists a* ***DLOGSPACE*** *Turing machine M such that for each $w \in L_{\mathcal{H}}$ representing some automata network $H_w \in \mathcal{H} : Q_H^n \to Q_H^n$, the machine produces a pair $M(w)$ which consists in:*

- *$w' \in L_{\mathcal{F}}$ with $F_{w'} : Q_F^{n_F} \to Q_F^{n_F}$,*
- *$T(n)$ and a representation of a block embedding $\phi : Q^{n_F} \to Q^n$,*

such that $n_F = S(n)$ and $F_{w'}$ simulates H_w in time $T = T(n)$ under block embedding ϕ.

The orbit graph G_F associated to a network F with nodes V and alphabet Q is the digraph with vertices Q^V and an edge from x to $F(x)$ for each $x \in Q^V$. We also denote $G_F^t = G_{F^t}$.

Definition 5. *Fix a map $\rho : \mathbb{N} \to \mathbb{N}$, we say that the orbit graph G_F of F with n nodes is ρ-succinct if F can be represented by circuits of size at most $\rho(n)$. We say that the orbit graph G_H of H with m nodes embeds G_F with distortion $\delta : \mathbb{N} \to \mathbb{N}$ if $m \leq \delta(n)$ and there is $T \leq \delta(n)$ such that G_F is a subgraph of G_{H^T}.*

3. Gadgets and glueing

In the same way as Boolean circuits are defined from Boolean gates, many automata network families can be defined by fixing a finite set of local maps $\mathcal{G}$ that we can freely connect together to form a global network, called a $\mathcal{G}$-network.

Such families can be strongly universal as we will see, even for very simple choices of $\mathcal{G}$, which is an obvious motivation to consider them. In this section, we introduce a general framework to prove simulation results of a $\mathcal{G}$-network family by some arbitrary family that amounts to a finite set of conditions to check. From this we will derive a framework to certify strong universality of an arbitrary family just by exhibiting a finite set of networks from the family that verify a finite set of conditions. As already said above, our goal is to analyze automata networks with symmetric communication graph (CSAN families). Our framework is targeted towards such families.

The idea behind is that of building large automata networks from small automata networks in order to mimic the way a $\mathcal{G}$-network is built from local maps in $\mathcal{G}$. The difficulty, and the main contribution of this section, is to formalize how small building blocks are glued together and what conditions on them guaranty that the large network correctly simulates the corresponding $\mathcal{G}$-networks. In particular, our formalism is perfectly suited to show that a family of *undirected* networks can simulate a family of *oriented* $\mathcal{G}$-networks.

We will now introduce all the concepts used in this framework progressively.

3.1. $\mathcal{G}$-networks

Let Q be a fixed alphabet and $\mathcal{G}$ be any set of maps of type $g : Q^{i(g)} \to Q^{o(g)}$ for some $i(g), o(g) \in \mathbb{N}$. We say g is *reducible* if it can be written as a disjoint union of two gates, and *irreducible* otherwise. Said differently, if G is the

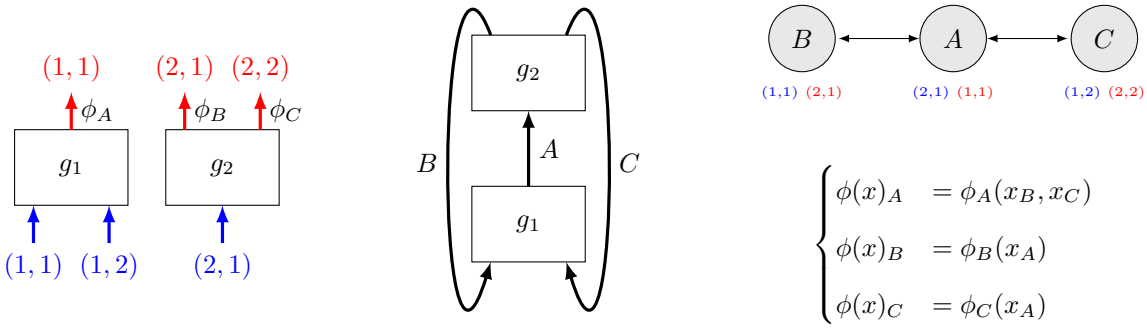


Figure 1: On the left a set of maps $\mathcal{G}$ over alphabet Q, in the middle an intuitive representation of input/output connections to make a $\mathcal{G}$-network, on the right the corresponding formal $\mathcal{G}$-network $\phi : Q^3 \to Q^3$ together with the global map associated to it. The bijections α and β from Definition 6 are represented in blue and red (respectively).

(bipartite) dependency graph of g describing on which inputs effectively depends each output, then g is irreducible if G is weakly connected.

From $\mathcal{G}$ we can define a natural family of networks: a $\mathcal{G}$-network is an automata network obtained by wiring outputs to inputs of a number of gates from $\mathcal{G}$. To simplify some later results, we add the technical condition that no output of a gate can be wired to one of its inputs (no self-loop condition).

Definition 6. *A $\mathcal{G}$-network is an automata network $F : Q^V \to Q^V$ with set of nodes V associated to a collection of gates $g_1, \dots, g_n \in \mathcal{G}$ with the following properties. Let*

$$I = \{(j,k) : 1 \le j \le n \text{ and } 1 \le k \le i(g_j)\} \text{ and}$$
$$O = \{(j,k) : 1 \le j \le n \text{ and } 1 \le k \le o(g_j)\}$$

be respectively the sets of inputs and outputs of the collection of gates $(g_j)_{1 \le j \le n}$. We require $|V| = |I| = |O|$ and the existence of two bijective maps $\alpha : I \to V$ and $\beta : V \to O$ with the condition that there is no $(j,k) \in I$ such that $\beta(\alpha(j,k)) = (j,k')$ for some k' (no self-loop condition). For $v \in V$ with $\beta(v) = (j,k)$, let $I_v = \{\alpha(j,1), \dots, \alpha(j, i(g_j))\}$ and denote by g_v the map: $x \in Q^{I_v} \mapsto g_j(\tilde{x})_k$ where $\tilde{x} \in Q^{i(g_j)}$ is defined by $\tilde{x}_k = x_{\alpha(j,k)}$. Then F is defined as follows:

$$F(x)_v = g_v(x_{I_v}).$$

Remark 1. *Once $\mathcal{G}$ is fixed, there is a bound on the degree of dependency graphs of all $\mathcal{G}$-networks. Thus, it is convenient to represent $\mathcal{G}$-networks by the standard representation of bounded degree automata networks (as a pair of a graph and a list of local update maps). Another representation choice following strictly Definition 6 consists in giving a list of gates $g_1, \dots, g_k \in \mathcal{G}$, fixing $V = \{1, \dots, n\}$ and give the two bijective maps $\alpha : I \to V$ and $\beta : V \to O$ describing the connections between gates (maps are given as a simple list of pairs source/image). One can check that these two representations are **DLOGSPACE** equivalent when the gates of $\mathcal{G}$ are irreducible: we can construct*

the interaction graph and the local maps from the list of gates and maps α and β in ***DLOGSPACE*** *(the incoming neighborhood of a node v, I_v, and its local map g_v are easy to compute as detailed in Definition 6); reciprocally, given the interaction graph G and the list of local maps (g_v), one can recover in* ***DLOGSPACE*** *the list of gates and their connections as follows:*

- *for v from 1 to n do:*
  - *gather the (finite) incoming neighborhood $N^-(v)$ of v then the (finite) outgoing neighborhood $N^+(N^-(v))$ and iterate this process until it converges (in finite time) to a set I_v of inputs and O_v of outputs with $v \in O_v$;*
  - *check that all $v' \in I_v \cup O_v$ are such that $v' \geq v$ otherwise jump to next v in the loop (this guaranties that each gate is generated only once);*
  - *since the considered gates are irreducible, I_v and O_v actually correspond to input and output sets of a gate $g \in \mathcal{G}$ that we can recover by finite checks from the local maps of nodes in O_v;*
  - *output gate g and the pairs source/image to describe α and β for nodes in I_v and O_v respectively.*

In the sequel we denote $\Gamma(\mathcal{G})$ the family of all posible $\mathcal{G}$-networks associated to their bounded degree representation.

3.2. Glueing of automata networks

In this section we define an operation that allows us to 'glue' two different abstract automata networks on a common part in order to create another one which, roughly, preserve some dynamical properties in the sense that it allows to glue pseudo-orbits of each network to obtain a pseudo-orbit of the glued network. One might find useful to think about the common part of the two networks as a dowel attaching two pieces of wood: each individual network is a piece of wood with the dowel inserted in it, and the result of the glueing is the attachment of the two pieces with a single dowel (see Figure 2).

Definition 7. *Consider $F_1 : Q^{V_1} \to Q^{V_1}$ and $F_2 : Q^{V_2} \to Q^{V_2}$ two automata networks with V_1 disjoint from V_2, C a set disjoint from $V_1 \cup V_2$, $\varphi_1 : C \to V_1$ and $\varphi_2 : C \to V_2$ two injective maps with $\varphi_1(C) \cap \varphi_2(C) = \emptyset$ and C_1, C_2 a partition of C in two sets. We define*

$$V' = C \cup (V_1 \setminus \varphi_1(C)) \cup (V_2 \setminus \varphi_2(C))$$

and the map $\alpha : V' \to V_1 \cup V_2$ by

$$\alpha(v) = \begin{cases} v & \text{if } v \notin C \\ \varphi_i(v) & \text{if } v \in C_i, \text{ for } i = 1, 2. \end{cases}$$

We then define the glueing of F_1 and F_2 over C as the automata network $F' : Q^{V'} \to Q^{V'}$ where

$$F'_v = \begin{cases} (F_1)_{\alpha(v)} \circ \rho_1 & \text{if } \alpha(v) \in V_1, \\ (F_2)_{\alpha(v)} \circ \rho_2 & \text{if } \alpha(v) \in V_2, \end{cases}$$

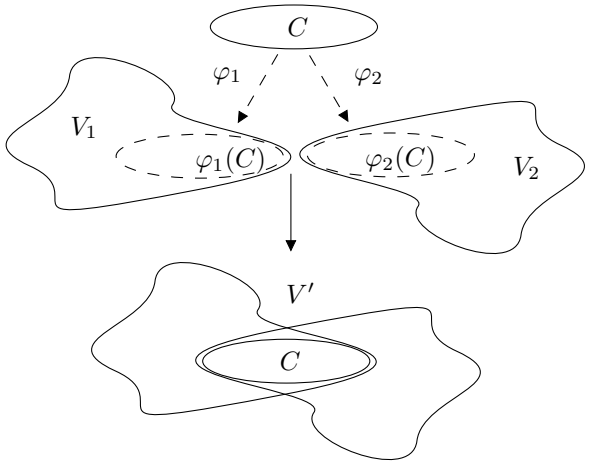


Figure 2: Scheme of a glueing

where $\rho_i : Q^{V'} \to Q^{V_i}$ *is defined by*

$$\rho_i(x)_v = \begin{cases} x_{\varphi_i^{-1}(v)} & \text{if } v \in \varphi_i(C), \\ x_v & \text{else.} \end{cases}$$

When necessary, we will use the notation $F' = F_1 \overset{\phi_1}{C_1} \oplus \overset{\phi_2}{C_2} F_2$ to underline the dependency of the glueing operation on its parameters.

Given an automata network $F : Q^V \to Q^V$ and a set $X \subseteq V$, we say that a sequence $(x^t)_{0 \leq t \leq T}$ of configurations from Q^V is a X-pseudo-orbit if it respects F as in a normal orbit, except on X where it can be arbitrary, formally: $x_v^{t+1} = F(x^t)_v$ for all $v \in V \setminus X$ and all $0 \leq t < T$. The motivation for Definition 7 comes from the following lemma.

Lemma 8 (Pseudo-orbits glueing)**.** *Taking the notations of Definition 7, let* $X \subseteq V_1 \setminus \varphi_1(C)$ *and* $Y \subseteq V_2 \setminus \varphi_2(C)$ *be two (possibly empty) sets. If* $(x^t)_{0 \leq t \leq T}$ *is a* $X \cup \varphi_1(C_2)$*-pseudo-orbit for* F_1 *and if* $(y^t)_{0 \leq t \leq T}$ *is a* $Y \cup \varphi_2(C_1)$*-pseudo-orbit for* F_2 *and if they verify for all* $0 \leq t \leq T$

$$\forall v \in C, x^t_{\varphi_1(v)} = y^t_{\varphi_2(v)}, \tag{1}$$

then the sequence $(z^t)_{0 \leq t \leq T}$ *of configurations of* $Q^{V'}$ *is a* $X \cup Y$*-pseudo-orbit of* F'*, where*

$$z_v^t = \begin{cases} x^t_{\alpha(v)} & \text{if } \alpha(v) \in V_1, \\ y^t_{\alpha(v)} & \text{if } \alpha(v) \in V_2. \end{cases}$$

Proof. Take any $v \in V' \setminus (X \cup Y)$. Suppose first that $\alpha(v) \in V_1$. By definition of F', we have $F'(z^t)_v = (F_1)_{\alpha(v)} \circ \rho_1(z^t)$ but $\rho_1(z^t) = x^t$ (using the Equation 1 in the hypothesis) so $F'(z^t)_v = F_1(x^t)_{\alpha(v)}$. Since (x^t) is a $X \cup \varphi_1(C_2)$-pseudo-orbit and since $\alpha(v) \notin X \cup \varphi_1(C_2)$, we have

$$F_1(x^t)_{\alpha(v)} = x^{t+1}_{\alpha(v)} = z_v^{t+1}.$$

We conclude that $z_v^{t+1} = F'(z^t)_v$. By a similar reasoning, we obtain the same conclusion if $\alpha(v) \in V_2$. We deduce that (z^t) is a $X \cup Y$-pseudo-orbit of F'. □

In order to illustrate the latter lemma, we show an example of glueing considering classical life-like automata network, given by the *Game of life*. In this case, we have that $B = \{3\}$ and $S = \{2, 3\}$ meaning that a dead cell can update its state to alive if it has exactly three alive neighbors and it will survive only it has exactly 2 or 3 alive neighbors.

First, we introduce, In Figure 3, the dynamics of a clock network (roughly, a network exhibiting a dynamics consisting in a periodic sequence of patterns that move from left to right). This network will be very important to the construction we will show in the next section and for the example of pseudo-orbit glueing that we are going to introduce hereunder. Now, observe that the communication graph of the clock network is composed by six layers of three independent nodes connected to all the nodes in the next layer. The layers in gray boxes (the first and the last) are connected. In addition, all the layers are connected to an auxiliary node. This node will allow the layer to return to state 0 when the next layer is in state 1 since it will provide an additional node in state 1 so all the nodes in the layer will have exactly four nodes in state 1 (note that each node in a layer has three neighbors in the adjacent layers and the auxiliary node). Observe that in each time step two layers are in state one and the rest o the layers are in state zero. This pattern is shifted in each time step and thus it takes 6 time steps in order to go from the first layer to the last one.

