Foundations of block-parallel automata networks

Kévin Perrot[a], Sylvain Sené[a], Léah Tapin[a]

[a]*Aix-Marseille Univ., CNRS, LIS, Marseille, France*

Abstract

We settle the theoretical ground for the study of automata networks under block-parallel update schedules, which are somehow dual to the block-sequential ones, but allow for repetitions of automaton updates. This gain in expressivity brings new challenges, and we analyse natural equivalence classes of update schedules: those leading to the same dynamics, and to the same limit dynamics, for any automata network. Countings and enumeration algorithms are provided, for their numerical study. We also prove computational complexity bounds for many classical problems, involving fixed points, limit cycles, the recognition of subdynamics, reachability, *etc.* The PSPACE-completeness of computing the image of a single configuration lifts the complexity of most problems, but the landscape keeps some relief, in particular for reversible computations.

Keywords: Automata networks, block-parallel, update modes, equivalence classes, counting, enumeration, computational, complexity.

1. Introduction

Since the seminal work of McCulloch and Pitts on neural networks [35], distributed models of computation, where individual entities collectively perform a global computation through local interactions, received a great amount of attention. Automata networks are such a model, which have successfully been employed for the modelling of gene regulation mechanisms [32, 45]. In particular, the biological interpretation of the limit dynamics of automata networks matches experimental results [36, 5, 24, 46].

One can readily observe that distributed models of computation are highly sensitive to variations in the update schedule among its entities. Regarding Boolean automata networks, despite the fact that fixed points obtained under the parallel update mode are also fixed points for any other update schedule [25], specific update modes may generate additional fixed points [18, 39]. Limit cycles are also known to greatly depend on the update mode [17, 28, 10, 27, 11].

In this work, we propose to address a new family of update schedules, namely the block-parallel update modes, which are motivated by the discovery of the

importance of chromatin dynamics in regulatory networks [18]. Indeed, it fits our current understanding of the temporality of mRNA transcriptional machinery [29, 12, 30, 21]. Block-parallel update modes permit local update repetitions, opening new doors towards the phenomenological modelling in systems biology. Their novel features challenge intuitions erected upon classical block-sequential ones, and our objective is to build solid theoretical foundations for their study.

On the one hand, we study the combinatorics of block-parallel modes, and provide countings and enumeration algorithms for various meaningful equivalence relations. This is the grounding necessary to perform efficient numerical simulations. On the other hand, we analyse the computational cost of standard decision problems one may be willing to answer on the dynamics of Boolean automata networks with these new update schedules (related ot fixed points, limit cycles, reachability, *etc*). While block-parallel update modes seem to be more expressive, in particular regarding the limit dynamics, this gain of expressivity comes at a high cost in terms of simulation. Indeed, most problems traditionally NP-complete become PSPACE-complete, however there are notable exceptions.

In Section 2, we present the main definitions and notations. This paper is divided in two main sections, the first one focused on the counting and enumeration of block-parallel update modes and the second on complexity issues. The first part, Section 3, is split into five subsections. Subsection 3.1 serves as a sort of introduction, by dealing with the intersection between block-sequential and block-parallel update modes. The three following subsections each present a subset of block-parallel update modes, by giving a formula for counting the elements of this subset, and an algorithm to enumerate them. Subsection 3.2 deals with the whole set of block-parallel update modes, Subsection 3.3 with the updates schedules up to dynamical equality, and Subsection 3.4 with the update schedules up to dynamical isomorphism on the limit set, which is the one we focused more on. The last subsection of this part presents the results from our implementations of the aforementioned algorithms. Section 4 exposes our results regarding complexity problems involving block-parallel update schedules. In Subsection 4.2, we first characterize classical problems on computing images, preimages, fixed points and limit cycles : they all jump from NP (under block-sequential update modes) to PSPACE (under block-parallel update modes). In Subsection 4.3, we then prove a general bound on the recognition of functional subdynamics. Regarding global properties, recognizing bijective dynamics remains coNP-complete, and recognizing constant dynamics becomes PSPACE-complete. The case of identity recognition is much subtler, and we provide three incomparable bounds: a trivial coNP-hardness one, a tough ModP-hardness, and a $\mathsf{FP}^{\mathsf{PSPACE}}$-completeness result derived from the recent literature. Finally, we summarize our results and expose some perspectives in Section 5.

2. Definitions and state of the art

We denote the set of integers by $[\![n]\!] = \{0, \dots, n-1\}$, the Booleans by $\mathbb{B} = \{0, 1\}$, the i-th component of a vector $x \in \mathbb{B}^n$ by $x_i \in \mathbb{B}$, and the restriction of x to domain $I \subset [\![n]\!]$ by $x_I \in \mathbb{B}^{|I|}$. Let e_i be the i-th base vector, and $\forall x, y \in \mathbb{B}^n$, let $x + y$ denote the bitwise addition modulo two. Let σ^i denote the circular-shift of order $i \in \mathbb{Z}$ on sequences (shifting the element at position 0 towards position i). For two graphs $G = (V(G), A(G))$ and $H = (V(H), A(H))$, we denote by $G \sim H$ when they are isomorphic, i.e., when there is a bijection $\pi : V(G) \to V(H)$ such that $(x, y) \in A(G) \iff (\pi(x), \pi(y)) \in A(H)$. We denote by $G \sqsubset H$ when G is a subgraph of H, i.e., when G' such that $G' \sim G$ can be obtained from H by vertex and arc deletions.

Boolean automata network A *Boolean automata network* (BAN) is a discrete dynamical system on $\mathbb{B}^n$. A configuration $x \in \mathbb{B}^n$ associates to each of the n automata among $[\![n]\!]$ a Boolean state among $\mathbb{B}$. The individual dynamics of a each automaton $i \in [\![n]\!]$ is described by a local function $f_i : \mathbb{B}^n \to \mathbb{B}$ giving its new state according to the current configuration. To get a dynamics, one needs to settle the order in which the automata update their state by application of their local function. That is, an *update schedule* must be given. The most basic is the parallel update schedule, where all automata update their state synchronously at each step, formally as $f : \mathbb{B}^n \to \mathbb{B}^n$ defined by $\forall x \in \mathbb{B}^n : f(x) = (f_0(x), f_1(x), \dots, f_{n-1}(x))$. In this work, we concentrate on the block-parallel update schedule, motivated by the biological context of gene regulatory networks, where each automaton is a gene and the dynamics give clues on cell phenotypes. Not all automata will be update simultaneously as in the parallel update mode. They will instead be grouped by subsets. For simplicity in defining the local functions of a BAN, we extend the $f_i : \mathbb{B}^n \to \mathbb{B}$ notation to subsets $I \subseteq [\![n]\!]$ as $f_I : \mathbb{B}^n \to \mathbb{B}^{|I|}$. We also denote $f_{(I)} : \mathbb{B}^n \to \mathbb{B}^n$ the update of automata from subset I, defined as:

$$\forall i \in [\![n]\!] : f_{(I)}(x)_i = \begin{cases} f_i(x) & \text{if } i \in I \\ x_i & \text{otherwise.} \end{cases}$$

Block-sequential update schedule A *block-sequential* update schedule is an *ordered partition* of $[\![n]\!]$, given as a sequence of subsets $(W_i)_{i \in [\![\ell]\!]}$ where $W_i \subseteq [\![n]\!]$ is a *block*. The automata within a block are updated simultaneously, and the blocks are updated sequentially. During one iteration (*step*) of the network, the state of each automaton is updated exactly once. The update of each block is called a *substep*. This update mode received great attention on many aspects. The concept of the *update digraph* is introduced in [8] and characterized in [7] to capture equivalence classes of block-sequential update schedules (leading to the same dynamics). Conversions between block-sequential and parallel update

schedules are investigated in [41] (how to parallelize a block-sequential update schedule), [27] (the preservation of cycles throughout the parallelization process), and [14] (the cost of sequentialization of a parallel update schedule).

Block-parallel update schedule A *block-parallel* update schedule is a *partitioned order* of $[\![n]\!]$, given as a set of subsets $\mu = \{S_k\}_{k\in[\![s]\!]}$ where $S_k = (i_0^k, \dots, i_{n_k-1}^k)$ is a sequence of $n_k > 0$ elements of $[\![n]\!]$ for all $k \in [\![s]\!]$, called an *o-block* (shortcut for *ordered-block*). Each automaton appears in exactly one o-block. It follows an idea dual to the block-sequential update mode: the automata within an o-block are updated sequentially, and the o-blocks are updated simultaneously. The o-block sequences are taken circularly at each substep, until we reach the end of each o-block simultaneously (which happens after the least common multiple (lcm) of their sizes). The set of block-parallel update modes of size n is denoted BP_n. Formally, the update of f under $\mu \in \mathsf{BP}_n$ is given by $f_{\{\mu\}} : \mathbb{B}^n \to \mathbb{B}^n$ defined, with $\ell = \mathrm{lcm}(n_1, \dots, n_s)$, as $f_{\{\mu\}}(x) = f_{(W_{\ell-1})} \circ \cdots \circ f_{(W_1)} \circ f_{(W_0)}(x)$, where for all $i \in [\![\ell]\!]$ we define $W_i = \{i_i^k \bmod n_k \mid k \in [s]\}$. In order to compute the set of automata updated at each substep, it is possible to convert a block-parallel update schedule into a sequence of blocks of length ℓ (which is usually not a block-sequential update schedule, because repetitions of automaton update may appear). We defined this map as φ:

$$\varphi(\{S_k\}_{k\in[\![s]\!]}) = (W_i)_{i\in[\![\ell]\!]} \text{ with } W_i = \{i_i^k \bmod n_k \mid k \in [s]\}.$$

An example is given on Figure 1. The parallel update schedule corresponds to the block-parallel update schedule $\mu_{\mathtt{par}} = \{(i) \mid i \in [\![n]\!]\} \in \mathsf{BP}_n$, with $\varphi(\mu_{\mathtt{par}}) = ([\![n]\!])$, i.e., a single block containing all automata is updated at each step (there is only one substep).

Block-parallel update schedules have been introduced in [18], motivated by applications to gene regulatory networks, and their ability to generate new stable configurations (compared to block-sequential update schedules).

Fixed point and limit cycle A BAN f of size n under block-parallel update schedule $\mu \in \mathsf{BP}_n$ defines a deterministic discrete dynamical system $f_{\{\mu\}}$ on configuration space $\mathbb{B}^n$. Since the space is finite, the orbit of any configuration is ultimately periodic. For $p \geq 1$, a sequence of configurations $x^0, \dots, x^{p-1}$ is a *limit cycle* of length p when $\forall i \in [\![p]\!] : f_{\{\mu\}}(x^i) = x^{i+1 \bmod p}$. For $p = 1$ we call $x \in \mathbb{B}^n$ such that $f_{\{\mu\}}(x) = x$ a *fixed point*.

Complexity To be given as input to a decision problem, a BAN is encoded as a tuple of n Boolean circuits, one for each local function $f_i : \mathbb{B}^N \to \mathbb{B}$ for $i \in [\![n]\!]$. This encoding can be seen as Boolean formulas for each automaton, and easily implements high-level descriptions with if-then-else statements (used intensively in our constructions).

$$f(x) : \begin{cases} f_0(x) &= x_0 \wedge x_1 \\ f_1(x) &= \neg x_0 \vee (x_1 \wedge \neg x_2) \\ f_2(x) &= \neg x_1 \vee x_2 \end{cases}$$

$$\mu = \{(0), (1, 2)\}$$
$$\varphi(\mu) = (\{0, 1\}, \{0, 2\})$$

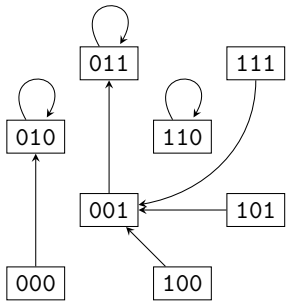

Figure 1: Example of an automata network of size $n = 3$ with a block-parallel update mode $\mu \in \mathsf{BP}_n$. Local functions (upper left), conversion of μ to a sequence of blocks (lower left), and dynamics of $f_{\{\mu\}}$ on configuration space $\mathbb{B}^3$ (right). One step is composed of two substeps: the first substep updates the block $\{0, 1\}$, the second substep updates the block $\{0, 2\}$. As an example, in computing the image of configuration `111`, the first substep (update of automata 0 and 1) gives `101`, and the second substep (update of automata 0 and 2) gives `001`.

The computational complexity of finite discrete dynamical systems has been explored on the related models of finite cellular automata [43] and reaction networks [19]. Regarding automata networks, fixed points received early attention in [6] and [22], with existence problems complete for NP. Because of the fixed point invariance for block-sequential update schedules [42], the focus switched to limit cycles [11, 13], with problems reaching the second level of the polynomial hierarchy. The interplay of different update schedules has been investigated in [11]. Finaly, let us mention the general complexity lower bounds, established for any first-order question on the dynamics, under the parallel update schedule [23].

3. Counting and enumerating block-parallel update modes

For the rest of this section, let $p(n)$ denote the number of integer partitions of n (multisets of integers summing to n), let $d(i)$ be the maximal part size in the i-th partition of n, let $m(i, j)$ be the multiplicity of the part of size j in the i-th partition of n. As an example, let $n = 31$ and assume the i-th partition is $(2, 2, 3, 3, 3, 3, 5, 5, 5)$, we have $d(i) = 5$ and $m(i, 1) = 0$, $m(i, 2) = 2$, $m(i, 3) = 4$, $m(i, 4) = 0$, $m(i, 5) = 3$. A partition will be the support of a partitioned order, where each part is an o-block. In our example, we can have:

$$\{(0, 1), (2, 3), (4, 5, 6), (7, 8, 9), (10, 11, 12), (13, 14, 15),$$
$$(16, 17, 18, 19, 20), (21, 22, 23, 24, 25), (26, 27, 28, 29, 30)\},$$

and we picture it as the following *matrix-representation*:

$$\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \end{pmatrix} \begin{pmatrix} 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \\ 26 & 27 & 28 & 29 & 30 \end{pmatrix}.$$

We call *matrices* the elements of size $j \cdot m(i,j)$ and denote them $M_1, \ldots, M_{d(i)}$, where M_j has $m(i,j)$ *rows* and j *columns* (M_j is empty when $m(i,j) = 0$). The partition defines the matrices' dimensions, and each row is an o-block.

For the comparison, the block-sequential update modes (ordered partitions of $[\![n]\!]$) are given by the ordered Bell numbers, sequence A000670 of OEIS [2, 38]. A closed formula for it is:

$$|\mathsf{BS}_n| = \sum_{i=1}^{p(n)} \frac{n!}{\prod_{j=1}^{d(i)} (j!)^{m(i,j)}} \cdot \frac{\left(\sum_{j=1}^{d(i)} m(i,j)\right)!}{\prod_{j=1}^{d(i)} m(i,j)!}.$$

Intuitively, an ordered partition of n gives a support to construct a block-sequential update mode: place the elements of $[\![n]\!]$ up to permutation within the blocks. This is the left fraction: $n!$ divided by $j!$ for each block of size j, taking into account multiplicities. The right fraction corrects the count because we sum on $p(n)$ the (unordered) partitions of n: each partition of n can give rise to different ordered partitions of n, by ordering all blocks (numerator, where the sum of multiplicities is the number of blocks) up to permutation within blocks of the same size which have no effect (denominator). The first ten terms are ($n = 1$ onward):

$$1, 3, 13, 75, 541, 4683, 47293, 545835, 7087261, 102247563.$$

3.1. Intersection of block-sequential and block-parallel modes

In order to be able to compare block-sequential with block-parallel update modes, both of them will be written here under their sequence of blocks form (the classical form for block-sequential update modes and the rewritten form for block-parallel modes).

First, we know that $\varphi(\mathsf{BP}_n) \cap \mathsf{BS}_n$ is not empty, since it contains at least

$$\mu_{\mathsf{par}} = ([\![n]\!]) = \varphi(\{(0), (1), \ldots, (n-1)\}).$$

However, neither $\mathsf{BS}_n \subseteq \varphi(\mathsf{BP}_n)$ nor $\varphi(\mathsf{BP}_n) \subseteq \mathsf{BS}_n$ are true. Indeed, $\mu_s = (\{0,1\},\{2\}) \in \mathsf{BS}_3$ but $\mu_s \notin \varphi(\mathsf{BP}_3)$ since a block-parallel cannot have blocks of different sizes in its sequential form. Symmetrically, $\mu_p = \varphi(\{(1,2),(0)\}) = (\{0,1\},\{0,2\}) \in \mathsf{BP}_3$ but $\mu_p \notin \mathsf{BS}_3$ since automaton 0 is updated twice. Despite this, we can precisely define the intersection $\mathsf{BS}_n \cap \varphi(\mathsf{BP}_n)$.

Lemma 1. *Let μ be an update mode written as a sequence of blocks of elements in $[\![n]\!]$. Then $\mu \in (\mathsf{BS}_n \cap \varphi(\mathsf{BP}_n))$ if and only if μ is an ordered partition and all of μ's blocks are of the same size.*

Proof. Let $n \in \mathbb{N}$.

($\Longrightarrow$) Let $\mu \in (\mathsf{BS}_n \cap \varphi(\mathsf{BP}_n))$. Since $\mu \in \mathsf{BS}_n$, μ is an ordered partition. Furthermore, $\mu \in \varphi(\mathsf{BP}_n)$ so all the μ's blocks are of the same size.

($\Longleftarrow$) Let $\mu = (W_i)_{i\in[\![\ell]\!]}$ be an ordered partition of $[\![n]\!]$ with all its blocks having the same size, denoted by s. Since μ is an ordered partition, $\mu \in \mathsf{BS}_n$. For each $\ell \in [\![p]\!]$, we can number arbitrarily the elements of W_ℓ from 0 to $s-1$ as $W_\ell = \{W_\ell^0, \dots, W_\ell^{s-1}\}$. Now, let us define the set of sequences $\{S_k\}_{k\in[\![s]\!]}$ the following way: $\forall k \in [\![s]\!], S_k = \{W_\ell^k \mid \ell \in [\![p]\!]\}$. It is a partitioned order such that $\varphi(\{S_k\}_{k\in[\![s]\!]}) = \mu$, which means that $\mu \in \varphi(\mathsf{BP}_n)$. □

Corollary 1. *If $\mu \in \mathsf{BP}_n$ and is composed of s o-blocks of size p, then $\varphi(\mu) \in \mathsf{BS}_n$ and is composed of p blocks of size s.*

As a consequence of Lemma 1 and Corollary 1, given $n \in \mathbb{N}$, the set SEQ_n of sequential update modes such that every automaton is updated exactly once by step and only one automaton is updated by substep, is a subset of $(\mathsf{BS}_n \cap \varphi(\mathsf{BP}_n))$.

Moreover, we can state the following proposition which counts the number of sequences of blocks which belongs to both BS_n and $\varphi(\mathsf{BP}_n)$.

Proposition 1. *Given $n \in \mathbb{N}$, we have:*

$$|\mathsf{BS}_n \cap \varphi(\mathsf{BP}_n)| = \sum_{d|n} \frac{n!}{(\frac{n}{d}!)^d}.$$

Proof. The proof derives directly from the sequence A061095 of OEIS [3], which counts the *number of ways of dividing n labeled items into labeled boxes with an equal number of items in each box.* In our context, the "items" are the automata, and the "labeled boxes" are the blocks of the ordered partitions. □

3.2. Partitioned orders

A block-parallel update mode is given as a partitioned order, *i.e.* an (unordered) set of (ordered) sequences. This concept is recorded as sequence A000262 of OEIS [1], described as the *number of "sets of lists".* A nice closed formula for it is:

$$|\mathsf{BP}_n| = \sum_{i=1}^{p(n)} \frac{n!}{\prod_{j=1}^{d(i)} m(i,j)!}.$$

Intuitively, for each partition, fill all the matrices ($n!$ ways to place the elements of $[\![n]\!]$) up to permutation of the rows within each matrix (matrix M_j has $m(i,j)$

rows). Another closed formula is presented in Proposition 2. This formula is particularly useful to generate all the block-parallel update modes.

Proposition 2. *For any $n \geq 1$ we have:*

$$|\mathsf{BP}_n| = \sum_{i=1}^{p(n)} \prod_{j=1}^{d(i)} \binom{n - \sum_{k=1}^{j-1} k \cdot m(i,k)}{j \cdot m(i,j)} \cdot \frac{(j \cdot m(i,j))!}{m(i,j)!}.$$

Proof. Each partition is a support to generate different partitioned orders (sum on i), by considering all the combinations, for each matrix (product on j), of the ways to choose the $j \cdot m(i,j)$ elements of $[\![n]\!]$ it contains (binomial coefficient, chosen among the remaining elements), and all the ways to order them up to permutation of the rows (ratio of factorials). Observe that developing the binomial coefficients with $\binom{x}{y} = \frac{x!}{y! \cdot (x-y)!}$ gives

$$\prod_{j=1}^{d(i)} \binom{n - \sum_k}{j \cdot m(i,j)} \cdot (j \cdot m(i,j))! = \prod_{j=1}^{d(i)} \frac{(n - \sum_k)!}{(n - \sum_k - j \cdot m(i,j))!} = \frac{n!}{0!} = n!,$$

where $\sum_k$ is a shorthand for $\sum_{k=1}^{j-1} k \cdot m(i,k)$, which leads to retrieve the OEIS formula. □

The first ten terms are ($n = 1$ onward):

$$1, 3, 13, 73, 501, 4051, 37633, 394353, 4596553, 58941091.$$

The formula from Proposition 2 can give us an algorithm to enumerate the partitioned orders of size n. For each partition i of n, it enumerates all the block-parallel update modes as a list of matrices as presented in the introduction of this section, with the matrices being filled one-by-one (the product on j in the formula). For each value of j, we choose a set of $j \cdot m(i,j)$ elements of $[\![n]\!]$ (that haven't been placed in a previous matrix) to put in matrix M_j. We then enumerate all the different ways to agence these numbers, up to permutation of the rows.

3.3. Partitioned orders up to dynamical equality

As for block-sequential update modes, given an AN f and two block-parallel update modes μ and μ', the dynamics of f under μ can be the same as that of f under μ'. To go further, in the framework of block-parallel update modes, there exist pairs of update modes μ, μ' such that for any AN f, the dynamics $f_{\{\mu\}}$ is the exact same as $f_{\{\mu'\}}$. As a consequence, in order to perform exhaustive searches among the possible dynamics, it is not necessary to generate all of them. We formalize this with the following equivalence relation.

Definition 1. *For $\mu, \mu' \in \mathsf{BP}_n$, we denote $\mu \equiv_0 \mu'$ when $\varphi(\mu) = \varphi(\mu')$.*

The following Lemma shows that this equivalence relation is necessary and sufficient in the general case of ANs of size n.

Lemma 2. *For any $\mu, \mu' \in \mathsf{BP}_n$, we have $\mu \equiv_0 \mu' \iff \forall f : X \to X, f_{\{\mu\}} = f_{\{\mu'\}}$.*

Proof. Let μ and μ' be two block-parallel update modes of BP_n.

($\Longrightarrow$) Let us consider that $\mu \equiv_0 \mu'$, and let $f : X \to X$ be an AN. Then, we have $f_{\{\mu\}} = f_{(\varphi(\mu))} = f_{(\varphi(\mu'))} = f_{\{\mu'\}}$.

($\Longleftarrow$) Let us consider that $\forall f : X \to X, f_{\{\mu\}} = f_{\{\mu'\}}$. Let us assume for the sake of contradiction that $\varphi(\mu) \neq \varphi(\mu')$. For ease of reading, we will denote as $t_{\mu,i}$ the substep at which automaton i is updated for the first time with update mode μ. Then, there is a pair of automata (i, j) such that $t_{\mu,i} \leq t_{\mu,j}$, but $t_{\mu',i} > t_{\mu',j}$. Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a Boolean AN such that $f(x)_i = x_i \vee x_j$ and $f(x)_j = x_i$, and $x \in \mathbb{B}^n$ such that $x_i = 0$ and $x_j = 1$. We will compare $f_{\{\mu\}}(x)_i$ and $f_{\{\mu'\}}(x)_i$, in order to prove a contradiction. Let us apply $f_{\{\mu\}}$ to x. Before step $t_{\mu,i}$ the value of automaton i is still 0 and, most importantly, since $t_{\mu,i} \leq t_{\mu,j}$, the value of j is still 1. This means that right after step $t_{\mu,i}$, the value of automaton i is 1, and will not change afterwards. Thus, we have $f_{\{\mu\}}(x)_i = 1$. Let us now apply $f_{\{\mu'\}}$ to x. This time, $t_{\mu',i} > t_{\mu',j}$, which means that automaton j is updated first and takes the value of automaton i at the time, which is 0 since it has not been updated yet. Afterwards, neither automata will change value since $0 \vee 0$ is still 0. This means that $f_{\{\mu'\}}(x)_i = 0$. Thus, we have $f_{\{\mu\}} \neq f_{\{\mu'\}}$, which contradicts our earlier hypothesis. □

Let $\mathsf{BP}_n^0 = \mathsf{BP}_n / \equiv_0$ denote the corresponding quotient set, *i.e.* the set of block-parallel update modes to generate for computer analysis of all the possible dynamics in the general case of ANs of size n.

Theorem 1. *For any $n \geq 1$, we have:*

$$|\mathsf{BP}_n^0| = \sum_{i=1}^{p(n)} \frac{n!}{\prod_{j=1}^{d(i)} (m(i,j)!)^j} \tag{1}$$

$$= \sum_{i=1}^{p(n)} \prod_{j=1}^{d(i)} \prod_{\ell=1}^{j} \binom{n - \sum_{k=1}^{j-1} k \cdot m(i,k) - (\ell - 1) \cdot m(i,j)}{m(i,j)} \tag{2}$$

$$= \sum_{i=1}^{p(n)} \prod_{j=1}^{d(i)} \left(\binom{n - \sum_{k=1}^{j-1} k \cdot m(i,k)}{j \cdot m(i,j)} \cdot \prod_{\ell=1}^{j} \binom{(j - \ell + 1) \cdot m(i,j)}{m(i,j)} \right). \tag{3}$$

Proof. $|\mathsf{BP}^0_n|$ can be viewed as three distinct formulas. We will show here that Formula 1 counts $|\mathsf{BP}^0_n|$, and the proof that these three formulas are equal will be in Appendix A.

For any pair $\mu, \mu' \in \mathsf{BP}_n$, we have $\mu \equiv_0 \mu'$ if and only if their matrix-representations are the same up to a permutation of the elements within columns (the number of equivalence classes is then counted by Formula 1). In the definition of φ, each block is a set constructed by taking one element from each o-block. Given that n_k in the definition of φ corresponds to j in the statement of the theorem, one matrix corresponds to all the o-blocks that have the same size n_k. Hence, the $\ell \mod n_k$ operations in the definition of φ amounts to considering the elements of these o-blocks which are in the same column in their matrix representation. Since blocks are unordered, the result follows. □

The first ten terms of the sequence $(|\mathsf{BP}^0_n|)_{n\geq 1}$ are:

$$1, 3, 13, 67, 471, 3591, 33573, 329043, 3919387, 47827093.$$

They match the sequence A182666 of OEIS [4], and the next lemma proves that they are indeed the same sequence (defined by its exponential generating function on OEIS). The *exponential generating function of a sequence* $(a_n)_{n\in\mathbb{N}}$ is $f(x) = \sum_{n\geq 0} a_n \frac{x^n}{n!}$.

Lemma 3. *The exponential generating function of* $(|\mathsf{BP}^0_n|)_{n\in\mathbb{N}}$ *is* $\prod_{j\geq 1} \sum_{k\geq 0} \left(\frac{x^k}{k!}\right)^j$.

The proof of this lemma can be found in Appendix A.

As with the previous enumerating formula, we can also extrapolate an enumeration algorithm from this one. It starts out similar to the previous one, but differs once the contents of M_j are chosen. In the previous algorithm, we needed to enumerate every matrix up to permutation of the rows. This time, we need to enumerate every matrix up to permutation within the columns. This means that, for each column of the matrix, we just choose the content separately, similarly to how we chose the content of each matrix.

3.4. Partitioned orders up to dynamical isomorphism on the limit set

The following equivalence relation defined over block-parallel update modes turns out to capture exactly the notion of having isomorphic limit dynamics. It is analogous to $\equiv_0$, except that a circular shift of order i may be applied on the sequences of blocks.

Definition 2. *For* $\mu, \mu' \in \mathsf{BP}_n$, *we denote* $\mu \equiv_\star \mu'$ *when* $\varphi(\mu) = \sigma^i(\varphi(\mu'))$ *for some* $i \in [\![|\varphi(\mu')|]\!]$ *called the* shift.

Remark 1. Note that $\mu \equiv_0 \mu'$ corresponds to the particular case $i = 0$ of $\equiv_\star$. Thus, $\mu \equiv_0 \mu' \implies \mu \equiv_\star \mu'$.

Notation 1. *Given $f_{\{\mu\}} : X \to X$, let $\Omega_{f_{\{\mu\}}} = \bigcap_{t\in\mathbb{N}} f_{\{\mu\}}(X)$ denote its* limit set *(abusing the notation of $f_{\{\mu\}}$ to sets of configurations), and $f^{\Omega}_{\{\mu\}} : \Omega_{f_{\{\mu\}}} \to \Omega_{f_{\{\mu\}}}$ its restriction to its limit set. Observe that, since the dynamics is deterministic, $f^{\Omega}_{\{\mu\}}$ is bijective.*

The following Lemma shows that, if one is generally interested in the limit behavior of ANs under block-parallel updates, then studying a representative from each equivalence class of the relation $\equiv_\star$ is necessary and sufficient to get the full spectrum of possible limit dynamics.

Lemma 4. *For any $\mu, \mu' \in \mathsf{BP}_n$, we have $\mu \equiv_\star \mu' \iff \forall f : X \to X, f^{\Omega}_{\{\mu\}} \sim f^{\Omega}_{\{\mu'\}}$.*

Proof. Let μ and μ' be two block-parallel update modes of BP_n.