Now, let us consider a slightly different network, which is essentially the same as in Figure 3 but the first and the last layer are not connected. We call this a wire network and we show its communication graph in Figure 4. We are going to glue two of the wire networks, using the previous lemma, in order to create a larger wire. In Figure 5, it is represented the glueing operation between two of these wire networks. The glueing parts are highlighted inside a box in both gadgets. Dashed boxes indicate the zones that are not ruled by the dynamics (the X, Y , C_1 and C_2 in the latter lemma). Observe that the pseudo-orbit which is induces the pattern that goes through the layers from left to right in both gadgets is preserved in the new network. This pseudo-orbit is shown in Figure 4. More precisely, the sequence of configurations $(w_1^i)_{i=0,\dots,4}$ and w_0 are both X and Y pseudo-orbits in each of the wire networks that we are glueing. Observe that both of this sequences satisfy the conditions asked by the lemma and thus, they induce a $X \cup Y$-pseudo orbit on the glued network.

Observe that in the latter example, the glueing is straightforward not only because the structure of the graph is uniform but due to the fact that the local rule is CSAN. However, even in the case of a CSAN family where the transition rules are determined by a labeled non-directed graph, the result of a glueing operation has no reason to belong to the family because the symmetry of the interaction graph might be broken (see Figure 6). The following lemma gives a sufficient condition in graph theoretical terms for glueing within a concrete family of automata networks. Intuitively, it consists in asking that, in each graph, all the connections of one half of the dowel to the rest of the graph goes through the other half of the dowel. Here the wooden dowel metaphor is particularly relevant: when considering a single piece of wood with the dowel inserted inside, one half of the dowel is 'inside' (touches the piece of wood), the other half is 'outside' (not touching the piece of wood); then, when the two pieces are attached, each position in the wood assembly is locally either like in one piece of wood with the dowel inserted or like in the other one with the dowel inserted.

Lemma 9 (Glueing for CSAN)**.** *Let* (G_1, λ_1, ρ_1) *and* (G_2, λ_2, ρ_2) *be two CSAN from the same CSAN family* $\mathcal{F}$ *where*

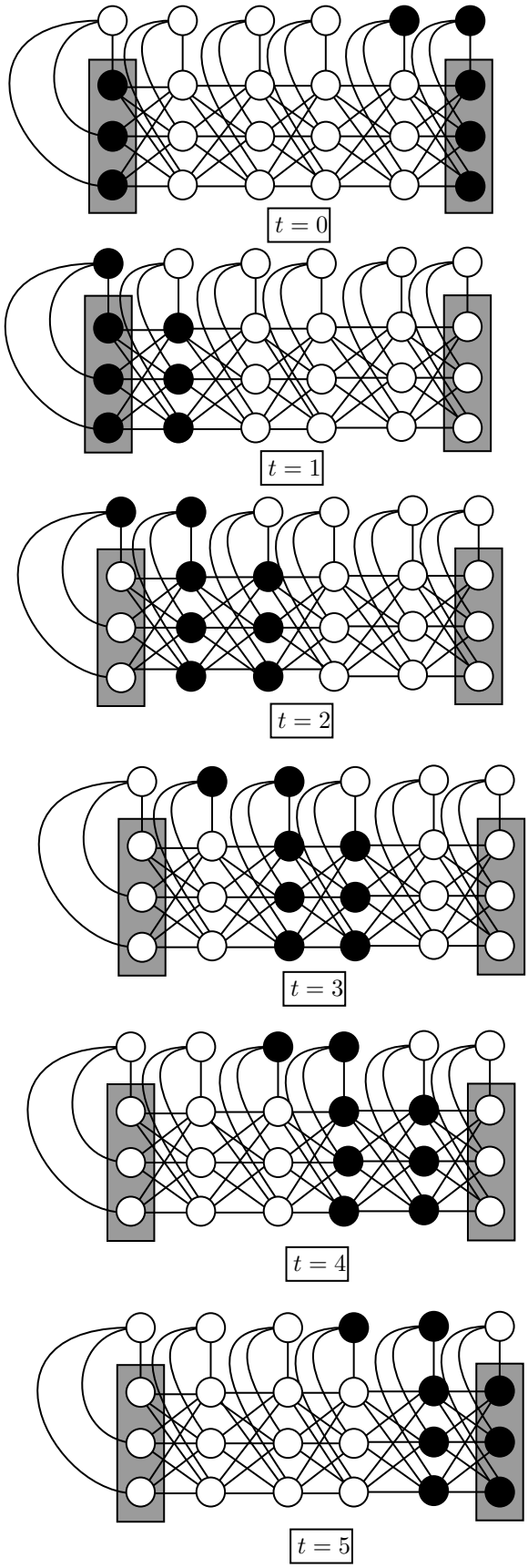


Figure 3: An orbit of a clock network. Gray-shaded nodes are connected. Nodes in state 1 are represented by black circles and those in state 0 are represented by white circles. The represented orbit is periodic of period 6.

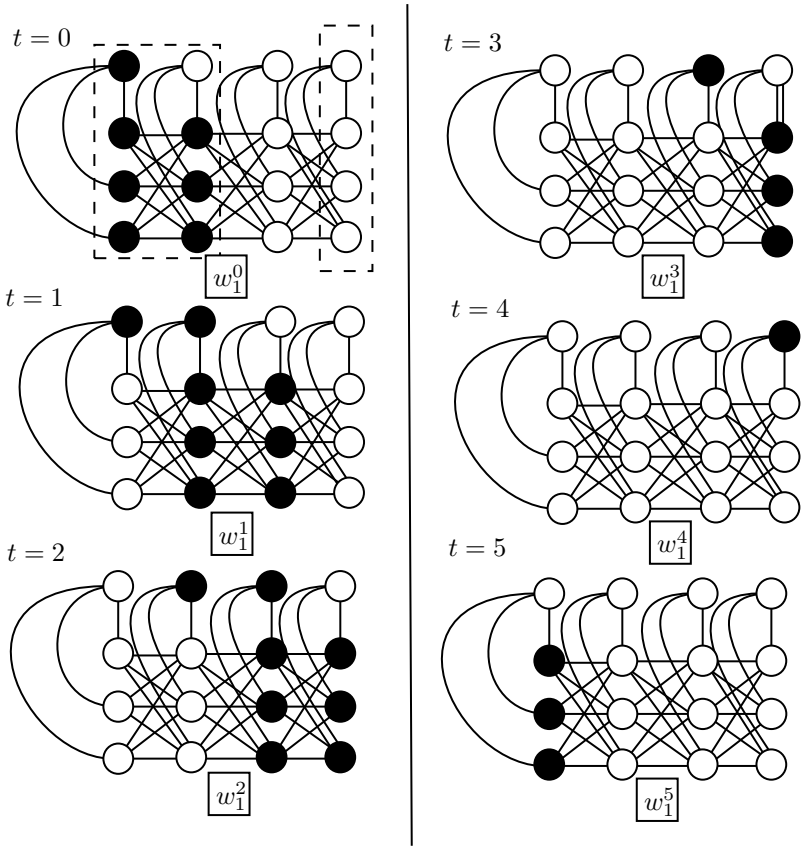


Figure 4: A pseudo-orbit representing a signal on a wire network. Black nodes are in state 1 and white nodes are in state 0. The configuration at each time step is represented by the notation $w_i^t, t = 0, 1, 2, 3, 4, 5 i = 0, 1$. This notation stands for the configuration in time t that codes a signal representing the bit i. Thus, the pseudo-orbit represented is $(w_1^0, w_1^1, w_1^2, w_1^3, w_1^4, , w_1^5)$. In time $t = 0$, it is marked in dotted boxes the parts of the networks that do not depend on the local rule of the network.

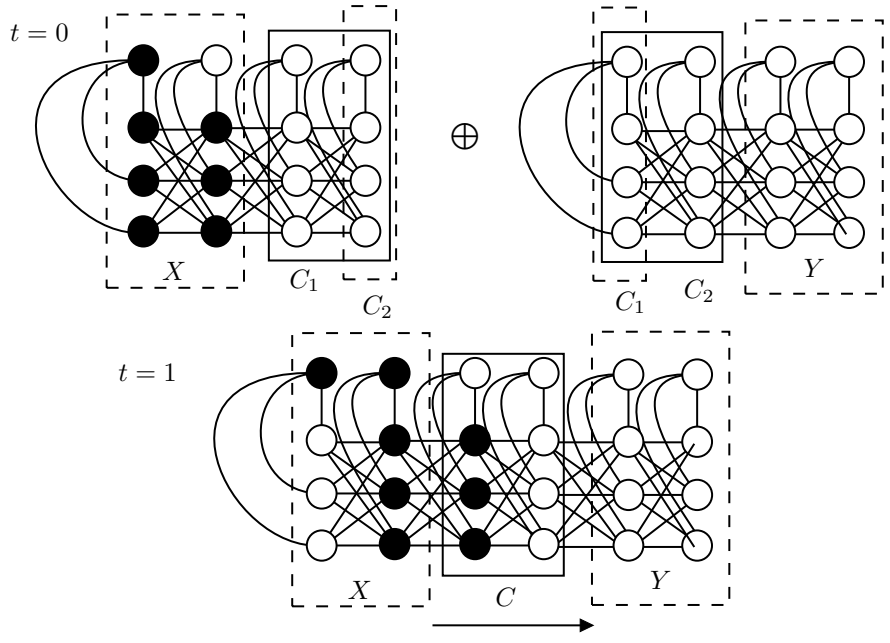


Figure 5: Example of two wire gadgets transmitting a 1 signal represented as a particular pseudo-orbit. Nodes in black are in state 1 and nodes in white are in state 0. Nodes inside dashed boxes represent the zones that are arbitrary for the pseudo-orbit. At time $t = 0$ it is shown a $X \cup C_2$-pseudo-orbit for the left gadget and in the right a $Y \cup C_1$-pseudo-orbit. At time $t = 1$ it is shown the result of the glueing of the two gadgets and the resulting $X \cup Y$-pseudo-orbit given by the theorem.

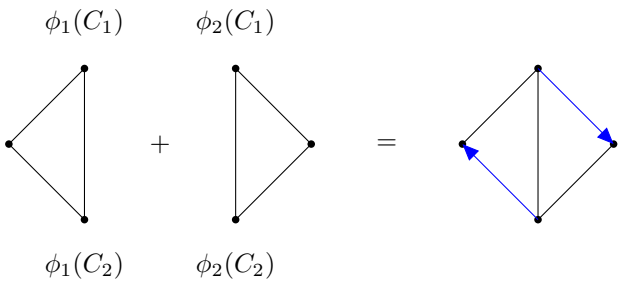


Figure 6: Symmetry breaking in interaction graph after a glueing operation. Arrows indicate influence of a node (source) on another (target), edges without arrow indicates bi-directional influence. Here C consists in two nodes only.

G_1 and G_2 are disjoint and F_1 and F_2 are the associated global maps. Taking again the notations of Definition 7, if the following conditions hold

- *the labeled graphs induced by $\varphi_1(C)$ and $\varphi_2(C)$ in G_1 and G_2 are the same (using the identification $\varphi_1(v) = \varphi_2(v)$)*
- $N_{G_1}(\varphi_1(C_2)) \subseteq \varphi_1(C)$
- $N_{G_2}(\varphi_2(C_1)) \subseteq \varphi_2(C)$

then the glueing $F' = F_1 \overset{\phi_1}{C_1} \oplus \overset{\phi_2}{C_2} F_2$ can be defined as the CSAN on graph $G' = (V', E')$ where V' is as in Definition 7 and each node $v \in V'$ has the same label and same labeled neighborhood as either a node of (G_1, λ_1, ρ_1) or a node of (G_2, λ_2, ρ_2). In particular F' belongs to $\mathcal{F}$.

Proof. Let us define $\beta_i : V_i \to V'$ by

$$\beta_i(v) = \begin{cases} \phi_i^{-1}(v) & \text{if } v \in \phi_i(C), \\ v & \text{else.} \end{cases}$$

Fix $i = 1$ or 2. According to Definition 7, if $v \in V'$ is such that $\alpha(v) \in V_i$ then $F'_v = (F_i)_{\alpha(v)} \circ \rho_i$. By definition of CSAN, this means that for any $x \in Q^{V'}$ we have $F'_v(x) = \psi_{i,\alpha(v)}(x_{|\beta(N_{G_i}(\alpha(v)))})$ where $\psi_{i,\alpha(v)}$ is a map depending only on the labeled neighborhood of $\alpha(v)$ in G_i as in Definition 2. So the dependencies of v in F' are in one-to-one correspondence through β with the neighborhood of $\alpha(v)$ in G_i. They key observation is that the symmetry of dependencies is preserved, formally for any $v' \in \beta(N_{G_i}(\alpha(v)))$:

- either $\alpha(v') \in V_i$ in which case the dependency of v' on v (in map $\psi_{i,\alpha(v')}$) is the same as the dependency of v on v' (in map $\psi_{i,\alpha(v)}$), and both are determined by the undirected labeled edge $\{\alpha(v), \alpha(v')\}$ of G_i;
- or $\alpha(v') \notin V_i$ and in this case necessarily $v \in C_i$ and $v' \in C_{3-i}$ (because $N_{G_i}(V_i \setminus \varphi_i(C)) \cap \varphi_i(C) \subseteq \varphi_i(C_i)$ from the hypothesis), so the dependency of v' on v is the same as the dependency of v on v' because the labeled graphs induced by $\phi_1(C)$ and $\phi_2(C)$ in G_1 and G_2 are the same.

Concretely, F' is a CSAN that can be defined on graph $G' = (V', E'_1 \cup E'_2 \cup E(C))$ with

$$E'_i = E(V_i \setminus \varphi_i(C)) \cup \{(u, v_i) : u \in V(C_i), v_i \in (V_i \setminus \varphi_i(C)), (\varphi_i(u), v_i) \in E_i\},$$

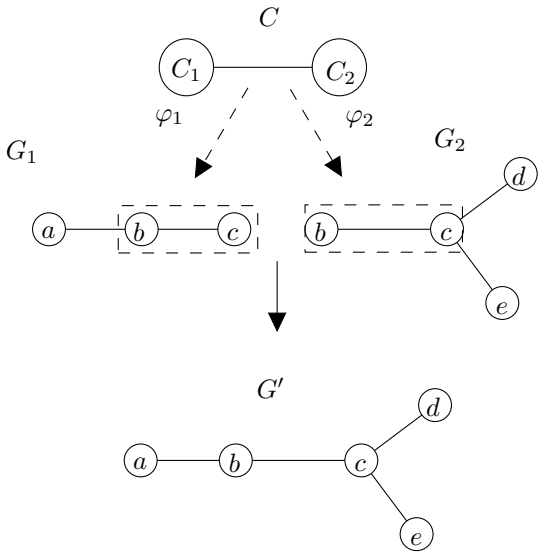


Figure 7: Example of a glueing of two compatibles CSAN. The labeling in nodes of G_1, G_2 and G' shows equalities between local λ maps of these three CSAN.

and labels as follows:

- on $E(C)$ as in both G_1 and G_2 (which agree through maps ϕ_1 and ϕ_2 on C),
- on $E(V_i \setminus \varphi_i(C))$ as in G_i,
- for each $u \in V(C_i), v_i \in (V_i \setminus \varphi_i(C))$ such that $(\varphi_i(u), v_i) \in E_i$, edge (u, v_i) has same label as $(\varphi_i(u), v_i)$.