($\Longrightarrow$) Let μ, μ' be such that $\mu \equiv_\star \mu'$ of shift $\hat{\imath} \in [\![p]\!]$, with $\varphi(\mu) = (W_i)_{i\in[\![\ell]\!]}$, $\varphi(\mu') = (W'_\ell)_{\ell\in[\![p]\!]}$ and $p = |\varphi(\mu)| = |\varphi(\mu')|$. It means that $\forall i \in [\![p]\!]$, we have $W'_i = W_{i+\hat{\imath} \mod p}$, and for any AN f, we deduce that $\pi = f_{(W_0,\dots,W_{\hat{\imath}-1})}$ is the desired isomorphism from $\Omega_{f_{\{\mu\}}}$ to $\Omega_{f_{\{\mu'\}}}$. Indeed, we have $f_{\{\mu\}}(x) = y$ if and only if $f_{\{\mu'\}}(\pi(x)) = \pi(y)$ because

$$f_{\{\mu'\}} \circ \pi = f_{(W_0,\dots,W_{\hat{\imath}-1},W'_0,\dots,W'_p)} = f_{(W'_{p-\hat{\imath}},\dots,W'_p)} \circ f_{\{\mu\}} = \pi \circ f_{\{\mu\}}.$$

Note that $\pi^{-1} = f^{(q-1)}_{\{\mu'\}} \circ f_{(W'_i \dots W'_{p-1})}$ with q the least common multiple of the limit cycle lengths, and $\pi^{-1} \circ \pi$ (*resp.* $\pi \circ \pi^{-1}$) is the identity on $\Omega_{f_{\{\mu\}}}$ (*resp.* $\Omega_{f_{\{\mu'\}}}$).

($\Longleftarrow$) We prove the contrapositive, from $\mu \not\equiv_\star \mu'$, by case disjunction.

(1) If in $\varphi(\mu)$ and $\varphi(\mu')$, there is an automaton $\hat{\imath}$ which is not updated the same number of times α and α' in μ and μ' respectively, then we assume without loss of generality that $\alpha > \alpha'$ and consider the AN f such that:

- $X_{\hat{\imath}} = [\![\alpha]\!]$ and $X_i = \{0\}$ for all $i \neq \hat{\imath}$; and
- $f_{\hat{\imath}}(x) = (x_{\hat{\imath}} + 1) \mod \alpha$ and $f_i(x) = x_i$ for all $i \neq \hat{\imath}$.

It follows that $f^{\Omega}_{\{\mu\}}$ has only fixed points since $+1 \mod \alpha$ is applied α times, whereas $f^{\Omega}_{\{\mu'\}}$ has no fixed point because $\alpha' < \alpha$. We conclude that $f^{\Omega}_{\{\mu\}} \not\sim f^{\Omega}_{\{\mu'\}}$.

(2) If in $\varphi(\mu)$ and $\varphi(\mu')$, all the automata are updated the same number of times, then the transformation from μ to μ' is a permutation on $[\![n]\!]$ which preserves the matrices of their matrix representations (meaning that any $i \in [\![n]\!]$ is in an o-block of the same size in μ and μ', which also implies that μ and μ' are constructed from the same partition of n). Then we consider subcases.

(2.1) If one matrix of μ' is not obtained by a permutation of the columns from μ, then there is a pair of automata $\hat{\imath}, \hat{\jmath}$ that appears in the k-th block of $\varphi(\mu)$ for some k, and does not appear in any block of $\varphi(\mu')$. Indeed, one can take $\hat{\imath}, \hat{\jmath}$ to be in the same column in μ but in different columns in μ'. Let S be the o-block of $\hat{\imath}$ and S' be the o-block of $\hat{\jmath}$. Let p denote the least common multiple of o-blocks sizes in both μ and μ'. In this case we consider the AN f such that:

- $X_{\hat{\imath}} = \mathbb{B} \times [\![\frac{p}{|S|}]\!]$, $X_{\hat{\jmath}} = \mathbb{B} \times [\![\frac{p}{|S'|}]\!]$, and $X_i = \{0\}$ for all $i \notin \{\hat{\imath}, \hat{\jmath}\}$. Given $x \in X$, we denote $x_{\hat{\imath}} = (x^b_{\hat{\imath}}, x^\ell_{\hat{\imath}})$ the state of $\hat{\imath}$ (and analogously for $\hat{\jmath}$); and
- $f_{\hat{\imath}}(x) = \begin{cases} (x^b_{\hat{\jmath}}, x^\ell_{\hat{\imath}} + 1 \mod \frac{p}{|S|}) & \text{if } x^\ell_{\hat{\imath}} = 0 \\ (x^b_{\hat{\imath}}, x^\ell_{\hat{\imath}} + 1 \mod \frac{p}{|S|}) & \text{otherwise} \end{cases}$,
  $f_{\hat{\jmath}}(x) = \begin{cases} (x^b_{\hat{\imath}}, x^\ell_{\hat{\jmath}} + 1 \mod \frac{p}{|S'|}) & \text{if } x^\ell_{\hat{\jmath}} = 0 \\ (x^b_{\hat{\jmath}}, x^\ell_{\hat{\jmath}} + 1 \mod \frac{p}{|S'|}) & \text{otherwise} \end{cases}$, and
  $f_i(x) = x_i$ for all $i \notin \{\hat{\imath}, \hat{\jmath}\}$.

Note that $\hat{\imath}$ (resp. $\hat{\jmath}$) is updated $\frac{p}{|S|}$ (resp. $\frac{p}{|S'|}$) times during a step in both μ and μ'. Therefore for any $x \in X$, its two images under μ and μ' verify $f_{\{\mu\}}(x)^\ell_{\hat{\imath}} = f_{\{\mu'\}}(x)^\ell_{\hat{\imath}} = x^\ell_{\hat{\imath}}$ (and analogously for $\hat{\jmath}$). Thus for the evolution of the states of $\hat{\imath}$ and $\hat{\jmath}$ during a step, the second element is fixed and only the first element (in $\mathbb{B}$) may change. We split X into $X^= = \{x \in X \mid x^b_{\hat{\imath}} = x^b_{\hat{\jmath}}\}$ and $X^\neq = \{x \in X \mid x^b_{\hat{\imath}} \neq x^b_{\hat{\jmath}}\}$, and observe the following facts by the definition of $f_{\hat{\imath}}$ and $f_{\hat{\jmath}}$:

- Under μ and μ', all the elements of $X^=$ are fixed points (indeed, only $x^b_{\hat{\imath}}$ and $x^b_{\hat{\jmath}}$ may evolve by copying the other).
- Under μ, let m, m' be the respective number of times $\hat{\imath}, \hat{\jmath}$ have been updated prior to the k-th block of $\varphi(\mu)$ in which they are updated synchronously. Consider the configurations $x, y \in X^\neq$ with $x_{\hat{\imath}} = (0, -m \mod \frac{p}{|S|})$, $x_{\hat{\jmath}} = (1, -m' \mod \frac{p}{|S'|})$, $y_{\hat{\imath}} = (1, -m \mod \frac{p}{|S|})$ and $y_{\hat{\jmath}} = (0, -m' \mod \frac{p}{|S'|})$. It holds that $f_{\{\mu\}}(x) = y$ and $f_{\{\mu\}}(y) = x$, because $x^b_{\hat{\imath}}$ and $x^b_{\hat{\jmath}}$ are exchanged synchronously when $x^\ell_{\hat{\imath}} = x^\ell_{\hat{\jmath}} = 0$ during the k-th block of $\varphi(\mu)$, and are not exchanged again during that step by the choice of the modulo. Hence, $f^\Omega_{\{\mu\}}$ has a limit cycle of length two.
- Under μ', for any $x \in X^\neq$, there is a substep with $x^\ell_{\hat{\imath}} = 0$ and there is a substep with $x^\ell_{\hat{\jmath}} = 0$, but they are not the same substep (because $\hat{\imath}$ and $\hat{\jmath}$ are never synchronized in μ'). As a consequence, $x^b_{\hat{\imath}}$ and $x^b_{\hat{\jmath}}$ will end up having the same value (the first to be updated copies the bit from the second, then the second copies its own bit),

i.e. $f_{\{\mu'\}}(x) \in X^{=}$, and therefore $f^{\Omega}_{\{\mu'\}}$ has only fixed points.

We conclude in this case that $f^{\Omega}_{\{\mu\}} \not\sim f^{\Omega}_{\{\mu'\}}$, because one has a limit cycle of length two, whereas the other has only fixed points.

(2.2) If the permutation preserves the columns within the matrices (meaning that the automata within the same column in μ are also in the same column in μ'), then we consider two last subcases:

(2.2.1) Moreover, if the permutation of some matrix is not circular (meaning that there are three columns which are not in the same relative order in μ and μ'), then there are three automata $\hat{\imath}$, $\hat{\jmath}$ and $\hat{k}$ in the same matrix such that in μ, automaton $\hat{\imath}$ is updated first, then $\hat{\jmath}$, then $\hat{k}$; whereas in μ', automaton $\hat{\imath}$ is updated first, then $\hat{k}$, then $\hat{\jmath}$. Let us consider the automata network f such that:

- $X = \mathbb{B}^n$;
- $f_{\hat{\imath}}(x) = x_{\hat{k}}$, $f_{\hat{\jmath}}(x) = x_{\hat{\imath}}$ and $f_{\hat{k}}(x) = x_{\hat{\jmath}}$; and
- $f_i(x) = x_i$ if $i \notin \{\hat{\imath}, \hat{\jmath}, \hat{k}\}$.

If the three automata are updated in the order $\hat{\imath}$ then $\hat{\jmath}$ then $\hat{k}$, as it is the case with μ, then after any update, they will all have taken the same value. It implies that $f_{\{\mu\}}$ has only fixed points, precisely the set $P = \{x \in \mathbb{B}^n \mid x_{\hat{\imath}} = x_{\hat{\jmath}} = x_{\hat{k}}\}$.

If they are updated in the order $\hat{\imath}$ then $\hat{k}$ then $\hat{\jmath}$, as with μ', however, the situation is a bit more complex. We consider two cases, according to the number of times they are updated during a period (recall that since they belong to the same matrix, they are updated repeatedly in the same order during the substeps):

- If they are updated an odd number of times each, then automata $\hat{\imath}$ and $\hat{\jmath}$ will take the initial value of automaton $\hat{k}$, and automaton $\hat{k}$ will take the initial value of automaton $\hat{\jmath}$. In this case, $f^{\Omega}_{\{\mu'\}}$ has the fixed points P and limit cycles of length two.
- If they are updated an even number of times each, then the reverse will occur: automata $\hat{\imath}$ and $\hat{\jmath}$ will take the initial value of automaton $\hat{\jmath}$, and automaton $\hat{k}$ will keep its initial value. In this case, $f^{\Omega}_{\{\mu'\}}$ has the fixed points $Q = \{x \in \mathbb{B}^n \mid x_{\hat{\imath}} = x_{\hat{\jmath}}\}$ which strictly contains P (*i.e.* $P \subseteq Q$ and $Q \setminus P \neq \emptyset$).

In both cases $f^{\Omega}_{\{\mu'\}}$ has more than the fixed points P in its limit set, hence we conclude that $f^{\Omega}_{\{\mu\}} \not\sim f^{\Omega}_{\{\mu'\}}$.

(2.2.2) Moreover, if the permutation of all matrices is circular, then we first observe that when $\varphi(\mu)$ and $\varphi(\mu')$ have one block in common, they have all blocks in common (because of the circular nature of permutations), *i.e.* $\mu \equiv_\star \mu'$. Thus, under our hypothesis, we deduce

that $\varphi(\mu)$ and $\varphi(\mu')$ have no block in common. As a consequence, there exist automata $\hat{\imath}, \hat{\jmath}$ with the property from case (2.1), namely synchronized in a block of $\varphi(\mu)$ but never synchronized in any block of $\varphi(\mu')$, and the same construction terminates this proof.

□

Let $\mathsf{BP}_n^\star = \mathsf{BP}_n / \equiv_\star$ denote the corresponding quotient set.

Theorem 2. *Let* $lcm(i) = lcm(\{j \in [\![1, d(i)]\!] \mid m(i,j) \geq 1\})$. *For any* $n \geq 1$, *we have:*

$$|\mathsf{BP}_n^\star| = \sum_{i=1}^{p(n)} \frac{n!}{\prod_{j=1}^{d(i)} (m(i,j)!)^j} \cdot \frac{1}{lcm(i)}.$$

Proof. Let $\mu, \mu' \in \mathsf{BP}_n$ two update modes such that $\mu \equiv \mu'$. Then their sequential forms are of the same length, and each automaton appears the same number of times in both of them. This means that, if an automaton is in an o-block of size k in μ's partitioned order form, then it is also in an o-block of the same size in μ''s. We deduce that two update modes of size n can only be equivalent as defined in Definition 2 if they are generated from the same partition of n.

Let $\mu \in \mathsf{BP}_n^0$, generated from partition i of n. Then $\varphi(\mu)$ is of length $\mathrm{lcm}(i)$. Since no two elements of BP_n^0 have the same block-sequential form, the equivalence class of μ in BP_n^0 contains exactly $\mathrm{lcm}(i)$ elements, all generated from the same partition i (all the blocks of $\varphi(\mu)$ are different). Thus, the number of elements of $\mathsf{BP}_n^\star$ generated from a partition i is the number of elements of BP_n^0 generated from partition i, divided by the number of elements in its equivalence class for $\mathsf{BP}_n^\star$, namely $\mathrm{lcm}(i)$. □

Remark 2. The formula for $|\mathsf{BP}_n^\star|$ can actually be obtained from any formula in Theorem 1 by multiplying by $\frac{1}{\mathrm{lcm}(i)}$ inside the sum on partitions (from $i = 1$ to $p(n)$).

While counting the elements of $\mathsf{BP}_n^\star$ was pretty straightforward, enumerating them by ensuring that no two partitioned orders are the same up to circular permutation of their block-sequential rewritings (Definition 2) is more challenging. This is performed by Algorithm 1. It works much like the one enumareting the elements of BP_n^0, except for the following differences. In the first function, `EnumBPiso`: right after choosing the partition, a list of coefficients $a[j]$ is determined, in a modified algorithm that computes $\mathrm{lcm}(i)$ inductively (lines 3-10). These coefficients are used in the second auxiliary function `EnumBlockIsoAux`, where the minimum min_j of the matrix M_j is forced to be in the first $a[j]$ columns of said matrix (lines 35-37, the condition is fulfilled when min_j has not been chosen within the $a[j] - 1$

Algorithm 1: Enumeration of $\mathsf{BP}_n^\star$

```
1  Function EnumBPiso(n):
2      foreach i ∈ partitions(n) do
3          a is a list of size d(i)
4          b ← 1
5          for j ← d(i) to 1 do
6              if m(i, j) > 0 then
7                  a[j] ← gcd(b, j)
8                  b ← lcm(b, j)
9              else
10                 a[j] ← j
11         EnumBPisoAux(n, i, 1, a)

12 Function EnumBPisoAux(n, i, j, a, M_1, ..., M_{j-1}):
13     if d(i) < j then
14         enumerate(M)
15         return
16     if m(i, j) > 0 then
17         foreach combination A of size j · m(i, j) among [[n]] \ ⋃_{k=1}^{j-1} M_k do
18             min_j ← min(A)
19             foreach M_j enumerated by EnumBlockIso(A, j, m(i, j), min_j, a[j]) do
20                 EnumBPisoAux(n, i, j + 1, a, M_1, ..., M_{j-1}, M_j)
21     else
22         EnumBPisoAux(n, i, j + 1, a, M_1, ..., M_{j-1}, ∅)

23 Function EnumBlockIso(A, j, m, min_j, a_j):
24     foreach C enumerated by EnumBlockIsoAux(A, m, min_j, a_j) do
25         for k ← 1 to m do
26             for ℓ ← 1 to j do
27                 M_j[k][ℓ] = C_ℓ[k]
28         enumerate(M_j)
29         return

30 Function EnumBlockIsoAux(A, j, m, min_j, a_j, C_1, ..., C_ℓ):
31     if A = ∅ then
32         enumerate(C)
33         return
34     else
35         if |A| = m · (j − a_j + 1) and min_j ∈ A then
36             foreach combination B of size m − 1 among A \ {min_j} do
37                 EnumBlockIsoAux(A \ (B ∪ {min_j}), m, min_j, a_j, C_1, ..., C_ℓ, (B ∪ {min_j}))
38         else
39             foreach combination B of size m among A do
40                 EnumBlockIsoAux(A \ B, m, min_j, a_j, C_1, ..., C_ℓ, B)
```

first columns, then it is placed in that column so only $m(i,j) - 1$ elements are chosen).

The proof of correction of Algorithm 1 can be found in Appendix A.

3.5. Implementations

Proof-of-concept Python implementations of the three enumeration algorithms mentioned are available on the following repository:

`https://framagit.org/leah.tapin/blockpargen`.

It is archived by Software Heritage at the following permalink:

https://archive.softwareheritage.org/browse/directory/
f1b4d83c854a4d042db5018de86b7f41ef312a07/?origin_url=
https://framagit.org/leah.tapin/blockpargen.

We have conducted numerical experiments on a standard laptop, presented on Figure 2.

4. Computational complexity under block-parallel updates

Computational complexity is important to anyone willing to use algorithmic tools in order to study discrete dynamical systems. Lower bounds inform on the best worst case time or space one can expect with an algorithm solving some problem. The n local functions of a BAN are encoded as Boolean circuits, which is a convenient formalism corresponding to the high level descriptions one usually employs. The update mode is given as a list of lists of integers, each of them being encoded either in unary or binary (this makes no difference, because the encoding of local functions already has a size greater than n).

In this section we characterize the computational complexity of typical problems arising in the framework of automata networks. We will see that almost all problems reach PSPACE-completeness. The intuition behind this fact is that the description of a block-parallel update mode may expend (through φ) to an exponential number of substeps, during which a linear bounded Turing machine may be simulated via iterations of a circuit. We first recall this folklore building block and present a general outline of our constructions (Subsection 4.1). Then we start with results on computing images, preimages, fixed points and limit cycles (Subsection 4.2), before studying reachability and global properties of the function $f_{\{\mu\}}$ computed by an automata network f under block-parallel update schedule μ (Subsection 4.3).

n	BP_n	BP_n^0	$\mathsf{BP}_n^\star$
1	1 -	1 -	1 -
2	3 -	3 -	2 -
3	13 -	13 -	6 -
4	73 -	67 -	24 -
5	501 -	471 -	120 -
6	4051 -	3591 -	795 -
7	37633 -	33573 0.103s	5565 -
8	394353 0.523s	329043 0.996s	46060 0.161s
9	4596553 6.17s	3919387 12.2s	454860 1.51s
10	58941091 1min24s	47827093 2min40s	4727835 16.3s
11	824073141 21min12s	663429603 38min31s	54223785 3min13s
12	12470162233 5h27min38s	9764977399 9h49min26s	734932121 45min09s

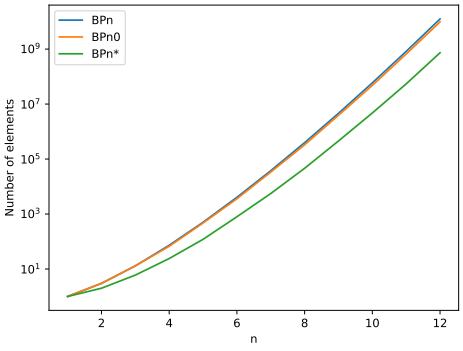

Figure 2: Numerical experiments of our Python implementation of the three algorithms on a standard laptop (processor Intel-Core$^{\text{TM}}$ i7 @ 2.80 GHz). For n from 1 to 12, the table (left) presents the size of BP_n, BP_n^0 and BP_n^* and running time to enumerate their elements (one representative of each equivalence class; a dash represents a time smaller than 0.1 second), and the graphics (right) depicts their respective sizes on a logarithmic scale. Observe that the sizes of BP_n and BP_n^0 are comparable, whereas an order of magnitude is gained with BP_n^*, which may be significant for advanced numerical experiments regarding limit dynamics under block-parallel udpate modes.

4.1. Outline of the PSPACE*-hardness constructions*

We will design polynomial time many-one reductions from the following PSPACE-complete decision problem, which appears for example in [26].

Iterated Circuit Value Problem (**Iter-CVP**) Input: a Boolean circuit $C : \mathbb{B}^n \to \mathbb{B}^n$, a configuration $x \in \mathbb{B}^n$, and $i \in [\![n]\!]$. Question: does $\exists t \in \mathbb{N} : C^t(x)_i = 1$?

Theorem 3 (folklore)**. Iter-CVP** *is* PSPACE*-complete.*

Before presenting the general outline of our constructions, we need a technical lemma related to the generation of primes (proof in Appendix A).

Lemma 5. *For all $n \geq 2$, a list of distinct prime integers $p_1, p_2, \ldots, p_{k_n}$ such that $2 \leq p_i < n^2$ and $2^n < \prod_{i=1}^{k_n} p_i < 2^{2n^2}$ can be computed in time $\mathcal{O}(n^2)$, with $k_n = \lfloor \frac{n^2}{2\ln(n)} \rfloor$.*

Our constructions of automata netwoks and block-parallel update schedules for the computational complexity lower bounds are based on the following.

Definition 3. *For any $n \geq 2$, let $p_1, p_2, \ldots, p_{k_n}$ be the k_n primes given by Lemma 5, and denote $q_j = \sum_{i=1}^{j} p_i$ their cumulative series for j from 0 to k_n. Define the automata network g_n on q_{k_n} automata $[\![q_{k_n}]\!]$ with constant* 0 *local functions, where the components are grouped in o-blocks of length p_i, that is with $\mu_n = \bigcup_{i \in [\![k_n]\!]} \{(q_i, q_i + 1, \ldots, q_{i+1} - 1)\}$.*

Lemma 6. *For any $n \geq 2$, one can compute g_n and μ_n in time $\mathcal{O}(n^4)$, and $|\varphi(\mu_n)| > 2^n$.*

Proof. The time bound comes from Lemma 5 and the fact that q_{k_n} is in $\mathcal{O}(n^4)$. The number of blocks in $\varphi(\mu_n)$ is the least common multiple of its o-block sizes, which is the product $\prod_{i=1}^{k_n} p_i$, hence from Lemma 5 we conclude that it is greater than 2^n. □

The general idea is now to add some automata to g_n and place them within singletons in μ_n, i.e., each of them in a new o-block of length 1. We propose an example implementing a binary counter on n bits.

Example 1. *Given $n \geq 2$, consider g_n and μ_n given by Lemma 6. Construct f from g_n by adding n Boolean components $\{q_{k_n}, \ldots, q_{k_n+n}\}$, whose local functions increment a binary counter on those n bits, until it freezes to $2^n - 1$ (all bits in state* 1*). Construct μ' from μ_n as $\mu' = \mu_n \cup \bigcup_{i \in [\![n]\!]} \{(q_{k_n} + i)\}$, so that the counter components are updated at each substep. Observe that the pair f, μ' can be still be computed from n in time $\mathcal{O}(n^4)$. Figure 3 illustrates an example of orbit for $n = 3$, and one can notice that $f_{\{\mu'\}}$ is a constant function sending any $x \in \mathbb{B}^n$ to $0^{q_{k_n}} 1^n$.*

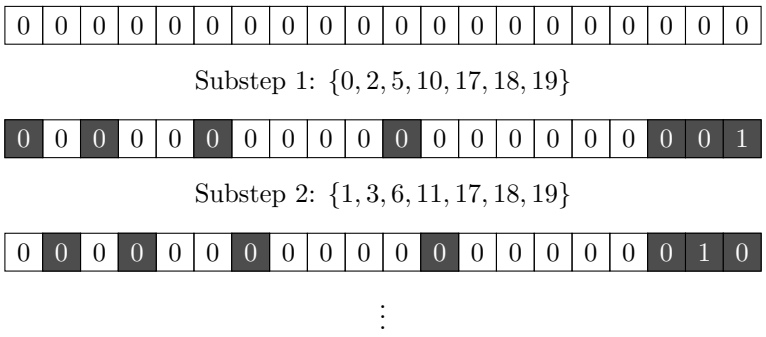

Figure 3: Substeps leading to the image of configuration $0^{q_{k_n}}010$ in $f_{\{\mu'\}}$ from Example 1 for $n = 3$ ($k_n = 4$ and $q_{k_n} = 2+3+5+7 = 17$). The last 3 bits implement a binary counter, freezing at 7 (111). Above each substep the block of updated automata is given.

Remark that we will prove complexity lower bounds by reduction from **Iter-CVP**, where n will be the number of inputs and outputs of the circuit to be iterated, hence the integer n itself will be encoded in unary. As a consequence, the construction of Example 1 is computed in polynomial time.

4.2. Images, preimages, fixed points and limit cycles

We start the study of the computational complexity of automata networks under block-parallel update schedules with the most basic problem of computing the image $f_{\{\mu\}}(x)$ of some configuration x through $f_{\{\mu\}}$ (i.e., one step of the evolution), which is already PSPACE-hard. We conduct this study as decision problems. It is actually hard to compute even a single bit of $f_{\{\mu\}}(x)$. The fixed point verification problem is a particular case of computing an image, which is still PSPACE-hard (unlike block-sequential update schedules for which this problem is in P). Recall that the encoding of μ (with integers in unary or binary) has no decisive influence on the input size, this latter being characterized by the circuits sizes and in particular their number of inputs, denoted n, which is encoded in unary.

Block-parallel step bit (**BP-Step-Bit**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i\in[\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$, $x \in \mathbb{B}^n$, $j \in [\![n]\!]$. Question: does $f_{\{\mu\}}(x)_j = 1$?

Block-parallel step (**BP-Step**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i\in[\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$, $x, y \in \mathbb{B}^n$. Question: does $f_{\{\mu\}}(x) = y$?

Block-parallel fixed point verification (**BP-Fixed-Point-Verif**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i\in[\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$, $x \in \mathbb{B}^n$. Question: does $f_{\{\mu\}}(x) = x$?

This first set of problems is related to the image of a given configuration x, which allows the reasonings to concentrate on the dynamics of substeps for that single configuration x, regardless of what happens for other configurations. Note that n will be the size of the **Iter-CVP** instance, while the size of the automata network will be $q_{k_n} + \ell' + n + 1$.

Theorem 4. BP-Step-Bit, BP-Step *and* **BP-Fixed-Point-Verif** *are* PSPACE*-complete.*

Proof. The problems **BP-Step-Bit**, **BP-Step** and **BP-Fixed-Point-Verif** are in PSPACE, with a simple algorithm obtaining $f_{\{\mu\}}(x)$ by computing the least common multiple of o-block sizes and then using a pointer for each block throughout the computation of that number of substeps (each substep evaluates local functions in polynomial time).

We give a single reduction for the hardness of **BP-Step-Bit**, **BP-Step** and **BP-Fixed-Point-Verif**, where we only need to consider the dynamics of the substeps starting from one configuration x. Given an instance of **Iter-CVP** with a circuit $C : \mathbb{B}^n \to \mathbb{B}^n$, a configuration $\tilde{x} \in \mathbb{B}^n$ and $i \in [\![n]\!]$, we apply Lemma 6 to construct g_n, μ_n on automata set $P = [\![q_{k_n}]\!]$. Automata from P have constant 0 local functions, and the number of substeps is $\ell = |\varphi(\mu_n)| > 2^n$ thanks to the prime's lcm. We define a BAN f by adding:

- $\ell' = \lceil \log_2(\ell) \rceil$ automata numbered $B = \{q_{k_n}, \dots, q_{k_n} + \ell' - 1\}$, implementing a counter that increments modulo ℓ at each substep, and remains fixed when x_B encodes an integer greater or equal to ℓ (case not considered in this proof);
- n automata numbered $D = \{q_{k_n} + \ell', \dots, q_{k_n} + \ell' + n - 1\}$, whose local functions iterate $C : \mathbb{B}^n \to \mathbb{B}^n$ while the counter is smaller than $\ell - 1$, and go to state $\tilde{x}$ when the counter reaches $\ell - 1$, i.e., with

$$f_D(x) = \begin{cases} C(x_D) & \text{if } x_B < \ell - 1, \\ \tilde{x} & \text{otherwise; and} \end{cases}$$

- 1 automaton numbered $R = \{q_{k_n} + \ell' + n\}$, whose local function

$$f_R(x) = x_R \vee x_{q_{k_n} + \ell' + i}$$

  records whether a state 1 appeared at automaton in relative position i within D.

We also add singletons to μ_n for each of these additional automata, by setting

$$\mu' = \mu_n \cup \bigcup_{j \in B \cup D \cup R} \{(j)\}.$$

Now, consider the dynamics of substeps in computing the image of configuration $x = 0^{q_{k_n}} 0^{\ell'} \tilde{x} 0$. During the first $\ell - 1$ substeps:

- automata P have constant $\mathtt{0}$ local function;
- automata B increment a counter from 0 to $\ell - 1$;
- automata D iterate circuit C from $\tilde{x}$; and
- automaton R records whether the i-th bit of D has been in state $\mathtt{1}$ during some iteration.

During the last substep, automata B go back to $\mathtt{0}^n$ because of the modulo, and automata D go back to state $\tilde{x}$. Since the number of substeps ℓ is greater than 2^n (Lemma 6), the iterations of C search the whole orbit of $\tilde{x}$, and at the end of the step automaton R has recorded whether the **Iter-CVP** instance is positive (went to state $\mathtt{1}$) or negative (still in state $\mathtt{0}$). The images are respectively $y_- = \mathtt{0}^{q_{k_n}}\mathtt{0}^{\ell'}\tilde{x}\mathtt{0}$ or $y_+ = \mathtt{0}^{q_{k_n}}\mathtt{0}^{\ell'}\tilde{x}\mathtt{1}$. This concludes the reductions, to **BP-Step-Bit** by asking whether automaton R (numbered $q_{k_n} + 2n$) is in state $\mathtt{1}$, to **BP-Step** by asking whether the image of x is y_+, and to **BP-Fixed-Point-Verif** because $y_- = x$ (coPSPACE-hardness). □

As a corollary, the associated functional problem of computing $f_{\{\mu\}}$ is computable in polynomial space and is PSPACE-hard for polynomial time Turing reductions (not for many-one reductions, as there is no concept of negative instance for total functional problems). Deciding whether a given configuration y has a preimage through $f_{\{\mu\}}$ is also PSPACE-complete (see Appendix A for details).