Since any CSAN families (Definition 2) is entirely based on local constraints on labels (vertex label plus set of labels of the incident edges), we deduce that F' is in $\mathcal{F}$. □

3.3. $\mathcal{G}$-gadgets, gadget glueing and simulation of $\mathcal{G}$-networks

We now give a precise meaning to the intuitively simple fact that, if a family of automata networks can coherently simulate a set of small building blocks (gates from $\mathcal{G}$), it should be able to simulate any automata network that can be built out of them ($\mathcal{G}$-networks).

The key idea here is that gates from $\mathcal{G}$ will be represented by networks of the family called $\mathcal{G}$-gadgets, and the wiring between gates to obtain a $\mathcal{G}$-network will translate into glueing between $\mathcal{G}$-gadgets. Following this idea there are two main conditions for the family to simulate any $\mathcal{G}$-network:

- the glueing of gadgets should be freely composable inside the family to allow the building of any $\mathcal{G}$-network;
- the gadgets corresponding to gates from $\mathcal{G}$ should correctly and coherently simulate the functional relation between inputs and outputs given by their corresponding gate.

For clarity, we separate these conditions in two definitions.

We start by developing a definition for gadget glueing. Recall first that Definition 7 relies on the identification of a common dowel in the two networks to be glued. Here, as we want to mimic the wiring of gates which connects inputs to outputs, several copies of a fixed network called *glueing interface* will be identified in each gadget, some of them corresponding to input, and the other ones to outputs. In this context, the only glueing operations we will use are those where some output copies of the interface in a gadget A are glued on input copies of the interface in a gadget B and some input copies of the interface in A are glued on output copies of the interface in B. Then, the global dowel used to formally apply Definition 7 is a disjoint union of the selected input/output copies of the interface. Figure 8 illustrates with the notations of the following Definition.

Definition 10 (Glueing interface and gadgets)**.** *Let $C = C_i \cup C_o$ be a fixed set partitioned into two sets. A* gadget *with glueing interface $C = C_i \cup C_o$ is an automata network $F : Q^{V_F} \to Q^{V_F}$ together with two collections of injective maps $\phi^i_{F,k} : C \to V_F$ for $k \in I(F)$ and $\phi^o_{F,k} : C \to V_F$ for $k \in O(F)$ whose images in V_F are pairwise disjoint and where $I(F)$ and $O(F)$ are disjoint sets which might be empty.*

Given two disjoint gadgets $(F, (\phi^i_{F,k}), (\phi^o_{F,k}))$ and $(G, (\phi^i_{G,k}), (\phi^o_{G,k}))$ with same alphabet and interface $C = C_i \cup C_o$, a gadget glueing *is a glueing of the form $H = F \overset{\phi_F}{\underset{C_F}{\oplus}} \overset{\phi_G}{\underset{C_G}{}} G$ defined as follows:*

- *a choice of a set A of inputs from F and outputs from G given by injective maps $\sigma_F : A \to I(F)$ and $\sigma_G : A \to O(G)$,*
- *a choice of a set B of outputs from F and inputs from G given by injective maps $\tau_F : B \to O(F)$ and $\tau_G : B \to I(G)$ (the set B is disjoint from A),*
- *C_F is a disjoint union of $|A|$ copies of C_i, and $|B|$ copies of C_o: $C_F = A \times C_i \cup B \times C_o$,*
- *C_G is a disjoint union of $|A|$ copies of C_o, and $|B|$ copies of C_i: $C_G = A \times C_o \cup B \times C_i$,*
- *$\phi_F : C_F \cup C_G \to V_F$ is such that $\phi_F(a, c) = \phi^i_{F,\sigma_F(a)}(c)$ for $a \in A$ and $c \in C$, and $\phi_F(b, c) = \phi^o_{F,\tau_F(b)}(c)$ for $b \in B$ and $c \in C$,*
- *$\phi_G : C_F \cup C_G \to V_G$ is such that $\phi_G(a, c) = \phi^o_{G,\sigma_G(a)}(c)$ for $a \in A$ and $c \in C$, and $\phi_G(b, c) = \phi^i_{G,\tau_G(b)}(c)$ for $b \in B$ and $c \in C$.*

The resulting network H is a gadget with same alphabet and same interface with $I(H) = I(F) \setminus \sigma_F(A) \cup I(G) \setminus \tau_G(B)$ and $O(H) = O(F) \setminus \tau_F(B) \cup O(F) \setminus \sigma_G(A)$ and $\phi^i_{H,k}$ is $\phi^i_{F,k}$ when $k \in I(F)$ and $\phi^i_{G,k}$ when $k \in I(G)$, and $\phi^o_{H,k}$ is $\phi^o_{F,k}$ when $k \in O(F)$ and $\phi^o_{G,k}$ when $k \in O(G)$.

Given a set of gadgets X with same alphabet and interface, its closure by gadget glueing is the closure of X by the following operations:

- *add a disjoint copy of some gadget from the current set,*
- *add the disjoint union of two gadgets from the current set,*
- *add a gadget glueing of two gadgets from the current set.*

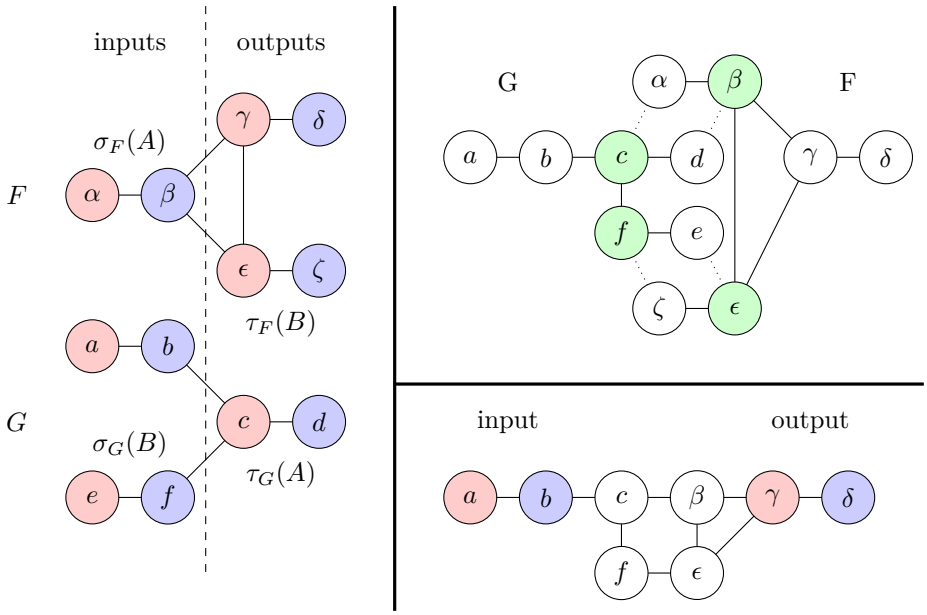


Figure 8: Gadget glueing as in Definition 10. On the left, two gadgets with interface $C = C_i \cup C_o$ where C_i part in each copy of the interface dowel is in red and C_o part in blue. The gadget glueing is done with input $\sigma_F(A)$ on output $\tau_G(A)$ (here A is a singleton) and output $\tau_F(B)$ on input $\tau_G(B)$ (B is also a singleton). On the upper right, a representation of the global glueing process where nodes in green are those in the copy of C_F in F or in the copy C_G in G; dotted links show the bijection between the embeddings of $C = C_F \cup C_G$ into V_F and V_G via maps ϕ_F and ϕ_G. On the lower right the resulting gadget with the same interface $C = C_i \cup C_o$ as the two initial gadgets.

Remark 2. *The representation of the result of a gadget glueing can be easily computed from the two gadgets F and G and the choices of inputs/outputs given by maps σ_F, σ_G, τ_F and τ_G: precisely, by definition of glueing (Definition 7) the local map of each node of the result automata network is either a local map of F (when in $V_F \setminus \phi_F(C_G)$ or in C_F) or a local map of G (when in $V_G \setminus \phi_G(C_F)$ or in C_G). Note also that the closure by gadget glueing of a finite set of gadgets X is always a set of automata networks of bounded degree.*

Lemma 9 gives sufficient conditions on a set of gadgets to have its closure by gadget glueing contained in a CSAN family.

Lemma 11. *Fix some alphabet Q and some glueing interface $C = C_i \cup C_o$ and some CSAN family $\mathcal{F}$. Let (G_n, λ_n, ρ_n) for $n \in S$ be a set of CSAN belonging to $\mathcal{F}$ with associated global maps F_n. Let $\phi^i_{F_n,k}$ for $k \in I(F_n)$ and $\phi^o_{F_n,k}$ for $k \in O(F_n)$ be maps as in Definition 10 so that $(F_n, (\phi^i_{F_n,k}), (\phi^o_{F_n,k}))$ is a gadget with interface $C = C_i \cup C_o$. Denote by X the set of such gadgets. If the following conditions hold:*

- *the labeled graphs induced by $\phi^i_{F_n,k}(C)$ and by $\phi^o_{F_n,k}(C)$ in G_n are all the same for all n and k with the identification of vertices given by the $\phi^*_{*,*}$ maps,*
- *$N_{G_n}(\phi^i_{F_n,k}(C_o)) \subseteq \phi^i_{F_n,k}(C)$ for all $n \in S$ and all $k \in I(F_n)$,*
- *$N_{G_n}(\phi^o_{F_n,k}(C_i)) \subseteq \phi^o_{F_n,k}(C)$ for all $n \in S$ and all $k \in O(F_n)$,*

then the closure by gadget glueing of X is included in $\mathcal{F}$.

Proof. Consider first the gadget glueing H of two gadgets F_n and $F_{n'}$ from X. Following Definition 10, the global dowel $C_{F_n} \cup C_{F_{n'}}$ used in such a glueing is a disjoint union of copies of C, and its embedding ϕ_{F_n} in G_n (resp. $\phi_{F_{n'}}$ in $G_{n'}$) is a disjoint union of maps $\phi^*_{F_n,*}$ (resp. $\phi^*_{F_{n'},*}$). Therefore the three conditions of Lemma 9 follow from the three conditions of the hypothesis on gadgets from X and we deduce that H belongs to family $\mathcal{F}$. Moreover, it is clear that gadget H then also verifies the three conditions from the hypothesis, and adding a copy of any gadget to the set also verifies the conditions. We deduce that the closure by gadget glueing of X is included in $\mathcal{F}$.

□

The second key aspect to have a coherent set X of $\mathcal{G}$-gadgets is of dynamical nature: there must exists a collection of pseudo-orbits on each gadget satisfying suitable conditions to permit application of Lemma 8 for any gadget glueing in the closure of X; moreover, these pseudo-orbits must simulate via an appropriate coding the input/output relations of each gate $g \in \mathcal{G}$ in the corresponding gadget. To obtain this, we rely on a standard set of traces on the glueing interface that must be respected on any copy of it in any gadget.

Definition 12 (Coherent $\mathcal{G}$-gadgets)**.** *Let $\mathcal{G}$ be any set of finite maps over alphabet Q and let $\mathcal{F}$ be any set of abstract automata networks over alphabet $Q_{\mathcal{F}}$. We say $\mathcal{F}$ has* coherent $\mathcal{G}$-gadgets *if there exists:*

- *a unique glueing interface $C = C_i \cup C_o$,*
- *a set X of gadgets $(F_g, (\phi^i_{g,k})_{1\leq k\leq i(g)}, (\phi^o_{g,k})_{1\leq k\leq o(g)})$ for each $g \in \mathcal{G}$ where $F_g : Q_{\mathcal{F}}^{V_g} \to Q_{\mathcal{F}}^{V_g} \in \mathcal{F}$ and sets V_g and C are pairwise disjoint, and the closure of X by gadget glueing is contained in $\mathcal{F}$,*
- *a* state configuration *$s_q \in Q_{\mathcal{F}}^C$ for each $q \in Q$ such that $q \mapsto s_q$ is an injective map,*
- *a* context configuration *$c_g \in Q_{\mathcal{F}}^{\hat{V}_g}$ for each $g \in \mathcal{G}$ where $\hat{V}_g = V_g \setminus \big(\cup_k \phi^i_{g,k}(C) \cup_k \phi^o_{g,k}(C)\big)$,*
- *a time constant T,*
- *a* standard trace *$\tau_{q,q'} \in (Q_{\mathcal{F}}^C)^{\{0,\dots,T\}}$ for each pair $q, q' \in Q$ such that $\tau_{q,q'}(0) = s_q$ and $\tau_{q,q'}(T) = s_{q'}$,*
- *for each $g \in \mathcal{G}$ and for any uples of states $q_{i,1}, \dots, q_{i,i(g)} \in Q$ and $q_{o,1}, \dots, q_{o,o(g)} \in Q$ and $q'_{i,1}, \dots, q'_{i,i(g)} \in Q$ and $q'_{o,1}, \dots, q'_{o,o(g)} \in Q$ such that $g(q_{i,1}, \dots, q_{i,i(g)}) = (q'_{o,1}, \dots, q'_{o,o(g)})$, a P_g-pseudo-orbit $(x^t)_{0\leq t\leq T}$ of F_g with $P_g = \bigcup_{1\leq k\leq i(g)} \phi^i_{g,k}(C_o) \cup \bigcup_{1\leq k\leq o(g)} \phi^o_{g,k}(C_i)$ and with*
  - *for each $1 \leq k \leq i(g)$, the trace $t \mapsto x^t_{\phi^i_{g,k}(C)}$ is exactly $\tau_{q_{i,k},q'_{i,k}}$,*
  - *for each $1 \leq k \leq o(g)$, the trace $t \mapsto x^t_{\phi^o_{g,k}(C)}$ is exactly $\tau_{q_{o,k},q'_{o,k}}$,*
  - *$x^0_{\hat{V}_g} = x^T_{\hat{V}_g} = c_g$.*

We can now state the key lemma of our framework: having coherent $\mathcal{G}$-gadgets is sufficient to simulate the whole family of $\mathcal{G}$-networks.