Now, we study the computational complexity of problems related to the existence of fixed points and limit cycles in an automata network under block-parallel update schedule. Again, we need to consider the image of all configurations, and have no control on neither the start configuration x nor the end configuration y during the dynamics of substeps. In particular, the counter may be initialized to any value, and the bit R may already be set to $\mathtt{1}$. We adapt the previous reductions accordingly.

Block-parallel fixed point (**BP-Fixed-Point**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i\in[\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$. Question: does $\exists x \in \mathbb{B}^n : f_{\{\mu\}}(x) = x$?

Block-parallel limit cycle of length k (**BP-Limit-Cycle-k**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i\in[\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$. Question: does $\exists x \in \mathbb{B}^n : f^k_{\{\mu\}}(x) = x$?

Block-parallel limit cycle (**BP-Limit-Cycle**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i\in[\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$, $k \in \mathbb{N}_+$. Question: does $\exists x \in \mathbb{B}^n : f^k_{\{\mu\}}(x) = x$?

On limit cycles we have a family of problems (one for each integer k), and a version where k is part of the input (encoded in binary). It makes no difference on the

complexity.

Theorem 5. BP-Fixed-Point, BP-Limit-Cycle-k *for any* $k \in \mathbb{N}_+$ *and* **BP-Limit-Cycle** *are* PSPACE*-complete.*

Proof. These problems still belong to PSPACE, because they amount to enumerating configurations and computing images by $f_{\{\mu\}}$, which can be performed from **BP-Step** (Theorem 4).

We start with the hardness proof for the fixed point existence problem, and we will then adapt it to limit cycle existence problems. Given an instance $C : \mathbb{B}^n \to \mathbb{B}^n$, $\tilde{x} \in \mathbb{B}^n$, $i \in [\![n]\!]$ of **Iter-CVP**, we construct the same block-parallel update schedule μ' as in the proof of Theorem 4, and modify the local functions of automata B and R as follows:

- automata B increment a counter modulo ℓ at each substep, and go to 0 when the counter is greater than (or equal to) $\ell - 1$; and
- automaton R records whether a state 1 appears at the i-th bit of x_D, and flips when the counter is equal to $\ell - 1$, i.e.,

$$f_R(x) = \begin{cases} x_R \vee x_{q_{k_n}+\ell'+i} & \text{if } x_B < \ell - 1, \\ \neg x_R & \text{otherwise.} \end{cases}$$

Recall that automata D iterate the circuit when $x_B < \ell - 1$ and go to $\tilde{x}$ otherwise, and that the number ℓ of substeps is larger than 2^n.

If the **Iter-CVP** instance is positive, then configuration $x = \mathtt{0}^{q_{k_n}}\mathtt{0}^{\ell'}\tilde{x}\mathtt{0}$ is a fixed point of $f_{\{\mu'\}}$. Indeed, during the ℓ-th and last substep, the primes P are still in state $\mathtt{0}^{q_{k_n}}$, the counter B goes back to 0 (state $\mathtt{0}^{\ell'}$), the circuit D goes back to $\tilde{x}$, and automaton R has recorded the 1 which is flipped into state 0.

Conversely, if there is a fixed point configuration x, then the counter must be at most $\ell - 1$ because of the modulo ℓ increment. Furthermore, automata D will encounter one substep during which it goes to $\tilde{x}$, hence the resulting configuration on D will be in the orbit of $\tilde{x}$, i.e., x_D is in the orbit of $\tilde{x}$. Finally, automaton R will also encounter exactly one substep during which it is flipped (when $x_B \geq \ell - 1$). As a consequence, in order to go back to its initial value x_R, the state of R must be flipped during another substep, which can only happen when it is in state 0 and automaton $q_{k_n} + \ell' + i$ is in state 1. We conclude that the i-th bit of a configuration in the orbit of $\tilde{x}$ is in state 1 during some iteration of the circuit C, meaning that the **Iter-CVP** instance is positive. Remark that in this case, configuration $\mathtt{0}^{q_{k_n}}\mathtt{0}^{\ell}\tilde{x}\mathtt{0}$ is one of the fixed points.

For the limit cycle existence problems, we modify the construction to let the counter go up to $k\ell - 1$. Precisely:

- $\ell' = \lceil \log_2(k\ell) \rceil$ automata B implement a binary counter which is incremented at each substep, and goes to 0 when $x_B \geq k\ell - 1$;

- n automata D iterate the circuit C if $x_B < \ell - 1$, else go to state $\tilde{x}$ (no change); and
- 1 automaton R records whether a state 1 appears in the i-th bit of x_D, and flips when the counter is equal to $\ell - 1$.

The reasoning is identical to the case $k = 1$, except that the counter needs k times ℓ substeps, i.e., k steps, in order to go back to its initial value. As a consequence, there is no x and $k' < k$ such that $f^{k'}_{\{\mu\}}(x) = x$, and the dynamics has no limit cycle of length smaller than k. Remark that when the **Iter-CVP** instance is positive, configurations $(0^{q_{k_n}} B_i \tilde{x} 0)_{i \in [\![k]\!]}$ with B_i the ℓ'-bits encoding of $i\ell$ form one of the limit cycles of length k. Also remark that the encoding of k in binary within the input has no consequence, neither on the PSPACE algorithm, nor on the polynomial time many-one reduction. □

Remark that our construction also applies to the notion of limit cycle $x^0, \dots, x^{p-1}$ where it is furthermore required that all configurations are different (this corresponds to having the minimum length p): the problem is still PSPACE-complete.

4.3. Reachability and general complexity bounds

In this part, we settle the computational complexity of the classical reachability problem, which is unsurprisingly still PSPACE-hard by reduction from another model of computation (see Appendix A for details). In light of what precedes, one may be inclined to think that any problem related to the dynamics of automata networks under block-parallel update schedules is PSPACE-hard. We prove that this is partly true with a general complexity bound theorem on subdynamics existing within $f_{\{\mu\}}$, based on our previous results on fixed points and limit cycles. However, we will also prove that a Rice-like complexity lower bound analogous to the main results of [23], i.e., which would state that any non-trivial question on the dynamics (on the functional graph of $f_{\{\mu\}}$) expressible in first order logics is PSPACE-hard, does not hold (unless a collapse of PSPACE to the first level of the polynomial hierarchy). Indeed, we will see that deciding the bijectivity $(\forall x, y \in \mathbb{B}^n : f_{\{\mu\}}(x) = f_{\{\mu\}}(y) \implies x = y)$ is complete for coNP. We conclude the section with a discussion on reversible dynamics.

From the fixed point and limit cycle theorems in Section 4.2, we now derive that any particular subdynamics is hard to identify within $f_{\{\mu\}}$ under block-parallel update schedule. A functional graph is a directed graph of out-degree exactly one, and we assimilate $f_{\{\mu\}}$ to its functional graph. We define a family of problems, one for each functional graph G to find as a subgraph of $f_{\{\mu\}}$, and prove that the problem is always PSPACE-hard. Since PSPACE = coPSPACE, checking the existence of a subdynamics is as hard as checking the absence of a subdynamics, even though the former is a local property whereas the latter is a global property at the dynamics scale. This is understandable in regard of the fact that PSPACE scales

everything to the global level (one can search the whole dynamics in PSPACE), because verifying that a given set of configurations (a certificate) gives the subgraph G is difficult (Theorem 4).

Block-parallel G as subdynamics (**BP-Subdynamics-G**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i \in [\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$. Question: does $G \sqsubset f_{\{\mu\}}$?

Remark that asking whether G appears as a subgraph or as an induced subgraph makes no difference when G is functional (has out-degree exactly one), because $f_{\{\mu\}}$ is also functional: it is necessarily induced since there is no arc to delete.

Theorem 6. BP-Subdynamics-G *is* PSPACE*-complete for any functional graph* G*.*

Proof. A polynomial space algorithm for **BP-G-Subdynamics** consists in enumerating all subsets $S \subseteq \mathbb{B}^n$ of size $|S| = |V(G)|$, and test for each whether the restriction of $f_{\{\mu\}}$ to S is isomorphic to G (functional graphs are planar hence isomorphism can be decided in logarithmic space [16]).

For the PSPACE-hardness, the idea is to choose a fixed point or limit cycle in G, and make it the decisive element whose existence or not lets G be a subgraph of the dynamics or not. Since G is a functional graph, it is composed of fixed points and limit cycles, with hanging trees rooted into them (the trees are pointing towards their root). Let $G(v)$ denote the unique out-neighbor of $v \in V(G)$.

Let us first assume that G has a limit cycle of length $k \geq 2$, or a fixed point with a tree of height greater or equal to 1 hanging (the case where G has only isolated limit cycles is treated thereafter). A fixed point is assimilated to a limit cycle of length $k = 1$. Let G' be the graph G without this limit cycle of size k, and let U be the vertices of G' without out-neighbor (if $k = 1$ then $U \neq \emptyset$). We reduce from **Iter-CVP**, and first compute the f, μ of size n obtained by the reduction from Theorem 5 for the problem **BP-Limit-Cycle-k**. We have that $f_{\{\mu\}}$ has a limit cycle of length k on configurations $(\mathtt{0}^{q_{k_n}} B_i \tilde{x} \mathtt{0})_{i \in [\![k]\!]}$ (or configuration $\mathtt{0}^{q_{k_n}} \mathtt{0}^{\ell} \tilde{x} \mathtt{0}$ for $k = 1$) if and only if the **Iter-CVP** instance is positive.

We construct g on $n + 1$ automata, and the update schedule μ' being the union of μ with a singleton o-block for the new automaton. We assume that $n \geq |V(G)| - k$, otherwise we pad f, μ to that size (with identity local functions for the new automata). The idea is that g will consist in a copy of f on the subspace $x_n = \mathtt{0}$, and a copy of G' on the subspace $x_n = \mathtt{1}$ where the images of the configurations corresponding to the vertices of U will be configurations of the potential limit cycle of $f_{\{\mu\}}$ (in the other subspace $x_n = \mathtt{0}$). Other configurations in the subspace $x_n = \mathtt{1}$ will be fixed points. Figure 4 illustrates the construction. Recall that G is fixed, and consider a mapping $\alpha : V(G) \to \{\mathtt{0}, \mathtt{1}\}^n$ such that

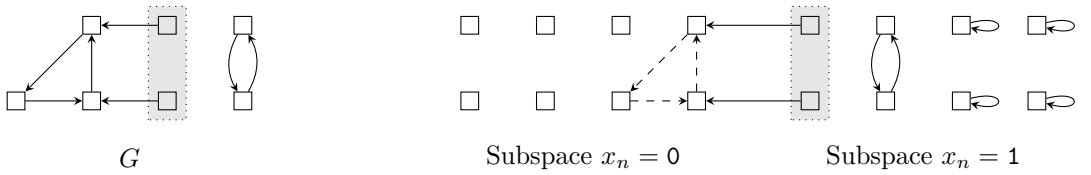

Figure 4: Construction of g in the proof of Theorem 6. Subspace $x_n = 0$ contains a copy of f with a potential limit cycle dashed. Subspace $x_n = 1$ implements G', and wires configurations of U (grey area) to the potential limit cycle in the copy of f (remaining configurations are fixed points).

vertices of the limit cycle of length k are sent to the configurations $(0^{q_{kn}} B_i \tilde{x} 0)_{i \in [\![k]\!]}$ respectively (or $0^{q_{kn}} 0^{\ell} \tilde{x} 0$ for $k = 1$). We define:

$$g(x) = \begin{cases} f(x_{[\![n]\!]})0 & \text{if } x_n = 0, \\ \alpha(G(v))0 & \text{if } x_n = 1 \text{ and } \exists v \in U : \alpha(v) = x_{[\![n]\!]}, \\ \alpha(G(v))1 & \text{if } x_n = 1 \text{ and } \exists v \in G' \setminus U : \alpha(v) = x_{[\![n]\!]}, \\ x & \text{otherwise.} \end{cases}$$

The obtained dynamics $g_{\{\mu'\}}$ has one copy of $f_{\{\mu\}}$ (in subspace $x_n = 0$), with a copy of G' (in subspace $x_n = 1$) which becomes a copy of G if configurations $(0^{q_{kn}} B_i \tilde{x} 0)_{i \in [\![k]\!]}$ (or $0^{q_{kn}} 0^{\ell} \tilde{x} 0$ in the case $k = 1$) form a limit cycle of length k. Moreover, it becomes a copy of G only if so by our assumption on the limit cycle or fixed point of G, because the remaining configurations in subspace $x_n = 1$ are all isolated fixed points. This concludes the reduction.

For the case where G is made of k isolated fixed points, we reduce from **BP-Fixed-Point** and construct an automata network with k copies of the dynamics of f, by adding $\lceil \log_2(k) \rceil$ automata with identity local functions. □

When the property of being a functional graph is dropped, that is when the out-degree of G is at most one (otherwise any instance is trivially negative), problem **BP-Subdynamics-G** is subtler. Indeed, one can still ask for the existence of fixed points, limit cycles and any functional subdynamics PSPACE-complete by Theorem 6, but new problems arise, some of which are provably complete only for coNP. The symmetry of existence versus non existence is broken. In what follows, we settle that deciding the bijectivity of $f_{\{\mu\}}$ is coNP-complete, and then discuss the complexity of decision problems which are subsets of bijective networks, such as the problem of deciding whether $f_{\{\mu\}}$ is the identity. We conclude the section by proving that it is nevertheless PSPACE-complete to decide whether $f_{\{\mu\}}$ is a constant map. These results hint at the subtleties behind a full characterization of the computational complexity of **BP-Subdynamics-G** for all graphs of out-degree at most one.

Block-parallel bijectivity (**BP-Bijectivity**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i \in [\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$. Question: is $f_{\{\mu\}}$ bijective?

Remark that, because the space of configurations is finite, injectivity, surjectivity and bijectivity are equivalent properties of $f_{\{\mu\}}$.