Lemma 13. *Let $\mathcal{G}$ be a set of irreducible gates. If an abstract automata network family $\mathcal{F}$ has coherent $\mathcal{G}$-gadgets then it contains a subfamily of bounded degree networks with the canonical bounded degree representation $(\mathcal{F}_0, \mathcal{F}_0^*)$ that simulates $\Gamma(\mathcal{G})$ in time T and space S where T is a constant map and S is bounded by a linear map.*

Proof. We take the notations of Definition 12. To any $\mathcal{G}$-network F with set of nodes V given as in Definition 6 by a list of gates $g_1, \dots, g_k \in \mathcal{G}$ and maps α and β (see Remark 1) we associate an automata network from $\mathcal{F}$ as follows. First, let $(F_{g_i})_{1 \leq i \leq k}$ be the gadgets corresponding to gates g_i and suppose they are all disjoint (by taking disjoint copies when necessary). Then, start from the gadget $F_1 = F_{g_1}$ and for any $1 \leq i < k$ we define F_{i+1} as the gadget glueing of F_i and $F_{g_{i+1}}$ on the input/outputs as prescribed by maps α and β. More precisely, the gadget glueing select the set of inputs (j,k) with $1 \leq j \leq i$ and $1 \leq k \leq i(g_j)$ such that $\beta(\alpha(j,k)) = (i+1,k')$ for some $1 \leq k' \leq o(g_{i+1})$ and glue them on their corresponding output $(i+1,k')$ of g_{i+1} (precisely, through maps σ_{F_i} and $\sigma_{F_{g_{i+1}}}$ of domain A_{i+1} playing the role of maps σ_F and σ_G of Definition 10), and, symmetrically, selects the inputs $(i+1,k)$ with $1 \leq k \leq i(g_{i+1})$ such that $\beta(\alpha(i+1,k)) = (j,k')$ for some $1 \leq j \leq i$ and $1 \leq k' \leq o(g_j)$ and glue their corresponding output (j,k') (precisely, through maps $\tau_{F_{g_{i+1}}}$ and τ_{F_i} of domain B_{i+1} playing the role of maps τ_G and τ_F from Definition 10). If both of these sets of inputs/outputs are empty, the gadget glueing is replaced by a simple disjoint union.

The final gadget F_k has no input and no output, and a representation of it as a pair graph and local maps can be constructed in **DLOGSPACE**, because the local map of each of its nodes is independent of the glueing sequence above and completely determined by the gadget F_{g_j} it belongs to and whether the node is inside some input or some output dowel or not (see Reamrk 2).

It now remains to show that the automata network F_k simulates F. To fix notations, let V_k be the set of nodes of F_k. For each $v \in V$, define $D_v \subseteq V_k$ as the copy of the dowel that correspond to node v of F, *i.e.* that was produced in the gadget glueing of F_i with $F_{g_{i+1}}$ for i such that $\beta(v) = (i+1,k')$ for some $1 \leq k' \leq o(g_{i+1})$ (or symmetrically $\alpha(i+1,k) = v$ for some $1 \leq k \leq i(g_j)$). More precisely, if $a \in A_{i+1}$ is such that $\sigma_{F_{g_{i+1}}}(a) = (i+1,k')$ then $D_v = \{a\} \times C$ (symmetrically if $b \in B_{i+1}$ is such that $\tau_{F_{g_{i+1}}}(b) = (i+1,k)$ then $D_v = \{b\} \times C$). Also denote by $\rho_v : D_v \to C$ the map such that $\rho_v(a,c) = c$ for all $c \in C$ (symmetrically, $\rho_v(b,c) = c$). With these notations, we have

$$V_k = \bigcup_{v \in V} D_v \cup \bigcup_{1 \leq i \leq k} \hat{V}_{g_i}$$

Let us define the block embedding $\phi : Q^V \to Q_{\mathcal{F}}^{V_k}$ as follows

$$\phi(x)(v') = \begin{cases} s_{x_v}(\rho_v(v')) & \text{if } v' \in D_v, \\ c_{g_i}(v') & \text{if } v' \in \hat{V}_{g_i}. \end{cases}$$

for any $x \in Q^V$ and any $v' \in V_k$, where s_q for $q \in Q$ are the state configurations and c_g for $g \in \mathcal{G}$ are the context configurations granted by Definition 12. Note that ϕ is injective because the map $q \mapsto s_q$ is injective. By inductive applications of Lemma 8, the P_{g_i}-pseudo-orbits of each F_{g_i} from Definition 12 can be glued together to form valid orbits of F_k that start from any configuration $\phi(x)$ with $x \in Q^V$ and ends after T steps in a configuration $\phi(y)$ for some $y \in Q^V$ which verifies $y = F(x)$. Said differently, we have the following equality on Q^V:

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

Note that T is a constant and that the size of V_k is at most linear in the size of V. The lemma follows. □

Remark 3. *Note that in Lemma 13 above, the block embedding that is constructed can be viewed as a collection of blocks of bounded size that encode all the information plus a context (see [19, Remark 2]).*

As it is stated in the first paper of this series (see [19, Remark 1]), in the case of CSAN families, there exists an efficient algorithm to go from the bounded degree representation to the CSAN representation (as a consequence of the fact that the representations use a constant amount of memory). Thus, we provide hereunder a simpler formulation of the Lemma.

Corollary 1. *If $\mathcal{G}$ is a set of irreducible gates and $\mathcal{F}$ a CSAN family which has coherent $\mathcal{G}$-gadgets then $\mathcal{F}$ simulates $\Gamma(\mathcal{G})$ in time T and space S where T is a constant map and S is bounded by a linear map.*

3.3.1. Game of life has coherent gadgets

In this section, we are going to show that *Game of life* has coherent $\mathcal{G}_{\text{NOR}}$ gadgets, where $\mathcal{G}_{\text{NOR}} = \{\text{NOR}(x, y) = (\overline{x \vee y}, \overline{x \vee y})\}$. In order to do that, we are going to show that we are able to simulate this two gates by combining wire networks and a clock network.

First, we show, in Figure 10, the structure of the communication graph of the NOR gadget. Observe that it is composed by 2 copies of the wire network and 2 copies of the clock network (see Figure 9). Now, we present the main result of this subsection:

Lemma 14. *Game of Life automata networks admits coherent $\mathcal{G}_{NOR}$ gadgets.*

Proof. We are going to show that the NOR gadget satisfies the conditions of the Definition 12. In fact we have that:

- The NOR gadget has a unique glueing interface C which is shown in Figure 10. The functions ϕ^i, ϕ^o are also represented in the same figure.
- Observe that the map $q \in \{0, 1\} \mapsto s_q = (w_0^q)|_C$ (observe that $w_0^i = w_0$ for each i since all the nodes are in state 0 in this configuration), is injective.
- The context configuration is given by:
  - For the clock network we use the initial condition of the dynamics of the clock network (see Figure 3).
  - For the copies of the wire network: the nodes are in state 0 with the exception of the nodes in the copies of C (see Figure 4).
- The time constant is $T = 6$
- The configurations $w_q^0, \ldots, w_q^5$ (see Figure 4) and w_0^t (or simply w_0) where w_0^t is the configuration in which each node is in state 0 for every $t = 0, \ldots, T$ define a standard trace for each pair $(0, 0), (0, 1), (1, 0), (1, 1)$ as follows: $\tau_{q,q'} = ((w_q^0)|_C, (w_q^1)|_C, (w_q^3)|_C, (w_q^4)|_C, (w_{q'}^5)|_C, (w_{q'^)}^6|_C = (w_{q'^)}^0|_C))$.
- The pseudo orbits are shown in Table 1 and Figure 9. In the table, the detail of the local computation produced by the central part of the gadget is given. Figure 9 shows how the nodes are labeled in the previous table. In addition, the latter figure shows a general picture on how the signals are transmitted and computed by the gadget. In particular, it is posible to verify that all the computation is produced in $T = 6$.

Nodes/Time	l_1	l_2	l_3	l'_1	l'_2	l'_3	c_1	c_2	c_3	a	v	r_1	r_2	r_3	r'_1	r'_2	r'_3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	x	x	x	y	y	y	1	1	1	0	0	0	0	0	0	0	0
3	x	x	x	y	y	y	1	1	1	0	$\overline{x \vee y}$	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	$\overline{x \vee y}$	$\overline{x \vee y}$	$\overline{x \vee y}$	$\overline{x \vee y}$	$\overline{x \vee y}$	$\overline{x \vee y}$	$\overline{x \vee y}$	$\overline{x \vee y}$
5	0	0	0	0	0	0	0	0	0	$\overline{x \vee y}$	0	$\overline{x \vee y}$	$\overline{x \vee y}$	$\overline{x \vee y}$	$\overline{x \vee y}$	$\overline{x \vee y}$	$\overline{x \vee y}$
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1: Pseudo-orbit associated to the NOR gadget implemented using Game of Life rules. The labelling is defined according to Figure 11

- After $T = 6$ time steps, the nodes that are not part of any copy of C will return to the state given by the context configuration.

We conclude that the NOR gadget satisfies the conditions given by Definition 12. Thus, the lemma holds. □

3.4. $\mathcal{G}_m$-networks and $\mathcal{G}_{m,2}$-networks as standard universal families

Let $i, o \in \{1, 2\}$ be two numbers. We define the functions $\mathrm{OR}, \mathrm{AND} : \{0,1\}^i \to \{0,1\}^o$ where $\mathrm{OR}(x) = \max(x)$ and $\mathrm{AND}(x) = \min(x)$. Note that in the case in which $i = o = 1$ we have $\mathrm{AND}(x) = \mathrm{OR}(x) = \mathrm{Id}(x) = x$ and also in the case $i = 1$ and $o = 2$ we have that $\mathrm{AND}(x) = \mathrm{OR}(x) = (x, x)$. We define the set $\mathcal{G}_m = \{\mathrm{AND}, \mathrm{OR}\}$. Observe that in this case o and i may take different values. In addition, we define the set $\mathcal{G}_{m,2}$ in which we fix $i = o = 2$.

It is folklore knowledge that monotone Boolean networks (with AND/OR local maps) can simulate any other network. Here we make this statement precise within our formalism: $\mathcal{G}_m$-networks are strongly universal. Note that there is more work than the classical circuit transformations involving monotone gates because we need to obtain a simulation of any automata network via block embedding. In particular we need to build monotone circuitry that is synchronized and reusable (*i.e.* that can be reinitialized to a standard state before starting a computation on a new input). Moreover, our definitions requires a production of $\mathcal{G}_m$-networks in **DLOGSPACE**. The main ingredient for establishing universality of $\mathcal{G}_m$-networks is an efficient circuit transformation due to Greenlaw, Hoover and Ruzzo in [21, Theorems 6.2.3 to 6.2.5]. Let us start by proving that this family is strongly universal, which is slightly simpler to prove.

Theorem 15. *The family $\Gamma(\mathcal{G}_m)$ of all $\mathcal{G}_m$-networks is strongly universal.*

Proof. Let Q an arbitrary alphabet and $F : Q^n \to Q^n$ an arbitrary automata network on alphabet Q such that the communication graph of F has maximum degree Δ. Let $C : \{0,1\}^n \to \{0,1\}^n$ be a constant depth circuit representing F. Let us assume that C has only OR, AND and NOT gates. We can also assume that C is synchronous because, as its depth does not depend on the size of the circuit, one can always add fanin one and fanout one OR gates in order to modify layer structure. We are going to use a very similar transformation to the one proposed in [21, Theorem 6.2.3] in order to efficiently construct an automata network in $\Gamma(\mathcal{G}_m)$. In fact, we are going to duplicate the original circuit by considering the coding $x \in \{0,1\} \to (x, 1-x) \in \{0,1\}^2$. Roughly, each gate will have a positive part (which is essentially a copy) and a negative part which is produces the negation of the original output by using De Morgan's laws. More precisely, we are going to replace each gate in the network by the gadgets shown in Figure 12.

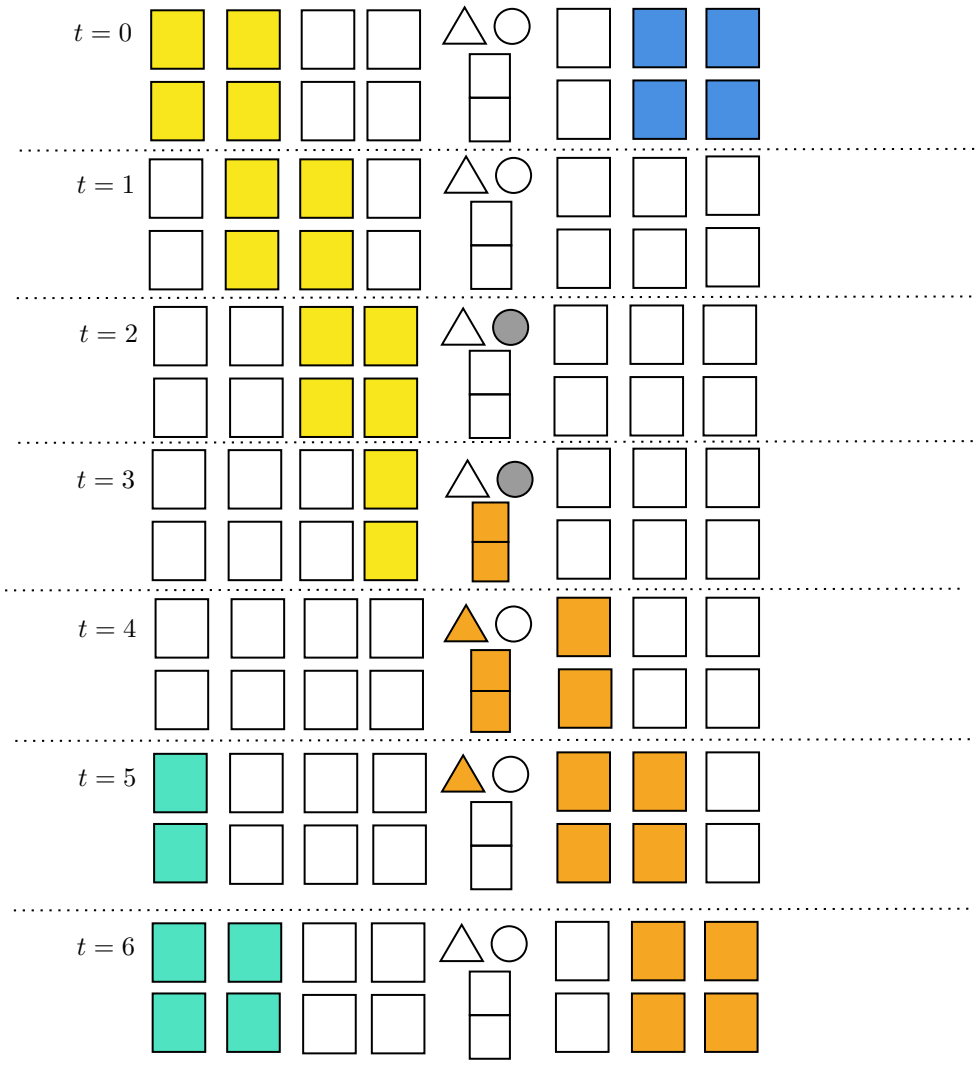


Figure 9: Representation of the pseudo-orbits used for the computation of the NOR gadget implemented using Game of Life automata networks.

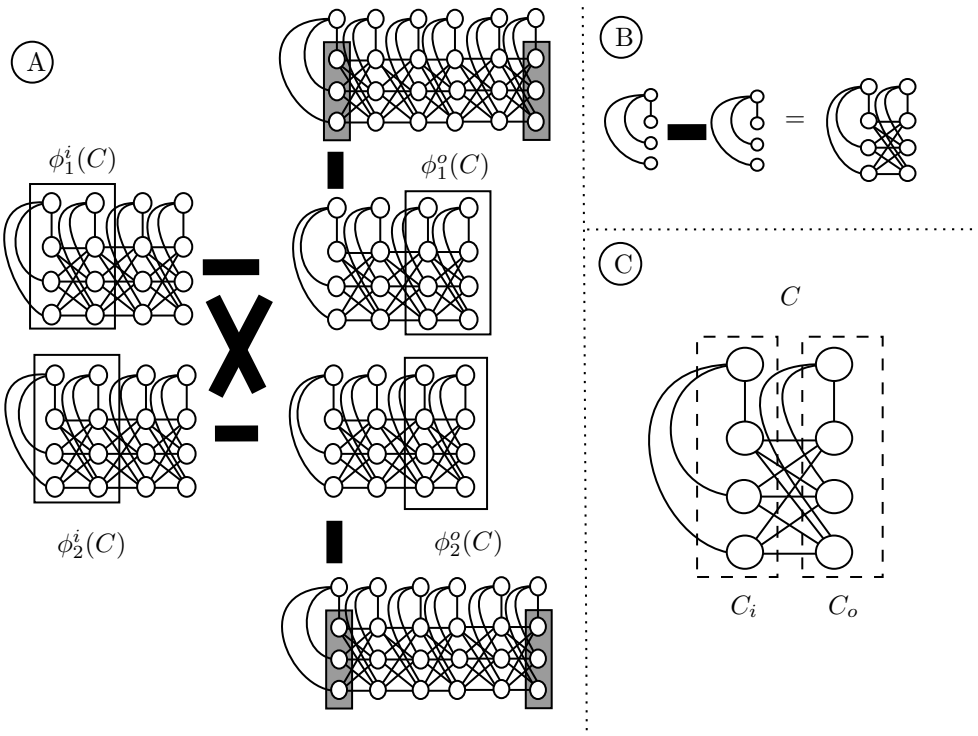


Figure 10: (A) Representation of the computation part of the NOR gadget implemented using Game of Life rules. (B) Thick lines represent complete connections between the different layers, i.e. each node in the central layer is connected with any node in the left layer. (C) Representation of the glueing interface.

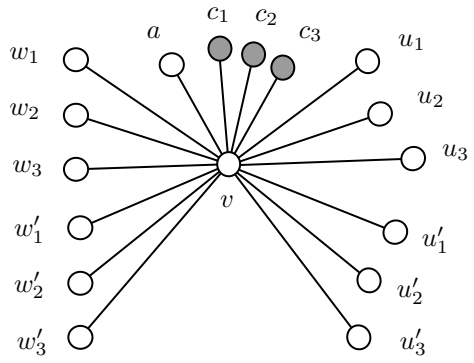


Figure 11: Structure of the connections of the central nodes in the NOR gadget implemented using Game of Life rules. Nodes labeled as l (resp. r) are nodes inside the last part (resp. first) of a wire network, nodes labeled as c are nodes inside one layer of a clock network (see Figure 4).

The main idea is that one can represent the function $x \wedge y$ by the coding: $(x \wedge y, \overline{x} \vee \overline{y})$ and $x \vee y$ by the coding: $(x \vee y, \overline{x} \wedge \overline{y})$. In addition, each time there is a NOT gate, we replace it by a fan in 1 fan out 1 OR gadget and we connect positive outputs to negative inputs in the next layer and negative outputs to positive inputs as it is shown in Figure 13. We are going to call C^* to the circuit constructed by latter transformations. Observe that C^* is such that it holds on $\{0,1\}^i$:

$$\phi \circ C = C^* \circ \phi$$

where $\phi : \{0,1\}^n \to \{0,1\}^{2n}$ is defined for any n by $\phi(x)_{2j} = x_j$ and $\phi(x)_{2j+1} = \neg x_j$.

Now consider the coding map $m_Q : Q \to \{0,1\}^k$ and let $n = k|V|$. Build from C^* the $\mathcal{G}_{m,2}$-network $F^* : \{0,1\}^{V^+} \to \{0,1\}^{V^+}$ that correspond to it (gate by gate) and where the output j is wired to input j for all $1 \leq j \leq 2n$. Define a block embedding of Q^V into $\{0,1\}^{V^+}$ as follows (see [19, Remark 2] for more details on simulation):

- for each $v \in V$ let D_v be the set of input nodes in F^* that code v (via m_Q and then double railed logic),
- let $C = V^+ \setminus \bigcup_v D_v$ be the remaining context block,
- let $p_{v,q} \in \{0,1\}^{D_v}$ be the pattern coding node v in state q,
- let $p_C = 0^C$ be the context pattern,
- let $\phi : Q^V \to \{0,1\}^{V^+}$ be the associated block embedding map.

We claim that F^* simulates F via block embedding ϕ with time constant equal to the depth of C^* plus 1. Indeed, F^* can be seen as a directed cycle of N layers where layer $L_{i+1 \bmod N}$ only depends on layer i. The block embedding is such that for any configuration $x \in Q^V$, $\phi(x)$ is 0 on each layer except the layer containing the inputs. On configurations where a single layer L_i is non-zero, F^* will produce a configuration where the only non-zero layer is $L_{i+1 \bmod N}$. From there, it follows by construction of F^* that $\phi \circ F(x) = (F^*)^N \circ \phi(x)$ for all $x \in Q^V$.

The fact that construction is obtainable in **DLOGSPACE** follows from the same reasoning used to show in [21, Theorem 6.2.3]. In fact, authors show that reduction is actually better as they show it is $\mathbf{NC}^1$. □

Theorem 16. *The family $\Gamma(\mathcal{G}_{m,2})$ simulates in constant time and linear space the family $\Gamma(\mathcal{G}_m)$, i.e. there exists a constant function $T : \mathbb{N} \to \mathbb{N}$ and a linear function $S : \mathbb{N} \to \mathbb{N}$ such that $\Gamma(\mathcal{G}_m) \preccurlyeq_S^T \Gamma(\mathcal{G}_{m,2})$*

Proof. Let $F : Q^V \to Q^V$ be an arbitrary $\mathcal{G}_m$-network coded by its standard representation defined by a list of gates $g_1, \ldots, g_n$ and two functions α an β mapping inputs to nodes in F and nodes in F to outputs respectively. We are going to construct in **DLOGSPACE** a $\mathcal{G}_{m,2}$-network G that simulates F in time $T = \mathcal{O}(1)$ and space $S = \mathcal{O}(|V|)$ where H is the communication graph of F. In order to do that, we are going to replace each gate g_k by a small gadget. More precisely, we are going to introduce the following coding function: $x \in \{0,1\} \to (x,x,0) \in \{0,1\}^3$. We are going to define gadgets for each gate. Let us take $k \in \{1, \ldots, n\}$ and call g_k^* the corresponding gadget associated to g_k. Let us that suppose g_k is an OR gate and that it has fanin 2 and fanout 1 then, we define $g^* : \{0,1\}^6 \to \{0,1\}^6$ as a function that for each input of the form $(x,x,y,y,0,0)$ produces the output $g^*((x,x,y,y,0,0)) = (x \vee y, x \vee y, 0,0,0,0)$. The case fanin 1 and fanout 1 is given by $g^*((x,x,0,0,0,0)) = (x,x,0,0,0,0)$, the case fanin 2 and fanout 2 is given

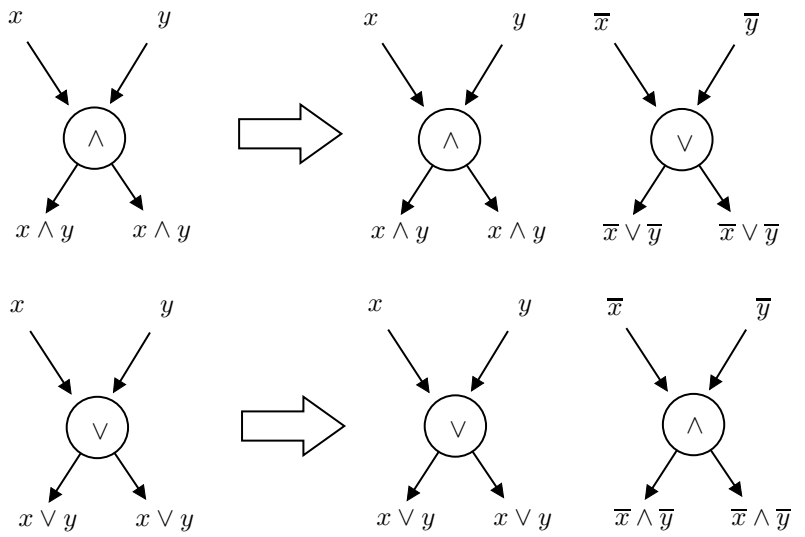


Figure 12: AND and OR gadgets for simulating AND/OR gates with fanin and fanout 2. For other values of fanin and fanout gadgets are the same but considering different number of inputs/outputs

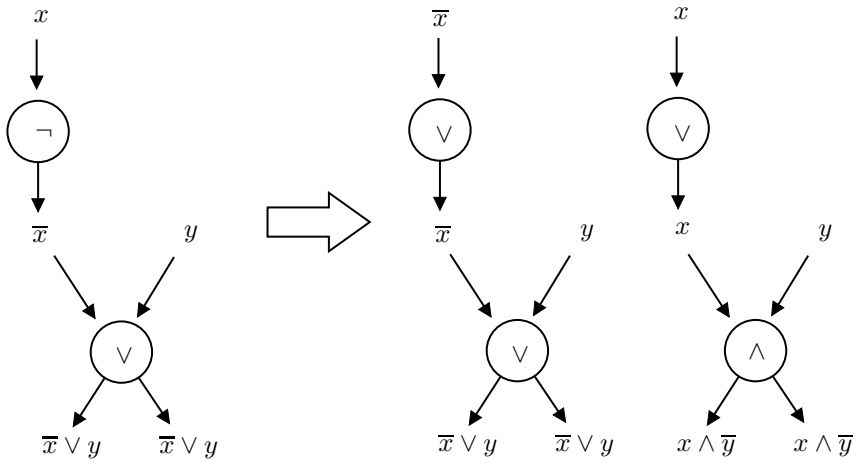


Figure 13: NOT gadget wiring for circuit simulation using gates from $\mathcal{G}_m$. In this case a NOT gate is connected to an OR gate in the original circuit. Copies of the NOT gate in the circuit performing simulation are connected to the copies of the OR gate switched: positive part is connected to negative part of the OR gate and viceversa.

by $g^*((x,x,y,y,0,0)) = (x \vee y, x \vee y, x \vee y, x \vee y, 0, 0)$ and finally case fanin 1 and fanout 2 is given by the same latter function but on input (x,x,0,0,0,0). The AND case is completely analogous. We are going to implement the previous functions as small (constant depth) synchronized circuits that we call block gadgets. More precisely, we are going to identify functions g^* with its correspondent block gadget. The detail on the construction of these circuits that define latter functions are provided in Figures 14, 15 and 16.

Once we have defined the structure of block gadgets, we have to manage connections between them and also manage the fixed 0 inputs that we have added in addition to the zeros that are produced by the blocks as outputs. In order to do that, let us assume that gates g_i and g_j are connected. Note from the discussion on coding above that AND/OR gadgets have between 2 and 4 inputs and outputs fixed to 0. In particular, as it is shown in Figures 14, 15 and 16, all the block gadgets have the same amount of zeros in the input and in the output with the exception of the fanin 1 fanout 2 gates and the fanin 2 fanout 1 gates. However, as $\mathcal{G}$-networks are closed systems (the amount of inputs must be the same that the amount of outputs) we have that, for each fanin 1 fanout 2 gate, it must be a fanin 2 fanout 1 gate and vice versa (otherwise there would be more input than outputs or more outputs than inputs). In other words, there is a bijection between the set of fanin 1 fanout 2 gates and the set of fanin 2 fanout 1. Observe that fanin 2 fanout 1 gates consume 2 zeros in input but produce 4 zeros in output while fanin 1 fanout 2 gates consume 4 in input and produce 2 zeros in output (see Figures 15 and 16). So, between g_i^* and g_j^* we have to distinct two cases: a) if both gates have the same number of inputs and outputs, connections are managed in the obvious way i.e., outputs corresponding to the computation performed by original gate are assigned between g_i^* and g_j^* and each gate uses the same zeros they produce to feed its inputs. b) if g_i^* or g_j^* have more inputs than outputs or vice versa, we have to manage the extra zeros (needed or produced). Without lost of generality, we assume that g_i^* is fanin 2 fanout 1. Then, by latter observation it must exists another gate g_k and thus, a gadget block g_k^* with fanin 1 and fanout 2. We simply connect extra zeros produced by g_i^* to block g_k^* and we do the same we did in previous case in order to manage connections.

Note that F^* is constructible in **DLOGSPACE** as it suffices to read the standard representation of F and produce the associated block gadgets which have constant size. In addition we have that previous encoding $g \to g^*$ induce a block map $\phi : \{0,1\}^V \to \{0,1\}^{V^+}$ where $|V^+| = \mathcal{O}(|V|)$ and that $\phi \circ F = F^{*T} \circ \phi$ where $T = 6$ is the size of each gadget block in F^*. We conclude that $F^* \in \Gamma(\mathcal{G}_{m,2})$ simulates F in space $|V^+| = \mathcal{O}(|V|)$ and time $T = 6$ and thus, $\Gamma(\mathcal{G}_m) \preccurlyeq_S^T \Gamma(\mathcal{G}_{m,2})$ where T is constant and S is a linear function. .