Lemma 7. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ a BAN and $\mu \in \mathsf{BP}_n$ a block-parallel update mode. Then $f_{\{\mu\}}$ is bijective if and only if $f_{(W)}$ is bijective for every block W of $\varphi(\mu)$.*

Proof. The right to left implication is obvious since $f_{\{\mu\}}$ is a composition of bijections $f_{(W)}$. We prove the contrapositive of the left to right implication, assuming the existence of a block W in $\varphi(\mu)$ such that $f_{(W)}$ is not bijective. Let W_ℓ be the first such block in the sequence $\varphi(\mu)$, so there exist $x, y \in \mathbb{B}^n$ such that $x \neq y$ but $f_{(W_\ell)}(x) = f_{(W_\ell)}(y) = z$. By minimality of ℓ, the composition $g = f_{(W_{\ell-1})} \circ \cdots \circ f_{(W_0)}$ is bijective, hence there also exist $x', y' \in \mathbb{B}^n$ with $x' \neq y'$ such that $g(x') = x$ and $g(y') = y$. That is, after the ℓ-th substep the two configurations x' and y' have the same image z, and we conclude that $f_{\{\mu\}}(x') = f_{\{\mu\}}(y') = f_{(W_{p-1})} \circ \cdots \circ f_{(W_{\ell+1})}(z)$ therefore $f_{\{\mu\}}$ is not bijective. □

Lemma 7 shows that bijectivity can be decided at the local level of circuits (not iterated), which can be checked in coNP and gives Theorem 7.

Theorem 7. BP-Bijectivity *is* coNP*-complete.*

Proof. A coNP algorithm can be established from Lemma 7, because it is equivalent to check the bijectivity at all substeps. A non-deterministic algorithm can guess a temporality $t \in [\![|\varphi(\mu)|]\!]$ (in binary) within the substeps, two configurations x, y, and then check in polynomial time that they certify the non-bijectivity of that substep as follows. First, construct W the t-th block of $\varphi(\mu)$, by computing t modulo each o-block size to get the automata from that o-block. Second, check that $f_{(W)}(x) = f_{(W)}(y)$.

The coNP-hardness is a direct consequence of that complexity lower bound for the particular case of the parallel update schedule [40, Theorem 5.17]. □

We now turn our attention to the recognition of identity dynamics.

Block-parallel identity (**BP-Identity**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i \in [\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$. Question: does $f_{\{\mu\}}(x) = x$ for all $x \in \mathbb{B}^n$?

This problem is in PSPACE, and is coNP-hard by reduction from the same problem in the parallel case [40, Theorem 5.18]. However, it is neither obvious to design a coNP-algorithm to solve it, nor to prove PSPACE-hardness by reduction from **Iter-CVP**.

Open problem 8. BP-Identity *is* coNP*-hard and in* PSPACE*. For which complexity class is it complete?*

A major obstacle to the design of an algorithm, or of a reduction from **Iter-CVP** to **BP-Identity**, lies in the fact that, by Theorem 7, "hard" instances of the latter are bijective networks (because non-bijective instances can be recognized in our immediate lower bound coNP, and they are all negative instances of **BP-Identity**). A reduction would therefore be related to the lengths of cycles in the dynamics of substeps, and whether they divide the least common multiple of o-block sizes (for $x \in \mathbb{B}^n$ such that $f(x) = x$) or not ($f(x) \neq x$).

Nonetheless, we are able to prove another lower bound, related to the hardness of computing the number of models of a given propositional formula. The canonical ModP-complete problem takes as input a formula ψ and two integers k, i encoded in unary, and consists in deciding whether the number of models of ψ is congruent to k modulo the i-th prime number (which can be computed in polytime). It generalizes classes $\mathsf{Mod_kP}$ (such as the parity case $\mathsf{Mod_2P} = \oplus\mathsf{P}$), and it is notable that #P polytime truth-table reduces to ModP [33].

Theorem 9. BP-Identity *is* ModP*-hard (for polytime many-one reduction).*

The proof of this theorem can be found in Appendix A.

Our attemps to prove PSPACE-hardness failed, for the following reasons. To get bijective circuits one could reduce from reversible Turing machines (RTM) and problem **Reversible Linear Space Acceptance** [34]. A natural strategy would be to simulate a RTM for an exponential number of subteps, and then simulate it backwards for that same number of substeps, while ending in the exact same configuration (identity map) if and only if the simulation did not halt or was not in the orbit of the given input w. The difficulty with this approach is that the dynamics of substeps must not be the identity map when a *conjunction* of two temporally separated events happens: first that the simulation has halted, and second that the starting configuration was w. It therefore requires to remember at least one bit of information, which is subtle in the reversible setting. Indeed, the constructions of [34] and [37] consider only starting configurations of the Turing machine in the initial state and with blank tapes. However, in the context of Boolean automata networks, any configuration must be considered (hence any configuration of the simulated Turing machine).

Regarding iterated circuits simulating reversible cellular automata (for which the whole configuration space is usualy considered), the literature focuses on decidability issues [31, 44], but a recent contribution fits our setting and we derive the following. $\mathsf{FP}^{\mathsf{PSPACE}}$ is the class of functions computable in polynomial time with an oracle in PSPACE.

Theorem 10 ([20, Theorem 5.7])**.** *There is a one-dimensional reversible cellular automaton for which simulating any given number of iterations, with periodic boundary conditions, is complete for* $\mathsf{FP}^{\mathsf{PSPACE}}$

Corollary 2. *Given* $(f_i : \mathbb{B}^n \to \mathbb{B})_{i\in[\![n]\!]}$ *as circuits,* $\mu \in \mathsf{BP}_n$ *such that* $f_{\{\mu\}}$ *is bijective,* $x \in \mathbb{B}^n$ *and* $t \in [\![|\varphi(\mu)|]\!]$ *in binary, computing the configuration at the* t*-th substep is complete for* $\mathsf{FP}^{\mathsf{PSPACE}}$*.*

The proof of this corollary can be found in Appendix A.

Intuitively, the dynamics of substeps embeds complexity. The relationship to the complexity of computing the configuration after the whole step composed of $|\varphi(\mu)|$ substeps, *i.e.* the image through $f_{\{\mu\}}$, is not obvious.

Being a constant map is another global property of the dynamics, which turns out to be PSPACE-complete to recognize for BANs under block-parallel update schedules.

Block-parallel constant (**BP-Constant**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i\in[\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$. Question: does there exist $y \in \mathbb{B}^n$ such that $f_{\{\mu\}}(x) = y$ for all $x \in \mathbb{B}^n$?

Theorem 11. BP-Constant *is* PSPACE*-complete.*

Proof. To decide **BP-Constant**, one can simply enumerate all configurations and compute their image (Theorem 4) while checking that it always gives the same result.

For the PSPACE-hardness proof, we reduce from **Iter-CVP**. Given a circuit $C : \mathbb{B}^n \to \mathbb{B}^n$, a configuration $\tilde{x}$ and $i \in [\![n]\!]$, we apply Lemma 6 to construct g_n, μ_n on automata set $P = [\![q_{k_n}]\!]$. Automata from P have constant $\mathtt{0}$ local functions, and the number of substeps is $\ell = |\varphi(\mu_n)| > 2^n$. We add (Figure 5 illustrates the obtained dynamics):

- $\ell' = \lceil \log_2(\ell) \rceil$ automata numbered $B = \{q_{k_n}, \dots, q_{k_n}+\ell'-1\}$, implementing a ℓ'-bits binary counter that increments at each substep, and sets all automata from B in state $\mathtt{1}$ when the counter is greater or equal to $\ell - 1$;
- n automata numbered $D = \{q_{k_n} + \ell', \dots, q_{k_n} + \ell' + n - 1\}$, whose local functions are given below; and
- 1 automaton numbered $R = \{q_{k_n} + \ell' + n\}$, whose local function is given below.

$$f_D(x) = \begin{cases} C(\tilde{x}) & \text{if } x_B = 0 \\ C(x_D) & \text{if } 0 < x_B < \ell - 1 \\ \mathtt{0}^n & \text{otherwise} \end{cases} \qquad f_R(x) = \begin{cases} \tilde{x}_i & \text{if } x_B = 0 \\ x_R \vee x_{q_{k_n}+\ell'+i} & \text{if } 0 < x_B < \ell \\ \mathtt{1} & \text{otherwise} \end{cases}$$

We also add singletons to μ_n for these additional automata, via $\mu' = \mu_n \cup \bigcup_{j\in B\cup D\cup R}\{(j)\}$.

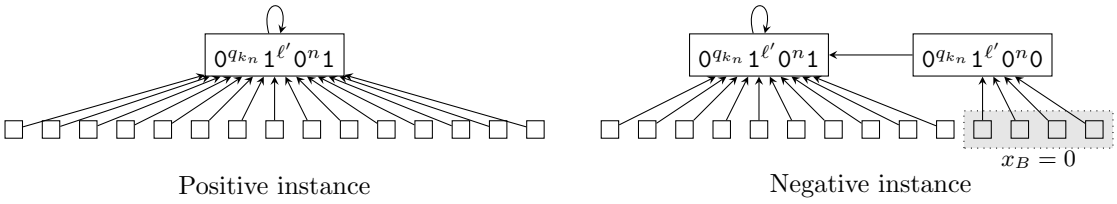

Figure 5: Illustration of the dynamics obtained for the reduction to **BP-Constant** in the proof of Theorem 11. Configurations x with the counter automata B initialized to $x_B = 0$ either go to $0^{q_{k_n}}1^{\ell'}0^n1$ (left, positive instance), or to $0^{q_{k_n}}1^{\ell'}0^n0$ (right, negative instance). Only the bit of automata R changes.

For any configuration x with a counter not initialized to 0, i.e., with $x_B \neq 0$, the counter will reach and remain in the all 1 state before the last substep, therefore automata from D will be updated to 0^n and automaton R will be updated to 1. We conclude that $f_{\{\mu'\}}(x) = 0^{q_{k_n}}1^{\ell'}0^n1$. For configurations x with $x_B = 0$, substeps proceed as follows:

- automata B count until $\ell - 1$ at the penultimate substep (recall that $\ell = |\varphi(\mu_n)| = |\varphi(\mu'_n)|$), which finally brings them all in state 1 during the last substep;
- automata D iterate the circuit C, starting from $C(\tilde{x})$ during the first substep; and
- automaton R records whether a 1 appears or not in the whole orbit of $\tilde{x}$ (recall that $\ell = |\varphi(\mu'_n)| > 2^n$), starting from $\tilde{x}$ itself during the first substep (even though $x_D \neq \tilde{x}$) and without encountering the "1 otherwise" case.

We conclude that the image of x on automata P is $0^{q_{k_n}}$, on B is $1^{\ell'}$, on D is 0^n, and on R it depends whether the **Iter-CVP** instance is positive (automaton R in state 1) or negative (automaton R in state 0). This completes the reduction: the image is always $0^{q_{k_n}}1^{\ell'}0^n1$ if and only if the **Iter-CVP** instance is positive. □

5. Conclusion and perspectives

This article presents a theoretical study of block-parallel update modes under two different angles.

The first part of this article focused on combinatorial aspects, in particular counting and enumerating not only the general set of block-parallel update mode, but also the classes for two equivalence relations. These relations weren't chosen arbitrarily, since the first one gives us the minimal number of update modes required to generate every possible distinct underlying dynamical system, and the second one generates dynamical systems that all have different sets of limit cycles (limit dynamics). Regarding the enumeration algorithms we presented, one of the first questions that comes to mind would be about their complexity. The three

algorithms we mentioned seem to belong to EnumP [15], but this requires a formal analysis. In particular, the question of which subclasses of EnumP they belong to still needs to be addressed.

The second half of this article didn't answer this question, rather focusing on the computational complexity of classical decision problems involving Boolean automata networks. While an automaton cannot be updated twice in a block-sequential update mode, hence limiting the number of substeps, update repetitions are allowed in block-parallel update modes, which leads to a much higher ceiling for the number of substeps. This provides a greater expressiveness, but also higher complexity costs. Mainly, computing a single transition goes from being feasible in polynomial time with all block-sequential schedules [41] to PSPACE-hard in this context (Theorem 4). This raises the complexity of classical problems related to the existence of preimages, fixed points, limit cycles, and the recognition of constant dynamics from NP-complete (existence problems) or coNP-complete (global dynamical properties) for block-sequential schedules to PSPACE-complete for block-parallel schedules.

One might be tempted to draw the following conjecture from these results.

Conjecture 1 (false)**.** *If a problem is* NP*-hard or* coNP*-hard and in* PSPACE *for block-sequential update schedules then it is* PSPACE*-complete for block-parallel update schedules.*

This conjecture is however disproven by the recognition of bijective dynamics. Since a single substep is enough to identify the absence of bijectivity, the increase in the number of substeps did not affect the complexity of the problem, and it stays coNP-complete for block-parallel schedules. The complexity of recognizing identity dynamics under block-parallel schedules is still an open problem.

After determining the complexity of recognizing preimages, image points or fixed points, the next logical step would be the complexity of counting them.

An important remark is that, while these proofs were written for Boolean automata networks, they also apply to multi-valued automata networks.

As mentioned earlier, the possibility of updating the same automaton multiple times in one step allows for a greater variety of possible dynamics. Especially, they can break the fixed point invariance property that holds for every block-sequential update mode, but may fail for block-parallel update mode with such repetitions. It would thus be pertinent to characterize what in these repetitions triggers the creation of new fixed points. More generally, it would be interesting to study the following problem: given an AN f , to which extent is f block-parallel sensible or robust? In [10, 9, 11], the authors addressed this question on block-sequential Boolean ANs by developing the concept of update digraphs, capturing conditions of dynamical equivalence at the syntactical level. This concept is however unapplicable to update schedules where automata repetitions appear. Creating an

equivalent of update digraphs that allows for update repetitions would be an important step in the understanding of updating sensitivity and robustness of ANs. Another approach would be to study specifically how interaction cycles behave when paired with block-parallel update modes.

References

[1] OEIS A000262. The online encyclopedia of integer sequences. https://oeis.org/A000262.

[2] OEIS A000670. The online encyclopedia of integer sequences. https://oeis.org/A000670.

[3] OEIS A061095. The online encyclopedia of integer sequences. https://oeis.org/A061095.

[4] OEIS A182666. The online encyclopedia of integer sequences. https://oeis.org/A182666.

[5] T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 17–28. Wold Scientific, 1999.

[6] N. Alon. Asynchronous threshold networks. *Graphs and Combinatorics*, 1:305—310, 1985. doi:10.1007/BF02582959.

[7] J. Aracena, J. Demongeot, É. Fanchon, and M. Montalva. On the number of different dynamics in Boolean networks with deterministic update schedules. *Mathematical Biosciences*, 242:188–194, 2013. doi:10.1016/j.mbs.2013.01.007.

[8] J. Aracena, É. Fanchon, M. Montalva, and M. Noual. Combinatorics on update digraphs in Boolean networks. *Discrete Applied Mathematics*, 159:401–409, 2011.

[9] J. Aracena, É. Fanchon, M. Montalva, and M. Noual. ombinatorics on update digraphs in Boolean networks. *Discrete Applied Mathematics*, 159:401–409, 2012.

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

[11] J. Aracena, L. Gómez, and L. Salinas. Limit cycles and update digraphs in Boolean networks. *Discrete Applied Mathematics*, 161:1–12, 2013.

[12] A. Benecke. Chromatin code, local non-equilibrium dynamics, and the emergence of transcription regulatory programs. *The European Physical Journal E*, 19:353–366, 2006.

[13] F. Bridoux, C. Gaze-Maillot, K. Perrot, and S. Sené. Complexity of Limit-Cycle Problems in Boolean Networks. In *Proceedings of SOFSEM'2021*, volume 12607 of *LNCS*, pages 135–146. Springer, 2021.