□

Corollary 2. *The family $\Gamma(\mathcal{G}_{m,2})$ is strongly universal.*

Proof. Result is direct from Theorem 15 ($\Gamma(\mathcal{G}_m)$ is strongly universal) and Theorem 16 ($\Gamma(\mathcal{G}_m) \preccurlyeq_S^T \Gamma(\mathcal{G}_{m,2})$ where T is constant and S is a linear function). □

Now we show the universality of $\Gamma(\mathcal{G}_m)$. The proof is essentially a consequence of [21, Theorem 6.2.5]. Roughly, latter result starts with alternated monotone circuit which has only fanin 2 and fanout 2 gates (previous results in the same reference show that one can always reduce to this case starting from an arbitrary circuit) and gives an $\mathbf{NC}^1$

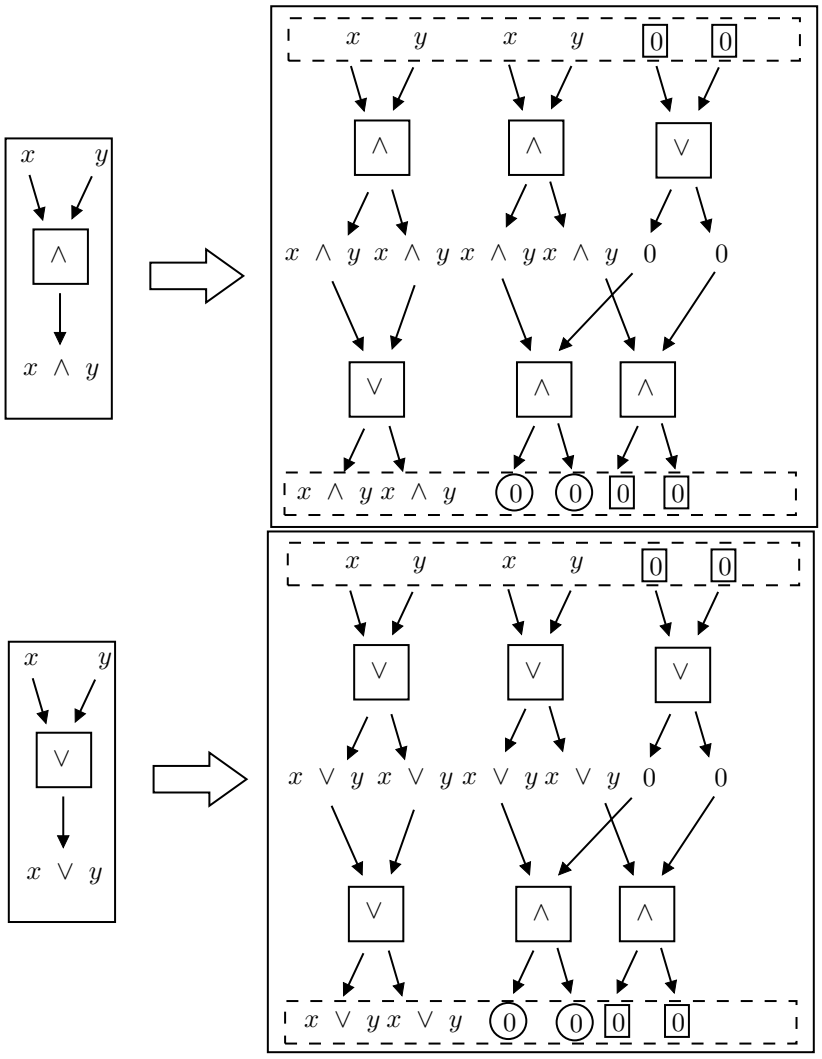


Figure 14: Block gadgets for simulating Fanin 2 Fanout 1 AND/OR gates using only gates in $\mathcal{G}_{m,2}$. Squared zeros represent the amount of zeros that can be used as inputs for the same block. Circled zeros correspond to extra zeros that need to be assigned to a Fanin 1 Fanout 2 gate.

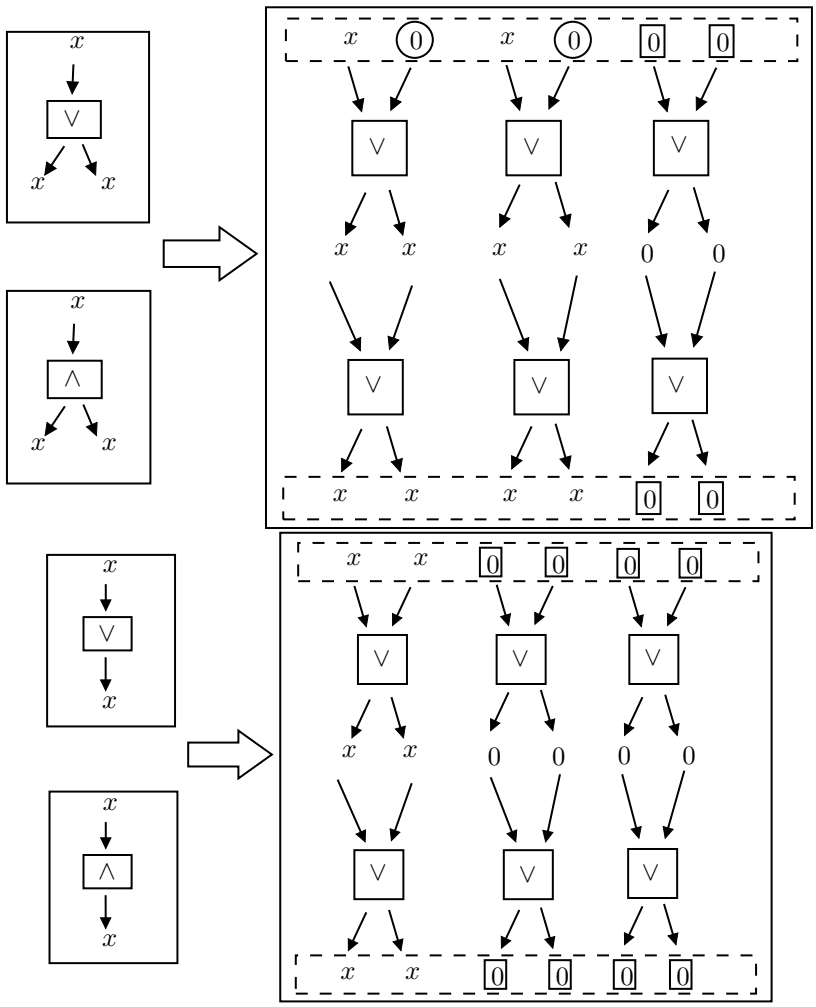


Figure 15: Block gadgets for simulating Fanin 1 Fanout 2 and Fanin 1 Fanout 1 AND/OR gates using only gates in $\mathcal{G}_{m,2}$. Squared zeros represent the amount of zeros that can be used as inputs for the same block. Circled zeros correspond to extra zeros that need to be received from a Fanin 2 Fanout 1 gate.

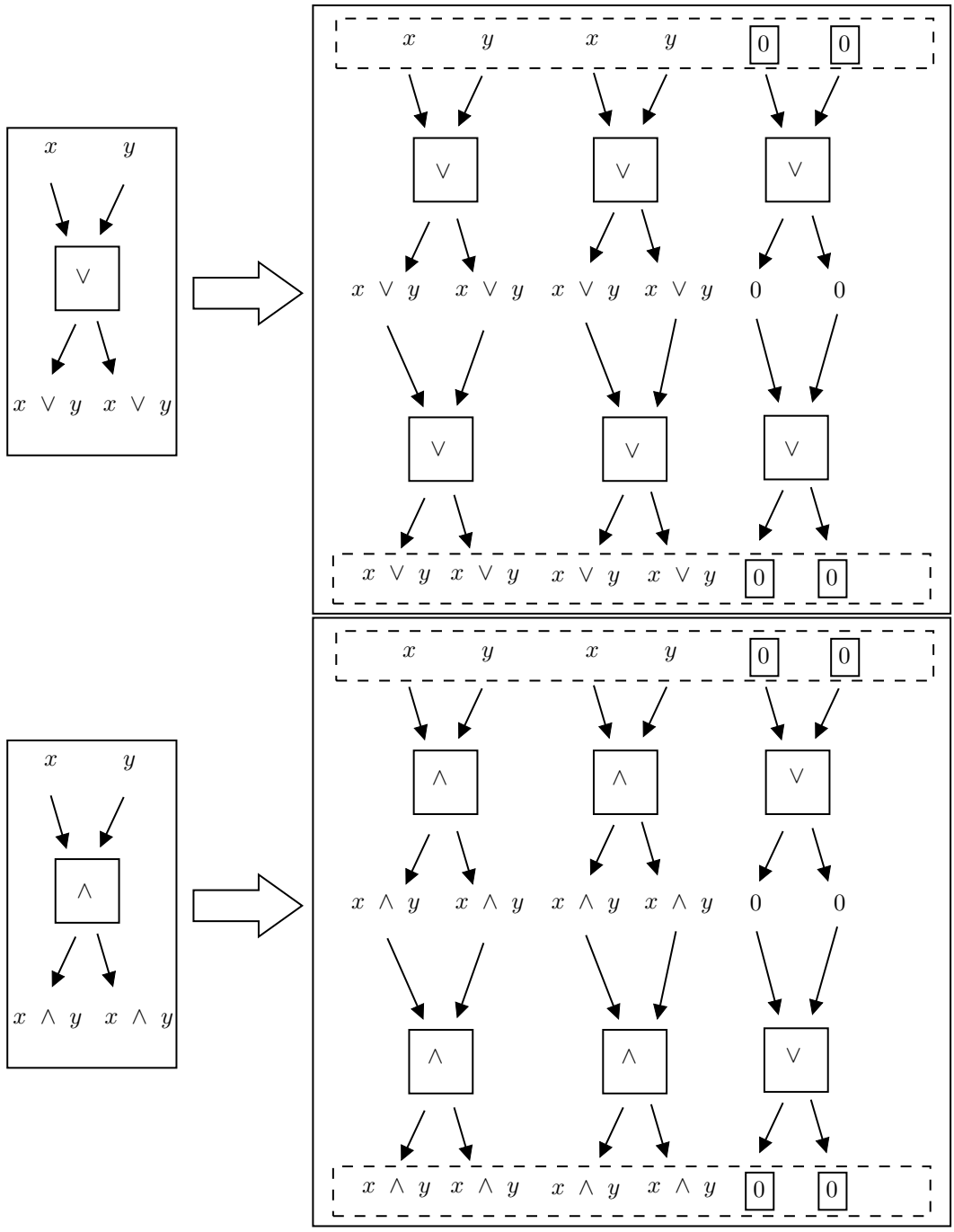


Figure 16: Block gadgets for simulating fanin 2 fanout 2 AND/OR gates using only gates in $\mathcal{G}_{m,2}$. Squared zeros represent the amount of zeros that can be used as inputs for the same block.

construction of a synchronous circuit preserving latter properties. We need additional care here because we want a reusable circuit whose output is fed back to its input. Note also that the construction uses quadratic space in the number of gates of the circuit given in input, so we cannot show strong universality this way but only universality.

Theorem 17. *The family $\Gamma(\mathcal{G}_m)$ of all $\mathcal{G}_m$-networks is universal*

Proof. Let $F : Q^k \to Q^k$ be some arbitrary network with a circuit representation $C : \{0,1\}^n \to \{0,1\}^n$ such that $n = k^{\mathcal{O}(1)}$. By [21, Theorem 6.2.5] we can assume that there exists a circuit $C' : \{0,1\}^{n'} \to \{0,1\}^{n'}$ where $n' = \mathcal{O}(n^2)$ such that C' is synchronous alternated and monotone. In addition, every gate in C' has fanin and fanout 2. We remark that latter reference do not only provides the standard encoding of C' but also give us a **DLOGSPACE** algorithm (it is actually $\mathbf{NC}^1$) which takes the standard representation of $C : \{0,1\}^n \to \{0,1\}^n$ and produces C'. We are going to slightly modify latter algorithm in order to construct not only a circuit but a $\mathcal{G}_m$-network. In fact, the only critical point is to manage the identification between outputs and inputs. This is not direct from the result by Ruzzo et al. as their algorithm involves duplication of inputs and also adding constant inputs. In order to manage this, it suffices to simply modify their construction in order to mark original, copies and constant inputs. Then, as $\mathcal{G}_m$ includes COPY gates and also AND/OR gates with fanout 1, one can always produce copies of certain input if we need more, or erase extra copies by adding and small tree of $\mathcal{O}(\log(n))$ depth. Same goes for constant inputs. Formally, at the end of the algorithm, the **DLOGSPACE** can read extra information regarding copies and constant inputs, and then can construct $\mathcal{O}(\log(n))$ depth circuit that produces a coherent encoding for inputs and outputs. This latter construction defines a $\mathcal{G}_m$-network $G : \{0,1\}^{n''} \to \{0,1\}^{n''}$ and an encoding $\phi : Q^n \to \{0,1\}^{n''}$ where $n'' = \mathcal{O}(n^2)$ such that $\phi \circ F = G^T \circ \phi$ where $T = \mathcal{O}(\text{depth(C')} + \log(n))$. Thus, $\mathcal{G}_m$ is universal. □

We can now state the following direct corollary.

Corollary 3. *Let $\mathcal{F}$ be a strongly universal automata network family. Then, $\mathcal{F}$ is universal.*

Proof. In order to show the result, it suffices to exhibit a $\mathcal{G}$-network family ($\mathcal{G}$-networks are bounded degree networks) which is strongly universal and universal at the same time. By Theorem 17 we take $\mathcal{G} = \mathcal{G}_m$ and thus, corollary holds. □

Corollary 4. *Let $\mathcal{G}$ be either $\mathcal{G}_m$ or $\mathcal{G}_{m,2}$. Any family $\mathcal{F}$ that has coherent $\mathcal{G}$-gadgets contains a subfamily of bound degree networks with bounded degree representation which is (strongly) universal. Any CSAN family with coherent $\mathcal{G}$-gadgets is (strongly) universal.*

3.5. Closure and synchronous closure

Although monotone gates are sometimes easier to realize in concrete dynamical system which make the above results useful, there is nothing special about them to achieve universality: any set of gates that are expressive enough for Boolean functions yields the same universality result. Given a set of maps $\mathcal{G}$ over alphabet Q, we define its *closure* $\overline{\mathcal{G}}$ as the set of maps that are computed by circuits that can be built using only gates from $\mathcal{G}$. More precisely, $\overline{\mathcal{G}}$ is the closure of $\mathcal{G}$ by composition, *i.e.* forming from maps $g_1 : Q^{I_1} \to Q^{O_1}$ and $g_2 : Q^{I_2} \to Q^{O_2}$ (with I_1, I_2, O_1, O_2

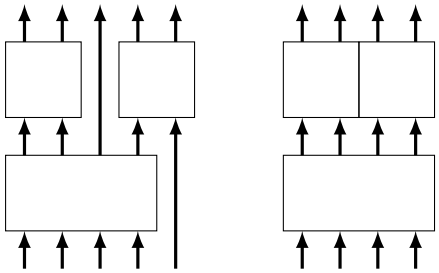


Figure 17: Non-synchronous composition (on the left) and synchronous composition (on the right).

disjoint) a composition g by plugging a subset of outputs $O \subseteq O_2$ of g_1 into a subset of inputs $I \subseteq I_2$ of g_2, thus obtaining $g : Q^{I_1 \cup I_2 \setminus I} \to Q^{O_1 \setminus O \cup O_2}$ with

$$g(x)_o = \begin{cases} g_1(x_{I_1})_o & \text{if } o \in O_1 \setminus O \\ g_2(y)_o & \text{if } o \in O_2 \end{cases}$$

where $y_j = x_j$ for $j \in I_2 \setminus I$ and $y_j = g_1(x_{I_1})_{\pi(j)}$ where $\pi : I \to O$ is the chosen bijection between I and O (the wiring of outputs of g_1 to inputs of g_2). A composition is *synchronous* if either $I = \emptyset$ or $I = I_2$. We then define the synchronous closure $\overline{\mathcal{G}}^2$ as the closure by synchronous composition. The synchronous composition correspond to synchronous circuits with gates in $\mathcal{G}$. A $\mathcal{G}$-circuit is a sequence of compositions starting from elements of $\mathcal{G}$. It is synchronous if the compositions are synchronous. The depth of a $\mathcal{G}$-circuit is the maximal length of a path from an input to an output. In the case of a synchronous circuits, all such path are of equal length.