[14] F. Bridoux, P. Guillon, K. Perrot, S. Sené, and G. Theyssier. On the cost of simulating a parallel boolean automata network by a block-sequential one. In *Proceedings of TAMC'2017*, volume 10185 of *LNCS*, pages 112–128, 2017. `arXiv:1702.03101`, `doi:10.1007/978-3-319-55911-7_9`.

[15] N. Creignou, M. Kröll, R. Pichler, S. Skritek, and H. Vollmer. A complexity theory for hard enumeration problem. *Discrete Applied Mathematics*, 268:191–209, 2019.

[16] S. Datta, N. Limaye, P. Nimbhorkar, T. Thierauf, and F. Wagner. Planar Graph Isomorphism is in Log-Space. In *Algebraic Methods in Computational Complexity*, volume 9421, pages 1–32. Schloss Dagstuhl, 2010. `doi:10.4230/DagSemProc.09421.6`.

[17] J. Demongeot, A. Elena, and S. Sené. Robustness in regulatory networks: a multi-disciplinary approach. *Acta Biotheoretica*, 56:27–49, 2008.

[18] J. Demongeot and S. Sené. About block-parallel Boolean networks: a position paper. *Natural Computing*, 19:5–13, 2020.

[19] A. Dennunzio, E. Formenti, L. Manzoni, and A. E. Porreca. Complexity of the dynamics of reaction systems. *Information and Computation*, 267:96–109, 2019.

[20] David Eppstein. The Complexity of Iterated Reversible Computation. *TheoretiCS*, 2, 2023. `doi:10.46298/theoretics.23.10`.

[21] B. Fierz and M. G. Poirier. Biophysics of chromatin dynamics. *Annual Review of Biophysics*, 48:321–345, 2019.

[22] P. Floréen and P. Orponen. On the computational complexity of analyzing Hopfield nets. *Complex Systems*, 3:577–587, 1989.

[23] G. Gamard, P. Guillon, K. Perrot, and G. Theyssier. Rice-like Theorems for automata networks. In *Proceedings of STACS'21*, volume 187 of *LIPIcs*, pages 32:1–32:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

[24] C. E. Giacomantonio and G. J. Goodhill. A Boolean model of the gene regulatory network underlying mammalian cortical area development. *PLoS Computational Biology*, 6:e1000936, 2010.

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

[26] E. Goles, P. Montealegre, V. Salo, and I. Törmä. PSPACE-completeness of majority automata networks. *Theoretical Computer Science*, 609:118–128, 2016. doi:10.1016/j.tcs.2015.09.014.

[27] E. Goles and M. Noual. Block-sequential update schedules and Boolean automata circuits. In *Proceedings of AUTOMATA'2010*, pages 41–50. DMTCS, 2010.

[28] E. Goles and L. Salinas. Comparison between parallel and serial dynamics of Boolean networks. *Theoretical Computer Science*, 396:247–253, 2008.

[29] J. C. Hanse and J. Ausio. Chromatin dynamics and the modulation of genetic activity. *Trends in Biochemical Sciences*, 17:187–191, 1992.

[30] M. R. Hübner and D. L. Spector. Chromatin dynamics. *Annual Review of Biophysics*, 39:471–489, 2010.

[31] J. Kari. Reversible cellular automata. In *Proceedings of DLT'2005*, volume 3572 of *LNCS*, pages 57–68. Springer, 2005. doi:10.1007/11505877_5.

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

[33] J. Köbler and S. Toda. On the power of generalized Mod-classes. *Mathematical Systems Theory*, 29:33—-46, 1996. doi:10.1007/BF01201812.

[34] K.-J. Lange, P. McKenzie, and A. Tapp. Reversible Space Equals Deterministic Space. *Journal of Computer and System Sciences*, 60(2):354–367, 2000. doi:10.1006/jcss.1999.1672.

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

[36] L. Mendoza and E. R. Alvarez-Buylla. Dynamics of the genetic regulatory network for *Arabidopsis thaliana* flower morphogenesis. *Journal of Theoretical Biology*, 193:307–319, 1998.

[37] K. Morita. *Theory of Reversible Computing*. Springer, first edition, 2017. doi:10.1007/978-4-431-56606-9.

[38] M. Noual and S. Sené. Towards a theory of modelling with Boolean automata networks - I. Theorisation and observations, 2011. arXiv:1111.2077.

[39] L. Paulevé and S. Sené. *Systems biology modelling and analysis: formal bioinformatics methods and tools*, chapter Boolean networks and their dynamics: the impact of updates, pages 173–250. Wiley, 2022.

[40] K. Perrot. *Études de la complexité algorithmique des réseaux d'automates*. Habilitation thesis, Université d'Aix-Marseille, 2022.

[41] P. Perrotin and S. Sené. Turning block-sequential automata networks into smaller parallel networks with isomorphic limit dynamics. In *Proceedings of CiE'23*, LNCS. Springer, 2023. To appear.

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

[43] K. Sutner. On the Computational Complexity of Finite Cellular Automata. *Journal of Computer and System Sciences*, 50(1):87–97, 1995. doi:10.1006/jcss.1995.1009.

[44] K. Sutner. The complexity of reversible cellular automata. *Theoretical Computer Science*, 325(2):317–328, 2004. doi:10.1016/j.tcs.2004.06.011.

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

[46] D. J. Wooten, S. M. Groves, D. R. Tyson, Q. Liu, J. S. Lim, R. Albert, C. F. Lopez, J. Sage, and V. Quaranta. Systems-level network modeling of small cell lung cancer subtypes identifies master regulators and destabilizers. *PLoS Computational Biology*, 15:e1007343, 2019.

Appendix A. Omitted proofs

Continuation of the proof of Theorem 1. Formula 1 is a sum for each partition of n (sum on i), of all the ways to fill all the matrices ($n!$) up to permutation within each column ($m(i,j)!$ for each of the j columns of M_j).

Formula 2 is a sum for each partition of n (sum on i), of the product for each column of the matrices (products on j and ℓ), of the choice of elements (among the remaining ones) to fill the column (regardless of their order within the column).

Formula 3 is a sum for each partition of n (sum on i), of the product for each matrix (product on j), of the choice of elements (among the remaining ones) to fill this matrix, multiplied by the number of ways to fill the columns of the matrix (product on ℓ) with these elements (regardless of their order within each column).

The equality between Formulas 1 and 2 is obtained by developing the binomial coefficients as follows: $\binom{x}{y} = \frac{x!}{y!\cdot(x-y)!}$, and by observing that the products of $\frac{x!}{(x-y)!}$ telescope. Indeed, denoting $a(j,\ell) = (n - \sum_{k=1}^{j-1} k\cdot m(i,k) - \ell\cdot m(i,j))!$, we have

$$\prod_{j=1}^{d(i)}\prod_{\ell=1}^{j}\frac{(n-\sum_{k=1}^{j-1}k\cdot m(i,k)-(\ell-1)\cdot m(i,j))!}{(n-\sum_{k=1}^{j-1}k\cdot m(i,k)-\ell\cdot m(i,j))!} = \prod_{j=1}^{d(i)}\prod_{\ell=1}^{j}\frac{a(j,\ell-1)}{a(j,\ell)} = \frac{n!}{0!} = n!$$

because $a(1,0) = n!$, then $a(1,j) = a(2,0)$, $a(2,j) = a(3,0)$, ..., until $a(d(i),j) = 0!$.

The equality between Formulas 2 and 3 is obtained by repeated uses of the identity $\binom{x}{z}\binom{x-z}{y} = \binom{x}{z+y}\binom{z+y}{y}$, which gives by induction on j:

$$\prod_{\ell=1}^{j}\binom{x-(\ell-1)\cdot y}{y} = \binom{x}{j\cdot y}\cdot\prod_{\ell=1}^{j}\binom{(j-\ell+1)\cdot y}{y}. \tag{A.1}$$

Indeed, $j = 1$ is trivial and, using the induction hypothesis on j then the identity

we get:

$$\begin{aligned}
\prod_{\ell=1}^{j+1}\binom{x-(\ell-1)\cdot y}{y} &= \binom{x-j\cdot y}{y}\cdot\prod_{\ell=1}^{j}\binom{x-(\ell-1)\cdot y}{y}\\
&= \binom{x-j\cdot y}{y}\cdot\binom{x}{j\cdot y}\cdot\prod_{\ell=1}^{j}\binom{(j-\ell+1)\cdot y}{y}\\
&= \binom{x}{(j+1)\cdot y}\cdot\binom{(j+1)\cdot y}{y}\cdot\prod_{\ell=1}^{j}\binom{(j-\ell+1)\cdot y}{y}\\
&= \binom{x}{(j+1)\cdot y}\cdot\prod_{\ell=0}^{j}\binom{(j-\ell+1)\cdot y}{y}\\
&= \binom{x}{(j+1)\cdot y}\cdot\prod_{\ell=1}^{j+1}\binom{(j+1-\ell+1)\cdot y}{y}.
\end{aligned}$$

As a result, Formula 3 is obtained from Formula 2 by applying Equation A.1 for each j with $x = n - \sum_{k=1}^{j-1} k\cdot m(i,k)$ and $y = m(i,j)$.

□

Proof of Lemma 3. We will start from the exponential generating function by finding the coefficient of x^n and proving that it is equal to $\frac{|\mathsf{BP}_n^0|}{n!}$, and thus that the associated sequence is $(|\mathsf{BP}_n^0|)_{n\in\mathbb{N}}$.

$$\begin{aligned}
\prod_{j\geq 1}\sum_{k\geq 0}\left(\frac{x^k}{k!}\right)^j &= \left(\sum_{k\geq 0}\frac{x^k}{k!}\right)\times\left(\sum_{k\geq 0}\frac{x^{2k}}{(k!)^2}\right)\times\left(\sum_{k\geq 0}\frac{x^{3k}}{(k!)^3}\right)\times\cdots\\
&= \underbrace{\left(1+x+\frac{x^2}{2!}+\cdots\right)}_{j=1}\times\underbrace{\left(1+x^2+\frac{x^4}{(2!)^2}+\cdots\right)}_{j=2}\times\underbrace{\left(1+x^3+\frac{x^6}{(2!)^3}+\cdots\right)}_{j=3}\times\cdots.
\end{aligned}$$

Each term of the distributed sum is obtained by associating a $k\in\mathbb{N}$ to each $j\in\mathbb{N}_+$, and by doing the product of the $\frac{1}{(k!)^j}\cdot x^{jk}$. Thus, if $\mathbb{N}^{\mathbb{N}_+}$ is the set of maps from $\mathbb{N}_+$ to $\mathbb{N}$, we have:

$$\prod_{j\geq 1}\sum_{k\geq 0}\left(\frac{x^k}{k!}\right)^j = \sum_{m\in\mathbb{N}^{\mathbb{N}_+}}\left(\prod_{j\geq 1}\frac{1}{(m(j)!)^j}\right)\cdot x^{\sum_{j\geq 1} j\cdot m(j)}.$$

From here, to get the coefficient of x^n, we need to do the sum only on the maps m such that $\sum_{j\geq 1} j\cdot m(j) = n$, which just so happen to be the partitions of n, with

$m(j)$ being the multiplicity of j in the partition. Thus, the coefficient of x^n is

$$\sum_{i=1}^{p(n)} \prod_{j\geq 1} \frac{1}{(m(i,j)!)^j} \;=\; \sum_{i=1}^{p(n)} \frac{1}{\prod_{j\geq 1}^{d(i)} (m(i,j)!)^j} \;=\; \frac{|\mathsf{BP}_n^0|}{n!}.$$

□

Proof of correction of Algorithm 1. We first argue that Algorithm 1 enumerates the correct number of block-parallel update modes, and then that any pair μ, μ' enumerated is such that $\mu \not\equiv_\star \mu'$.

For ease of reading in the rest of this proof, we will denote $a[j]$ as a_j and $m(i,j)$ as m_{ij}. From the placement of min_j described above (forced to be within the first a_j columns of matrix M_j), the difference with the previous algorithm is that, instead of having $\prod_{\ell=1}^{j} \binom{(j-\ell+1)\cdot m_{ij}}{m_{ij}}$ ways of filling matrix M_j, we only have the following number of ways (recall that M_j has j columns and m_{ij} rows):

$$\sum_{k=1}^{a_j} \left(\prod_{\ell=1}^{k-1} \binom{(j-\ell+1)\cdot m_{ij} - 1}{m_{ij}} \right) \cdot \binom{(j-k+1)\cdot m_{ij} - 1}{m_{ij} - 1} \cdot \left(\prod_{\ell=k+1}^{j} \binom{(j-\ell+1)\cdot m_{ij}}{m_{ij}} \right).$$

Indeed, the formula above sums, for each choice of a column k from 1 to a_j where min_j will be placed, the number of ways to place some elements within columns 1 to $k-1$ (first product on ℓ), times the number of ways to choose some elements that will accompany min_j within column k (middle binomial coefficient), times the number of ways to place some other elements within the remaining columns $k+1$ to j (second product on ℓ). Now, we have $\binom{(j-k+1)\cdot m_{ij}-1}{m_{ij}-1} = \frac{m_{ij}}{(j-k+1)\cdot m_{ij}} \cdot \binom{(j-k+1)\cdot m_{ij}}{m_{ij}} = \frac{1}{(j-k+1)} \cdot \binom{(j-k+1)\cdot m_{ij}}{m_{ij}}$. We also have $\binom{(j-\ell+1)\cdot m_{ij}-1}{m_{ij}} = \frac{j-l}{j-(l-1)} \cdot \binom{(j-\ell+1)\cdot m_{ij}}{m_{ij}}$. This means that the sum of the possible ways to choose the content of matrix M_j can be rewritten as follows:

$$\begin{aligned}
&\sum_{k=1}^{a_j} \left(\frac{1}{(j-k+1)} \cdot \prod_{\ell=1}^{k-1} \frac{j-\ell}{j-(\ell-1)} \cdot \prod_{\ell=1}^{j} \binom{(j-\ell+1)\cdot m_{ij}}{m_{ij}} \right) \\
&\qquad = \sum_{k=1}^{a_j} \left(\frac{1}{(j-k+1)} \cdot \frac{j-k+1}{j} \cdot \prod_{\ell=1}^{j} \binom{(j-\ell+1)\cdot m_{ij}}{m_{ij}} \right) \\
&\qquad = \sum_{k=1}^{a_j} \left(\frac{1}{j} \cdot \prod_{\ell=1}^{j} \binom{(j-\ell+1)\cdot m_{ij}}{m_{ij}} \right) \\
&\qquad = \frac{a_j}{j} \cdot \prod_{\ell=1}^{j} \binom{(j-\ell+1)\cdot m_{ij}}{m_{ij}}.
\end{aligned}$$

Hence in total, the algorithm enumerates the following number of update modes:

$$\sum_{i=1}^{p(n)} \prod_{j=1}^{d(i)} \left(\binom{n - \sum_{k=1}^{j-1} k \cdot m_{ik}}{j \cdot m_{ij}} \cdot \frac{a_j}{j} \cdot \prod_{\ell=1}^{j} \binom{(j-\ell+1) \cdot m_{ij}}{m_{ij}} \right).$$