Remark 4. *The above definitions are very close to the classical notion of clones [22]. However, we stress that, in our case, projections maps $Q^k \to Q$ are generally not available, nor duplication maps $x \mapsto (x, x)$ allowing to use the same variable several times. This is important because in a given dynamical systems, erasing or duplicating information might be impossible (think about reversible systems) and hiding it into some non-coding part might be complicated.*

Proposition 1. *Fix some alphabet Q and consider two finite sets of maps $\mathcal{G}$ and $\mathcal{G}'$ over alphabet Q such that:*

- *either contains the identity map $Q \to Q$ and is such that $\overline{\mathcal{G}}$ contains $\mathcal{G}'$,*
- *or there is an integer k such that $\overline{\mathcal{G}}^2_k$, the set of elements of $\overline{\mathcal{G}}^2$ that can be realized by a circuit of depth k, contains $\mathcal{G}'$.*

Then, any family $\mathcal{F}$ that has coherent $\mathcal{G}$-gadgets has coherent $\mathcal{G}'$-gadgets.

Proof. Suppose first that the first item holds. Since $\overline{\mathcal{G}}$ contains $\mathcal{G}'$ there must exist a circuit made of gates from $\mathcal{G}$ that produces any given element $g \in \mathcal{G}'$. One then wants to apply gadget glueing on gadgets from $\mathcal{G}$ to mimic the composition and thus obtain a gadget corresponding to g. However this doesn't work as simply because propagation delay is a priori not respected at each gate in the circuit composition yielding g and there is a risk that information arrives distinct delays at different outputs. However, since $\mathcal{G}$ contains the identity map, there is a corresponding gadget in the family that actually implements a delay line. This additionnal gadget solves the problem: it is straightforward to transform by padding with identity gates all circuit with gates in $\mathcal{G}$ into synchronous ones.

Moreover, by padding again, we can assume that the finite set of such circuits computing elements of $\mathcal{G}_m$ are all of same depth. It is then straightforward to translate this set of circuits into coherent $\mathcal{G}_m$-gadgets by iterating gadget glueing and using Lemma 8.

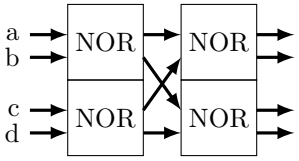
If the second item holds the situation is actually simpler because the synchronous closure contains only synchronous circuits of gates from $\mathcal{G}_{m,2}$ so we can directly translate the circuits producing the maps of $\mathcal{G}_{m,2}$ into gadgets via gadget glueing by Lemma 8 as in the previous case. Moreover, the hypothesis is that all elements of $\mathcal{G}'$ are realized by circuit of same depth so we get gadgets that share the same time constant.

□

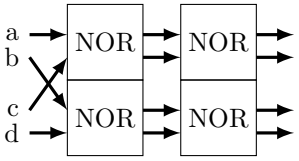
As a direct corollary of Proposition 1, we can extend the results about strong universality of $\mathcal{G}_{m,2}$ to other families of $\mathcal{G}$-networks associated to elementary Boolean gates, like $\mathcal{G}_{\mathrm{NOR}}$ and $\mathcal{G}_{\mathrm{NAND}} = \{\mathrm{NAND}(x,y) = (\overline{x \wedge y}, \overline{x \wedge y})\}$. Note however that classical results on Boolean gates and clone theory cannot be applied immediately (see Remark 4) and the expected universality result requires a little bit of care.

Corollary 5. *The families $\Gamma(\mathcal{G}_{NOR})$ and $\Gamma(\mathcal{G}_{NAND})$ are strongly universal.*

Proof. First, since NAND and NOR gates are conjugated by negation, it is clear that families $\Gamma(\mathcal{G}_{\mathrm{NOR}})$ and $\Gamma(\mathcal{G}_{\mathrm{NAND}})$ simulate each other with time constant 1 via a block embedding that just apply $x \mapsto \overline{x}$ at each node. It is thus sufficient to prove that $\Gamma(\mathcal{G}_{\mathrm{NOR}})$ is strongly universal. Consider the two maps $\alpha : \{0,1\}^4 \to \{0,1\}^4$ and $\omega : \{0,1\}^4 \to \{0,1\}^4$ that are synchronous $\mathcal{G}_{\mathrm{NOR}}$-circuits of depth 2 defined by :



$\alpha(a,b,c,d)$



$\omega(a,b,c,d)$

By Proposition 1 (second item), the family $\Gamma(\mathcal{G}_{\mathrm{NOR}})$ has coherent $\mathcal{G}$-gadgets where $\mathcal{G} = \{\alpha, \omega\}$ and therefore simulates the family $\Gamma(\mathcal{G})$ with constant spatio-temporal rescaling factors by Lemma 13. Now observe that for any $x, y \in \{0,1\}$ it holds that $\alpha(x,x,y,y) = (a,a,a,a)$ with $a = x \wedge y$ and $\omega(x,x,y,y) = (o,o,o,o)$ with $o = x \vee y$. This implies that family $\Gamma(\mathcal{G})$ simulates $\Gamma(\mathcal{G}_{m,2})$ with spatial rescaling factor 2 and temporal rescaling factor 1, simply by doubling each node because AND and OR gates of type $\{0,1\}^2 \to \{0,1\}^2$ in $\mathcal{G}_{m,2}$ are such that $AND(x,y) = (a,a)$ and $OR(x,y) = (o,o)$. We deduce that $\Gamma(\mathcal{G})$ and therefore $\Gamma(\mathcal{G}_{\mathrm{NOR}})$ are strongly universal. □

3.5.1. Game of life is strongly universal

Theorem 18. *The family of outer-totalistic CSAN networks with $B = 3$ and $S = 2, 3$ i.e.* Game of life *automata networks, is strongly universal.*

Proof. The result holds as a direct consequence of the Lemma 14 which tell us that Game of life automata networks have coherent $\mathcal{G}_{\mathrm{NOR}}$ gadgets, the Corollary 1 which tell us that the family of Game of life automata networks simulates

$\Gamma(\mathcal{G}_{\mathrm{NOR}})$ in constant time and linear space and finally, the Corollary 5 which tell us that the familly $\Gamma(\mathcal{G}_{\mathrm{NOR}})$ is strongly universal and thus, the family of Game of life automata networks are strongly universal. □

Remark 5. *We would like to stress three points about the latter result:*

1. *the gadget used in the proof of Lemma 14 is simpler than the case of cellular automata and intrinsic universality (see [23] for more details). In particular, the fact that the communication graph can be chosen freely, allow us to transmit information and perform calculations in less time.*
2. *the result is an improvement of the result obtained in the cellular automata context since, as shown in [19, Corollary 4], intrinsic universality is not enough for strong universlity. In fact, Theorem 18 in the same reference provides some insight on how the properties of the communication graph play an important role in terms of the universality.*
3. *the result is an application of the tools developed in the present article which can be easily applied to any other family: given a family of automata networks one can show the strong universality by simply showing that the family admits a set of coherent gadgets. We stress that this approach allows to derive a perfectly rigorous and computer checkable proof of many facts implied by strong universality (e.g. [19, Corollary 1] and [19, Theorem 15]) from a rather small set of observations on a finite set of pseudo-orbits of small automata networks (see proof of Lemma 14).*

3.6. Super-polynomial periods without universality

A universal family must exhibit super-polynomial periods, however universality is far from necessary to have this dynamical feature. In this subsection we define the family of wire networks to illustrate this.

In order to do that, we need the following classical result about the growth of Chebyshev function and prime number theorem.

Lemma 19. *[24] Let $m \geq 2$ and $\mathcal{P}(m) = \{p \leq m \mid p \text{ prime}\}$. If we define $\pi(m) = |\mathcal{P}(m)|$ and $\theta(m) = \sum_{p \in \mathcal{P}(m)} \log(p)$ then we have $\pi(m) \sim \frac{m}{\log(m)}$ and $\theta(m) \sim m$.*

By using the Lemma 19 we can construct automata networks with non-polynomial cycles simply by making disjoint union of rotations (*i.e.* network whose interaction graph is a cycle that just rotate the configuration at each step). Indeed, it is sufficient to consider rotations on cycle whose length are successive prime numbers. It turns out that these automata networks are exactly $\mathcal{G}_w$-networks where $\mathcal{G}_w$ is a single 'wire gate': $\mathcal{G}_w = \{id_B\}$ where id_B is the identity map over $\{0, 1\}$.

Formally, according to Definition 6, for any $\mathcal{G}_w$-network $F : Q^V \to Q^V$ there exist a partition $V = C_1 \cup C_2 \ldots \cup C_k$ where $C_i = \{u_1^i \ldots, u_{l_i}^i\}$ with $l_i \geq 2$ for each $i = 1, \ldots, k$ and $F(x)_{u_{s+1 \bmod l_i}^i} = x_{u_s^i}$ for any $x \in Q^V$ and $0 \leq s \leq l_i$.

Theorem 20. *Any family $\mathcal{F}$ that has coherent $\mathcal{G}_w$-gadgets has superpolynomial cycles, more precisely: there is some $\alpha > 0$ such that for infinitely many $n \in \mathbb{N}$, there exists a network $F_n \in \mathcal{F}$ with $O(n)$ nodes and a periodic orbit of size $\Omega(\exp(n^\alpha))$.*

Proof. Taking the notations of Lemma 19, define for any n the $\mathcal{G}_w$-network G_n made of disjoint union of circuits of each prime length less than n. G_n has size at most $n\pi(n)$ and if we consider a configuration x which is in state 1 at exactly one node in each of the $\pi(n)$ disjoint circuit, it is clear that the orbit of x is periodic of period $\exp\theta(n)$. Therefore, from Lemma 19, for any n, G_n is a circuit of size $m \leq n\pi(n)$ with a periodic orbit of size $\theta(n) \in \Omega(\exp(\sqrt{m \log m}))$. By hypothesis there are linear maps T and S such that for any n, there is F_n that simulates G_n (by Lemma 13), therefore F_n also has a super-polynomial cycles (see [19, Lemma 6] for more results). □

3.7. Conjunctive networks and $\mathcal{G}_{conj}$-networks

Let $G = (V, E)$ be any directed graph. The conjunctive network associated to G is the automata network $F_G : \{0,1\}^V \to \{0,1\}^V$ given by $F(x)_i = \wedge_{j \in N^-(i)} x_j$ where $N^-(i)$ denotes the incoming neighborhood of i. Conjunctive networks are thus completely determined by the interaction graph and a circuit representation can be deduced from this graph in **DLOGSPACE**. We define the family $\mathcal{F}_{conj}$ as the set of conjunctive networks together with the standard representation $\mathcal{F}^*_{conj}$ which are just directed graphs encoded as finite words in a canonical way.

Remark 6. *We can of course do the same with disjunctive networks. Any conjunctive network F_G on graph G is conjugated to the disjunctive network F'_G on the same graph by the negation map $\rho : \{0,1\}^V \to \{0,1\}^V$ defined by $\rho(x)_i = 1 - x_i$, formally $\rho \circ F_G = F'_G \circ \rho$. In particular, this means that the families of conjunctive and disjunctive networks simulate each other. In the sequel we will only state results for conjunctive networks while they hold for disjunctive networks as well.*

Let us now consider the set $\mathcal{G}_{conj} = \{\text{AND}, \text{COPY}\}$. $\mathcal{G}_{conj}$-networks are nothing else than conjunctive networks with the following degree constraints: each node has either in-degree 1 and out-degree 2, or in-degree 2 and out-degree 1. The following theorem shows that, up to simulation, these constraints are harmless.

Theorem 21. *The family of $\mathcal{G}_{conj}$-networks simulates the family $(\mathcal{F}_{conj}, \mathcal{F}^*_{conj})$ of conjunctive networks in linear time and polynomial space.*

Proof. Let F be an arbitrary conjunctive network on graph $G = (V, E)$ with n nodes. Its maximal in/out degree is at most n. For each node of indegree $i \leq n$ we can make a tree-like $\mathcal{G}_{conj}$-gadget with i inputs and 1 output that computes the conjunction of its i inputs in exactly n steps: more precisely, we can build a sub-network of size $O(n)$ with i identified 'input' nodes of fanin 1 and one identified output node of fanout 1 such that for any $t \in \mathbb{N}$ the state of the output node at time $t+n$ is the conjunction of the states of the input nodes at time t (the only sensible aspect is to maintain synchronization in the gadget, see Figure 18).

We do the same for copying the output of a gate i times and dealing with arbitrary fanout. Then we replace each node of F by a meta node made of the two gadgets to deal with fanin/fanout and connect everything together according to graph G (note that fanin/fanout is granted to be 1 in the gadgets so connections respect the degree constraints). We obtain in **DLOGSPACE** a $\mathcal{G}_{conj}$-network of size polynomial in n that simulates F in linear time. □

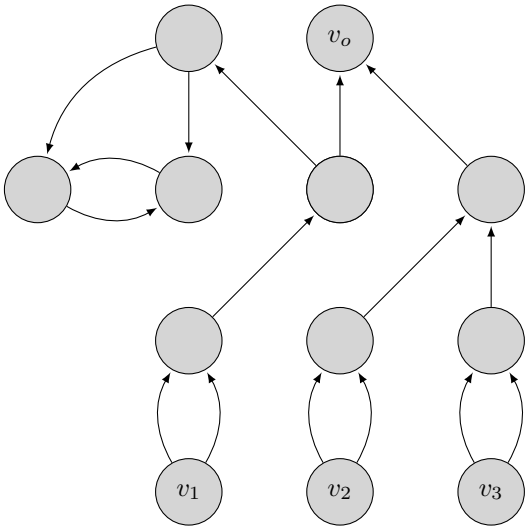


Figure 18: Fanin gadget of degree 3. For any configuration x, $F^3(x)_{v_o} = x_{v_1} \wedge x_{v_2} \wedge x_{v_3}$.

Remark 7. *The family of conjunctive networks can produce super-polynomial periods but is not universal. There are several ways to show this. It is for instance impossible to produce super-polynomial transients within the family [25, Theorem 3.20] so [19, Corollary 1] allows to conclude. One could also use [19, Corollary 2] since a node in a strongly connected component of a conjunctive network must have a trace period of at most the size of the component (actually much more in known about periods in conjunctive networks through the concept of loop number or cyclicity, see [25]).*

3.8. Super-polynomial transients and periods without universality

Let us consider in this section alphabet $Q = \{0, 1, 2\}$. We are going to define a set $\mathcal{G}_t$ such that $\mathcal{G}_t$-networks exhibit super-polynomial transients but are not universal. To help intuition, $\mathcal{G}_t$-networks can be though as standard conjunctive networks on $\{0, 1\}$ that can in some circumstances produce state 2 which is a spreading state (a node switches to state 2 if one of its incoming neighbors is in state 2). The extra state 2 will serve to mark super-polynomial transients, but it cannot escape a strongly connected component once it appears in and, as we will see, $\mathcal{G}_t$-networks are therefore too limited in their ability to produce large periodic behavior inside strongly connected components.

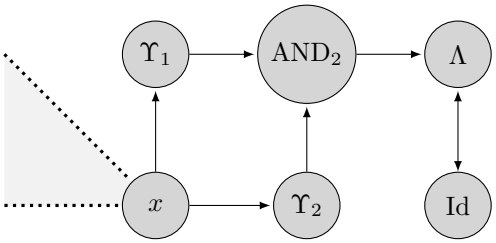


Figure 19: Freezing the result of a test in a $\mathcal{G}_t$-network. The module $T(x)$ is made of the nodes marked Υ, AND_2, Λ and Id. Observe that each node represents some output of its corresponding label (for more details on $\mathcal{G}$-networks see Definition 6). Each gate has one output with the exception of the gate Υ which is represented by two nodes. The module $T(x)$ reads the value of node x belonging to an arbitrary $\mathcal{G}_t$-network (represented in light gray inside dotted lines). The output Λ is fed back to its control input via the Id node (self-loops are forbidden in $\mathcal{G}_t$-networks). Note that x as well as the rest of the network is not influenced by the behavior of the gates of the module $T(x)$.

$\mathcal{G}_t$ is made of the following maps:

$$\mathrm{AND}_{\{0,1\}} : (x,y) \mapsto \begin{cases} 2 & \text{if } 2 \in \{x,y\} \\ x \wedge y & \text{else.} \end{cases}$$

$$\mathrm{AND}_2 : (x,y) \mapsto \begin{cases} 2 & \text{if } 2 \in \{x,y\} \text{ or } x = y = 1 \\ 0 & \text{else.} \end{cases}$$

$$\Lambda : (x,y) \mapsto \begin{cases} 2 & \text{if } 2 \in \{x,y\} \\ x & \text{else.} \end{cases}$$

$$\mathrm{Id} : x \mapsto x.$$

$$\Upsilon : x \mapsto (x,x).$$

$\mathcal{G}_t$-networks can produce non-polynomial periods by disjoint union of rotations of prime lengths as in Theorem 20, but they can also wait for a global synchronization of all rotations and freeze the result of the test for this synchronization condition inside a small feedback loop attached to a "controlled AND map".

More precisely, as shown in Figure 19 we can use in the context of any $\mathcal{G}_t$-network a small module $T(x)$ of made of five nodes with the following property: if the Λ node of the module is in state 0 in some initial configuration, then it stays in state 0 as long as nodes x is not in state 1, and when $x = 1$ at some time step t then from step $t + 2$ on the Λ node is in state 2 at least one step every two steps. This module is the key to control transient behavior.

Besides, the map $\mathrm{AND}_{\{0,1\}}$ behaves like standard Boolean AND map when its inputs are in $\{0, 1\}$. More generally, by combining such maps in a tree-like fashion, one can build modules $A(x_1, \ldots, x_k)$ for any number k of inputs with a special output node which has the following property for some time delay $\Delta \in O(\log(k))$: the output node at time $t + \Delta$ is in state 1 if and only if all nodes x_i (with $1 \leq i \leq k$) are in state 1 at time t.

Combining these two ingredients, we can build upon the construction of Theorem 20 to obtain non-polynomial transients in any family having coherent $\mathcal{G}_t$-gadgets.

Theorem 22. *Any family $\mathcal{F}$ that has coherent $\mathcal{G}_t$-gadgets has superpolynomial transients, more precisely: there is*

some $\alpha > 0$ such that for any $n \in \mathbb{N}$, there exists a network $F_n \in \mathcal{F}$ with $O(n)$ nodes and a configuration x such that $F_n^t(x)$ is not in an attractor of F_n with $t \in \Omega(\exp(n^\alpha))$.

Proof. Like in Theorem 20, the key of the proof is to show that there is a $\mathcal{G}_t$-network with transient length as in the theorem statement, then the property immediately holds for networks of the family $\mathcal{F}$ by Lemma 13 and [19, Lemma 6].

For any $n > 0$ we construct a $\mathcal{G}_t$-network G_n made of two parts:

- the 'bottom' part of G_n uses a polynomial set of nodes B_n and consists in a disjoint union of circuits for each prime length less than n as in Theorem 20, but where for each prime p, the circuit of length p has a node v_p which implements a copy gate COPY, thus not only sending its value to the next node in the circuit, but also outputting it to the second part of G_n;
- the 'top' part of G_n is made of a module $A(x_1, \dots, x_k)$ connected to all nodes v_p as inputs and whose output is connected to a test module $T(x)$ as in Figure 19.

Note that the size of G_n is polynomial in n. With this construction we have the following property as soon as the modules $A(x_1, \dots, x_k)$ and $T(x)$ are initialized to state 0 everywhere: as long as nodes v_p are not simultaneously in state 1 then the output of the test module $T(x)$ stays in state 0; moreover, if at some time t nodes v_p are simultaneously in state t, then after time $t + O(\log(t))$ the output node of module $T(x)$ is in state 1 one step every two steps. This means that $t + O(\log(t))$ is a lower bound on the transient of the considered orbit. To conclude the theorem it is sufficient to consider the initial configuration where all nodes are in state 0 except the successor of node v_p in each circuit of prime length p, which are in state 1. In this case it is clear that the first time t at which all nodes v_p are in state 1 is the product of prime numbers less than n. As in theorem 20, we conclude thanks to Lemma 19. □

As said above, $\mathcal{G}_t$-networks are limited in their ability to produce large periods. More precisely, as shown by the following lemma, their behavior is close enough to conjunctive networks so that it can be analyzed as the superposition of the propagation/creation of state 2 above the behavior of a classical Boolean conjunctive network. To any $\mathcal{G}_t$-network F we associate the Boolean conjunctive network F^* with alphabet $\{0, 1\}$ as follows: nodes with local map $\mathrm{AND}_{\{0,1\}}$ or AND_2 are simply transformed into nodes with Boolean conjunctive local maps on the same neighbors, nodes with local maps Υ or Id are left unchanged (only their alphabet changes), and nodes with map $\Lambda(x, y)$ are transformed into a node with only x as incoming neighborhood.

Lemma 23. *Let F be a $\mathcal{G}_t$-network with node set V and F^* its associated Boolean conjunctive network. Consider any $x \in \{0, 1, 2\}^V$ and any $x^* \in \{0, 1\}^V$ such that the following holds:*

$$\forall v \in V : x_v \in \{0, 1\} \Rightarrow x^*_v = x_v,$$

then the same holds after one step of each network:

$$\forall v \in V : F(x)_v \in \{0, 1\} \Rightarrow F^*(x^*)_v = F(x)_v.$$

Proof. It is sufficient to check that if $F(x)_v \neq 2$, it means that all its incoming neighbors are in $\{0,1\}$ so x and x^* are equal on these incoming neighbors, and that it only depend on neighbor a in the case of a local map $\Lambda(a,b)$. In any case, we deduce $F^*(x^*)_v = F(x)_v$ by definition of F^*. □

$\mathcal{G}_t$-networks are close to Boolean conjunctive networks as shown by the previous lemma. The following result shows that this translates into strong limitations in their ability to produce large periods and prevents them to be universal.

Theorem 24. *The family of $\mathcal{G}_t$-networks is not universal.*

Proof. Consider a Boolean conjunctive automata network F, a configuration x with periodic orbit under F and some node v such that there is a walk of length L from v to v. We claim that $x_v = F^L(x)_v$ so the trace at node v in x is periodic of period less than L. Indeed, in a conjunctive network state 0 is spreading so clearly if $x_v = 0$ then $F^L(x)_v = 0$ and, more generally, $F^{kL}(x)_v = 0$ for any $k \geq 1$. On the contrary, if $x_v = 1$ then we can't have $F^L(x)_v = 0$ because then $F^{Pk}(x)_v = 0$ with P the period of x which would imply $x_v = 0$.

With the same reasoning, if we consider any $\mathcal{G}_t$-network F, any configuration x with periodic orbit and some node v such that there is a walk of length L from v to v, then it holds:

$$x_v = 2 \Leftrightarrow F^L(x)_v = 2.$$

We deduce thanks to Lemma 23 that for any configuration x with periodic orbit of some $\mathcal{G}_t$-network F with n nodes, and for any node v belonging to some strongly connected component, the period of the trace at v starting from x is less than n^2: it is a periodic pattern of presence of state 2 of length less than n superposed on a periodic trace on $\{0,1\}$ of length less than n. We conclude that the family of $\mathcal{G}_t$-networks cannot be universal thanks to [19, Corollary 2]. □

4. Perspectives

The main contribution of this paper is a proof technique to show intrinsic universality of families of automata networks, with all the dynamical and computational consequences such a result implies. As announced earlier, the first perspective is to use of this framework to show universality results of known families. In the third paper of this series, we show using these tools how some non-universal concrete families can recover universality by changing the update schedule of the system[26].

In addition, several research directions directly connected to the notions developed in the present paper are worth being considered. We detail some of them below.

Glueing. We think it would be interesting to understand the properties of the glueing process itself and see what information on the result of the glueing process can be deduced from the knowledge of each network to be glued. We are particularly interested in dynamical properties. In addition, it would be very interesting to explore if latter

process can be seen in the opposite way, i.e., given an automata network, determine if it is possible to decompose the network into glued blocks satisfying some particular properties as gadgets do.

Universality and gadgets. We know how to build families which are universal but not strongly universal by adding a somewhat artificial mechanism that slows down polynomially any useful computation made by networks in the family, giving examples which are universal but requires a superlinear spatio-temporal rescaling factor. However, we don't have any natural example so far of such 'weakly universal' families and we would like to better understand this territory. In the same spirit, we can ask how a strongly universal family can fail to have coherent $\mathcal{G}_m$-gadgets (recall that Corollary 4 only gives a sufficient condition to be strongly universal). We don't think that strongly universality implies coherent $\mathcal{G}_m$-gadgets in general, but the implication might at least be true under some additional hypothesis, and possibly in natural families like $\mathcal{G}$-networks.

$\mathcal{G}$-networks. Proposition 1 together with theorems 20 and 24 provide an interesting starting point to explore the link between different gate sets and the richness of their synchronous closure and the associated family of $\mathcal{G}$-networks. It is natural to further study the hierarchy between sets of gates and we believe that a promising direction would be to study reversible gate sets such as Toffoli or Fredkin gates. Also, we would like to understand how easy it is to deduce global properties of the family of $\mathcal{G}$-networks from the knowledge of $\mathcal{G}$. Typically, one can consider the following decision problem: given a set of gates $\mathcal{G}$, decide whether the family of $\mathcal{G}$-networks strongly universal. Is this problem undecidable? If it is the case, what is the minimum number of gates in $\mathcal{G}$ to obtain undecidability?

References

[1] W. S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, The Bulletin of Mathematical Biophysics 5 (4) (1943) 115–133. doi:10.1007/bf02478259.

[2] R. Thomas, Boolean formalization of genetic control circuits, Journal of Theoretical Biology 42 (3) (1973) 563–585. doi:10.1016/0022-5193(73)90247-6.

[3] S. Kauffman, Homeostasis and differentiation in random genetic control networks, Nature 224 (5215) (1969) 177–178. doi:10.1038/224177a0.

[4] M. Gadouleau, S. Riis, Memoryless computation: new results, constructions, and extensions, Theoretical Computer Science 562 (2015) 129–145.

[5] E. G. Ch., P. Montealegre, Computational complexity of threshold automata networks under different updating schemes, Theor. Comput. Sci. 559 (2014) 3–19. doi:10.1016/j.tcs.2014.09.010.

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

[7] A. Wu, A. Rosenfeld, Cellular graph automata. ii. graph and subgraph isomorphism, graph structure recognition, Information and Control 42 (1979) 330–353. doi:10.1016/S0019-9958(79)90296-1.

[8] A. Wu, A. Rosenfeld, Cellular graph automata. i. basic concepts, graph property measurement, closure properties, Information and Control 42 (3) (1979) 305 – 329. doi:10.1016/S0019-9958(79)90288-2.

[9] E. Goles, J. Olivos, Periodic behaviour of generalized threshold functions, Discrete Mathematics 30 (2) (1980) 187 – 189. doi:http://dx.doi.org/10.1016/0012-365X(80)90121-1.

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

[11] E. Goles, P. Montealegre, V. Salo, I. Törmä, Pspace-completeness of majority automata networks, Theor. Comput. Sci. 609 (2016) 118–128. doi:10.1016/j.tcs.2015.09.014.

[12] E. Goles, P. Montealegre, Computational complexity of threshold automata networks under different updating schemes, Theoretical Computer Science 559 (2014) 3–19.

[13] E. Goles, P. Montealegre, V. Salo, I. Törmä, Pspace-completeness of majority automata networks, Theoretical Computer Science 609 (2016) 118–128.

[14] E. Goles, P. Montealegre, K. Perrot, Freezing sandpiles and boolean threshold networks: Equivalence and complexity, Advances in Applied Mathematics 125 (2021) 102161.

[15] E. Goles, M. Montalva-Medel, P. Montealegre, M. Ríos-Wilson, On the complexity of generalized q2r automaton, Advances in Applied Mathematics 138 (2022) 102355.

[16] C. L. Barrett, H. B. Hunt, M. V. Marathe, S. Ravi, D. J. Rosenkrantz, R. E. Stearns, Complexity of reachability problems for finite discrete dynamical systems, Journal of Computer and System Sciences 72 (8) (2006) 1317–1345. doi:10.1016/j.jcss.2006.03.006.

[17] C. Barrett, H. B. H. III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, On some special classes of sequential dynamical systems, Annals of Combinatorics 7 (4) (2003) 381–408. doi:10.1007/s00026-003-0193-z.

[18] M. Folschette, L. Paulevé, M. Magnin, O. Roux, Sufficient conditions for reachability in automata networks with priorities, Theoretical Computer Science 608 (2015) 66–83. doi:10.1016/j.tcs.2015.08.040.

[19] M. Ríos-Wilson, G. Theyssier, Intrinsic Universality in Automata Networks I: Families and Simulations, preprint available on HAL and arXiv (2023).

[20] M. Gadouleau, On the influence of the interaction graph on a finite dynamical system, Natural Computing 19 (1) (2019) 15–28. doi:10.1007/s11047-019-09732-y.

[21] R. Greenlaw, H. J. Hoover, W. L. Ruzzo, Limits to Parallel Computation, Oxford University Press, 1995. doi:10.1093/oso/9780195085914.001.0001.

[22] H. E. Vaughan, Emil l. post. the two-valued iterative systems of mathematical logic. annals of mathematics studies, no. 5 Journal of Symbolic Logic 6 (3) (1941) 114–115. doi:10.2307/2268608.
URL https://doi.org/10.2307%2F2268608

[23] B. Durand, Z. Róka, Cellular Automata: a Parallel Model, Vol. 460 of Mathematics and its Applications., Kluwer Academic Publishers, 1999, Ch. The game of life:universality revisited., pp. 51–74.

[24] G. Hardy, E. Wright, R. Heath-Brown, D. Heath-Brown, J. Silverman, A. Wiles, An Introduction to the Theory of Numbers, Oxford mathematics, OUP Oxford, 2008.
URL https://books.google.fr/books?id=rey9wfSaJ9EC

[25] B. D. Schutter, B. D. Moor, On the sequence of consecutive powers of a matrix in a boolean algebra, SIAM Journal on Matrix Analysis and Applications 21 (1) (1999) 328–354. doi:10.1137/s0895479897326079.
URL https://doi.org/10.1137%2Fs0895479897326079

[26] M. Ríos-Wilson, G. Theyssier, Intrinsic Universality in Automata Networks III: Symmetry versus Asynchronism, preprint available on HAL and arXiv (2023).