In order to prove that this number is equal to $|\mathsf{BP}_n^\star|$, we need to prove that $\prod_{j=1}^{d(i)} \frac{a_j}{j} = \frac{1}{\mathrm{lcm}(i)}$. Denoting $L(j) = \mathrm{lcm}(\{k \in [\![j, d(i)]\!] \mid m_{ik} > 0\})$, we prove by induction that at the end of each step of the **for** loop from lines 5-10, we have:

$$\prod_{k=j}^{d(i)} \frac{a_k}{k} = \frac{1}{L(j)},$$

and the claim follows (when $j = 1$, we get $L(j) = \mathrm{lcm}(i)$). At the first step, $j = d(i)$, and

$$\frac{a_{d(i)}}{d(i)} = \frac{\gcd(\{d(i), 1\})}{d(i)} = \frac{1}{L(j)}.$$

We assume as induction hypothesis that for a given j, we have $\prod_{k=j}^{d(i)} \frac{a_k}{k} = \frac{1}{L(j)}$. There are two possible cases for $j-1$:

- If $m_{i(j-1)} = 0$, then

$$a_{j-1} = j - 1 \text{ and } \prod_{k=j-1}^{d(i)} \frac{a_k}{k} = \frac{j-1}{(j-1) \cdot L(j)} = \frac{1}{L(j-1)}.$$

- Otherwise,

$$\prod_{k=j-1}^{d(i)} \frac{a_k}{k} = \frac{\gcd(\{j-1, L(j)\})}{(j-1) \cdot L(j)}.$$

  And since $\frac{a \cdot b}{\gcd(\{a,b\})} = \mathrm{lcm}(\{a, b\})$, we have

$$\prod_{k=j-1}^{d(i)} \frac{a_k}{k} = \frac{1}{\mathrm{lcm}(\{j-1, L(j)\})} = \frac{1}{L(j-1)}.$$

We conclude that at the end of the loop, we have $\prod_{j=1}^{d(i)} \frac{a_j}{j} = \frac{1}{\mathrm{lcm}(i)}$, and thus that the algorithm enumerates the following number of update modes (cf. Remark 2):

$$\sum_{i=1}^{p(n)} \prod_{j=1}^{d(i)} \left(\binom{n - \sum_{k=1}^{j-1} k \cdot m_{ik}}{j \cdot m_{ij}} \cdot \prod_{\ell=1}^{j} \binom{(j-\ell+1) \cdot m_{ij}}{m_{ij}} \right) \cdot \frac{1}{\mathrm{lcm}(i)} = |\mathsf{BP}_n^\star|.$$

We now need to prove that the algorithm does not enumerate two equivalent update modes (in the sense of $\equiv_\star$). Algorithm 1 is heavily based on the algorithm from the previous section, in such a way that, for a given input, the output of Algorithm 1 will be a subset of that of the aforementioned algorithm. Said algorithm enumerates BP_n^0, which implies that every update mode enumerated by it has a different image by φ. This means that every block-parallel update mode enumerated by Algorithm 1 also has a different image by φ, and that the algorithm does not enumerate two equivalent update modes with a shift of 0.

We now prove by contradiction that the algorithm does not enumerate two equivalent update modes with a non-zero shift. Let $\mu, \mu' \in \mathsf{BP}_n$ be two update modes, both enumerated by Algorithm 1, such that $\mu \equiv_\star \mu'$ with a non-zero shift. Then, there is $k \in \{1, \dots, |\varphi(\mu')| - 1\}$ such that $\varphi(\mu) = \sigma^k(\varphi(\mu'))$, and μ, μ' are both generated from the same partition i, with $|\varphi(\mu)| = |\varphi(\mu')| = \mathrm{lcm}(i)$. Moreover, for each $j \in [\![d(i)]\!]$ the matrix M_j must contain the same elements A in both μ and μ' (so that they are repeated every j blocks in both $\varphi(\mu)$ and $\varphi(\mu')$), hence in particular min_j is the same in both enumerations.

We will prove by induction that every $j \in [\![d(i)]\!]$ such that $m(i,j) > 0$ divides k, and therefore $k = 0$ (equivalently $k = \mathrm{lcm}(i)$), leading to a contradiction. For the base case $j = d(i)$, we have $a_{d(i)} = 1$ (first iteration of the **for** loop lines 5-10), hence the call of `EnumBlockIsoAux` for any set A passes the condition of line 35 and $min_{d(i)}$ is immediately chosen to belong to C_1 (the first column of $M_{d(i)}$). When converted to block-sequential update modes, it means that $min_{d(i)}$ appears in all blocks indexed by $d(i) \cdot t$ with $t \in \mathbb{N}$, hence k must be a multiple of $d(i)$ so that σ^k maps blocks containing $min_{d(i)}$ to blocks containing $min_{d(i)}$.

As induction hypothesis, assume that for a given j, every $\ell \in [\![j, d(i)]\!]$ such that $m_{i\ell} > 0$ divides k. We will prove that j', the biggest number in the partition i (*i.e.* with $m_{ij'} > 0$) that is smaller than j, also divides k. In the matrix $M_{j'}$, the minimum $min_{j'}$ is forced to appear within the $a_{j'}$ first columns. This means that block indexes where it appears in both $\varphi(\mu)$ and $\varphi(\mu')$ can be written respectively as $j' \cdot t + b$ and $j' \cdot t + b'$ respectively, with $t \in \mathbb{N}$ and $b, b' \in [\![1, a_{j'}]\!]$. As a consequence, an automaton from $M_{j'}$ that is at the position b in $\varphi(\mu)$ is at a position of the form $j' \cdot t + b'$ in $\varphi(\mu')$. It follows that $b + k = j' \cdot t + b'$, which can be rewritten as $k = t \cdot j' + b' - b = t \cdot j' + d$, with $t \in \mathbb{N}$ and $d = b' - b \in [\![-a_{j'} + 1, a_{j'} - 1]\!]$. Moreover, we know by induction hypothesis that every number in the partition i that is greater than j' divides k, making k a common multiple of these numbers. We deduce that their lowest common multiple also divides k. Given that $a_{j'}$ is the gcd of j' and said lcm (lines 7-8), it means that $a_{j'}$ divides both j' and k, which implies that it also divides d. Since d is in $[\![-a_{j'} + 1, a_{j'} - 1]\!]$, we have $d = 0$ and thus, j' divides k. This concludes the induction.

If every number of the partition i divides k and $k \in [\![\mathrm{lcm}(i)]\!]$, then $k = 0$,

leading to a contradiction. This concludes the proof of correctness. □

Proof of Lemma 5. By the prime number theorem, there are approximately $\frac{N}{\ln(N)}$ primes lower than N. As a consequence, distinct prime integers $p_1, p_2, \ldots, p_{k_n}$ with $k_n = \lfloor \frac{n^2}{\ln(n^2)} \rfloor$ can be computed in time $\mathcal{O}(n^2)$ using Atkin sieve algorithm. Since $2 \leq p_i < n^2$, we have $2^{k_n} \leq \prod_{i=1}^{k_n} p_i < n^{2k_n}$. It holds that $2^{k_n} = 2^{\lfloor \frac{n^2}{2\ln(n)} \rfloor} > 2^n$, and $n^{2k_n} \leq n^{\frac{n^2}{\ln(n)}}$ with

$$\log_2\left(n^{\frac{n^2}{\ln(n)}}\right) = \frac{\frac{n^2}{\ln(n)}}{\log_n(2)} = \frac{n^2}{\ln(2)}$$

meaning that $n^{2k_n} \leq 2^{\frac{n^2}{\ln(2)}} < 2^{2n^2}$. □

Block-parallel preimage (**BP-Preimage**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i \in [\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$, $y \in \mathbb{B}^n$. Question: does $\exists x \in \mathbb{B}^n : f_{\{\mu\}}(x) = y$?

Theorem 12. BP-Preimage *is* PSPACE*-complete.*

The difficulty in this reduction is that we need to take into account the image of every configuration x. We modify the preceding construction by setting automata D to $\tilde{x}$ when the counter B encodes 0.

Proof. The algorithm for **BP-Preimage** computes the image of each configuration (enumerated in polynomial space with a simple counter) using the same procedure as **BP-Step** (Theorem 4), and decides whether there is some x such that $f_{\{\mu\}}(x) = y$.

Given an instance $C : \mathbb{B}^n \to \mathbb{B}^n$, $\tilde{x} \in \mathbb{B}^n$, $i \in [\![n]\!]$ of **Iter-CVP**, we construct the same block-parallel update schedule μ' as in the proof of Theorem 4, and modify the local functions of automata D and R as follows:

$$f_D(x) = \begin{cases} C(\tilde{x}) & \text{if } x_B = 0 \\ C(x_D) & \text{if } 0 < x_B < \ell - 1 \\ \mathtt{0}^n & \text{otherwise} \end{cases} \qquad f_R(x) = \begin{cases} \tilde{x}_i & \text{if } x_B = 0 \\ x_R \vee x_{q_{k_n}+\ell'+i} & \text{otherwise} \end{cases}$$

The purpose is that D iterates the circuit from $\tilde{x}$ when the counter is initialized to 0, and that R records whether the i-th bit of D has been in state $\mathtt{1}$ (including the initial substep). We set $y = \mathtt{0}^{q_{k_n}}\mathtt{0}^{\ell}\mathtt{0}^{n}\mathtt{1}$.

If the **Iter-CVP** instance is positive, then we have $f_{\{\mu'\}}(\mathtt{0}^{q_{k_n}}\mathtt{0}^{\ell}\mathtt{0}^{n}\mathtt{0}) = y$ (automata B go back to $\mathtt{0}^{q_{k_n}}$, automata D iterate circuit C from $\tilde{x}$ and end in state $\mathtt{0}^n$, and automaton R has recorded that the i-th bit of D has been to state $\mathtt{1}$).

Conversely, if there is a configuration x such that $f_{\{\mu'\}}(x) = y$, then the automata from the counter B must have started in state $x_B = \mathsf{0}^{q_{k_n}}$, because of the increment modulo ℓ which is the number of substeps. We deduce that D iterate circuit C for the whole orbit of $\tilde{x}$ and end in state $\mathsf{0}^n$, and that automaton R records the answer to the **Iter-CVP** instance. Since it it ends in state $y_R = \mathsf{1}$ by our assumption that $f_{\{\mu'\}}(x) = y$, we conclude that it is positive. □

Block-parallel reachability (**BP-Reachability**) Input: $(f_i : \mathbb{B}^n \to \mathbb{B})_{i \in [\![n]\!]}$ as circuits, $\mu \in \mathsf{BP}_n$, $x, y \in \mathbb{B}^n$. Question: does $\exists t \in \mathbb{N} : f^t_{\{\mu\}}(x) = y$?

Theorem 13. BP-Reachability *is* PSPACE*-complete.*

Proof. The problem belongs to PSPACE, because is can naively be solved by simulating the dynamics of $f_{\{\mu\}}$ starting from configuration x, for 2^n time steps.

Reachability problems in cellular automata and related models are known to be PSPACE-complete on finite configurations [43]. We reduce from the reachability problem for reaction systems, which can be seen as a particular case of Boolean automata networks, and is also known to be PSPACE-complete [19]. Given a reaction system (S, A) where S is a finite set of entities, and A is a set of reactions of the form (R, I, P) where R are the reactants, I the inhibitors and P the products, we construct the BAN of size $n = |S|$ with local functions:

$$\forall i \in [\![n]\!] : f_i(x) = \bigvee_{\substack{(R,I,P) \in A \\ \text{such that } i \in P}} \left(\bigwedge_{j \in R} x_j \wedge \bigwedge_{k \in I} \neg x_k \right).$$

A configuration $x \in \mathbb{B}^n$ of the BAN corresponds to a state of the reaction system with each automaton indicating the presence or absence of its corresponding entity. The parallel evolution of f (under $\mu_{\mathtt{par}}$) is in direct correspondance with the evolution of the reaction system. □

Proof of theorem 9. Given a formula ψ on n variables, m and i in unary, we apply Lemma 6 to construct g_n, μ_n on automata set $P = [\![q_{k_n}]\!]$. Automata from P have identity local functions, and the number of substeps is $\ell = |\varphi(\mu_n)| > 2^n$. Let p_i be the i-th prime number. We add:

- $\ell' = \lceil \log_2(\ell) \rceil$ automata numbered $B = \{q_{k_n}, \ldots, q_{k_n} + \ell' - 1\}$, implementing a ℓ' bits binary counter that increments modulo ℓ at each substep, except for configurations with a counter greater of equal to ℓ which are left unchanged.

- $\ell'' = \lceil \log_2(p_i) \rceil$ automata numbered $R = \{q_{k_n} + \ell', \dots, q_{k_n} + \ell' + \ell'' - 1\}$, whose local functions are:

$$f_R(x) = \begin{cases} x_R - m + 1 \mod p_i & \text{if } x_B = 0 \text{ and } x_B \text{ satisfies } \psi \\ x_R - m \mod p_i & \text{if } x_B = 0 \text{ and } x_B \text{ does not satisfy } \psi \\ x_R + 1 \mod p_i & \text{if } 0 < x_B < 2^n \text{ and } x_B \text{ satisfies } \psi \\ x_R & \text{otherwise.} \end{cases}$$

We also add singletons to μ_n for each of these additional automata, with $\mu' = \mu_n \cup \bigcup_{j \in B \cup R} \{(j)\}$. The resulting dynamics of $f_{\{\mu'\}}$ proceeds as follows.

Configurations x such that $x_B \geq \ell$ verify $f_{\{\mu'\}}(x) = x$, because all local functions are identities in this case. For configurations x such that $x_B < \ell$, during the dynamics of substeps from x to $f_{\{\mu'\}}(x)$, the counter x_B takes exactly once the values from 0 to $\ell - 1$, with $f_{\{\mu'\}}(x)_B = x_B$ (it goes back to its initial value). Meanwhile, at each substep with $x_B < 2^n$, the record of automata R is incremented if and only if x_B satisfies ψ, with a substraction of m when $x_B = 0$. Since $\ell > 2^n$ each valuation of ψ is checked exactly once, and x_R gets added the number of models of ψ minus m, modulo p_i (when $2^n \leq x_B < \ell$ automata R are left unchanged). Consequently, we have $f_{\{\mu'\}}(x)_R = x_R$ if and only if it has been incremented m times modulo p_i, i.e., f, μ' is a positive instance of **BP-Identity** if and only if ψ, m, i is a positive instance of **Mod-SAT** (the number of models of ψ is congruent to k modulo p_i). □

Proof of corollary 2. For a fixed reversible cellular automaton (of any dimension), given a configuration of size n and a time t, one can compute in polynomial time a block-parallel update schedule μ and circuits for the local functions of a Boolean automata network of large enough size (to encode the CA's state space in binary), such that:

- $|\varphi(\mu)| > t$ (by Lemma 6; these automata are left aside with identity local functions),
- one substep of $f_{\{\mu\}}$ simulates one step of the CA; and
- $f_{\{\mu\}}$ is bijective (because the CA is reversible, padding with identity).

This gives a functional Turing many-one reduction from Theorem 10. □