Complexity of Boolean Automata Networks Under Block-Parallel Update Modes

Kévin Perrot
Université publique, Marseille, France
Aix-Marseille Univ, CNRS, LIS, Marseille, France

Sylvain Sené
Université publique, Marseille, France
Aix-Marseille Univ, CNRS, LIS, Marseille, France

Léah Tapin
Aix-Marseille Univ, CNRS, LIS, Marseille, France

Abstract

Boolean automata networks (*aka* Boolean networks) are space-time discrete dynamical systems, studied as a model of computation and as a representative model of natural phenomena. A collection of simple entities (the automata) update their 0-1 states according to local rules. The dynamics of the network is highly sensitive to update modes, i.e., to the schedule according to which the automata apply their local rule. A new family of update modes appeared recently, called block-parallel, which is dual to the well studied block-sequential. Although basic, it embeds the rich feature of update repetitions among a temporal updating period, allowing for atypical asymptotic behaviors. In this paper, we prove that it is able to breed complex computations, squashing almost all decision problems on the dynamics to the traditionally highest (for reachability questions) class PSPACE. Despite obtaining these complexity bounds for a broad set of local and global properties, we also highlight a surprising gap: bijectivity is still coNP.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Problems, reductions and completeness

Keywords and phrases Boolean networks, finite dynamical systems, block-parallel update schedule

Digital Object Identifier 10.4230/LIPIcs.SAND.2024.19

Related Version *Full Version*: https://arxiv.org/abs/2402.06294

Acknowledgements This work received support from ANR-18-CE40-0002 FANs, STIC AmSud CAMA 22-STIC-02 (Campus France MEAE) and HORIZON-MSCA-2022-SE-01-101131549 ACANCOS projects.

1 Introduction

Automata networks are distributed models of computation, defined locally by means of individual entities (called automata) interacting with each other over discrete time, and collectively performing global computations. The model originates from the seminal work of McCulloch and Pitts on neural networks [32] (with local threshold Boolean functions). It raised fundamental complexity and computability questions on their dynamics, with notable considerations of feedback shift registers [26, 14], and perceptrons [39]. The Boolean case serves as a framework for biological modelling, as proposed by Kauffman and Thomas on gene regulation [28, 42], and repeatedly confirmed since the 1990s [33, 1, 19, 43] (where limit dynamics receive biological interpretations matching experiments).

Our contribution is at the frontier between theoretical computer science, discrete mathematics, and systems biology. When working on Boolean automata networks, it is utmost important to define the way automata update their state over time (namely the update mode), in order to obtain a discrete dynamical system. Indeed:

- Even if the fixed points obtained under the parallel update mode are fixed points obtained under any other update mode [20], specific update modes may generate additional fixed points (e.g., block-parallel).
- The limit cycles which are not fixed points, obtained under a given update mode, are not necessarily conserved under another update mode [11, 23, 5, 22, 6].

In other words, a Boolean automata network may admit a large number of distinct dynamics (depending on the update mode), which requires a strong attention, in particular when it is employed as a phenomenological model in systems biology. In the context of gene regulatory networks, chromatin dynamics has emerged as a full-fledged research track to understand the temporality of mRNA transcriptional machinery (which has no clear biological answer at present) [24, 7, 25, 16]. From a theoretical standpoint, advances on chromatin dynamics tend to show that genetic expression is neither purely asynchronous nor purely synchronous, hence supporting studies of in-between update modes.

In this line, this paper aims at studying the peculiar role and impact of block-parallel update modes, shown to have relevant features from both formal and applied standpoints [12, 36], in the sense that *(i)* they can generate fixed points which are not fixed points of the dynamical system obtained when the underlying network evolves synchronously, and *(ii)* they can implement specific biological timers which are intrinsically governed by phenomena exogenous to regulatory control. We take the lens of complexity theory, and provide ground results on classical decision problems related to fixed points and limit cycles, reachability, *etc.* These new complexity bounds highlight that most decision problems known to be NP-complete under block-sequential update modes, such as the image/preimage problems, and fixed point problems [17, 8, 35], are PSPACE-complete under block-parallel update modes. It suggests that the "expressivity" of such update modes comes at a high cost in terms of simulation, which strengthens the need for structural results. However, there are unexpected exceptions, related to bijectivity and steadyness.

In Section 2, we define formally the model and present known results. Section 3 exposes our results. Classical problems on computing images, preimages, fixed points and limit cycles are characterized: they all jump from NP (under block-sequential update modes) to PSPACE (under block-parallel update modes). Then we 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. In Section 4, we summarize the results and expose perspectives.

2 Definitions and state of the art

We denote the set of integers by $[\![n]\!] = \{0, \ldots, n-1\}$, the Booleans by $\mathbb{B} = \{\mathtt{0}, \mathtt{1}\}$, 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 $x_I \in \mathbb{B}^{|I|}$. 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 [4] and characterized in [3] 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 [37] (how to parallelize a block-sequential update schedule), [22] (the preservation of cycles throughout the parallelization process), and [9] (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 $\ell = \text{lcm}(n_1, \dots, n_s)$, as $f_{\{\mu\}}(x) = f_{(W_{\ell-1})} \circ \dots \circ f_{(W_1)} \circ f_{(W_0)}(x)$, where for all $i \in [\![\ell]\!]$ we define $W_i = \{i^k_{i \bmod n_k} \mid k \in [s]\}$. 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 [36]). We defined this map as φ:

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

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

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

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

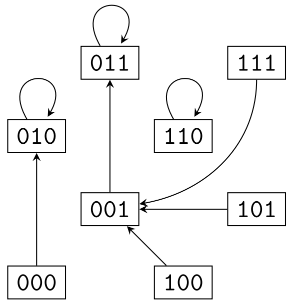

Figure 1 Example of an automata network of size $n = 3$ with a block-parallel update mode $\mu \in \mathsf{BP}_n$. Local functions (upper left), conversion of μ to a sequence of blocks (lower left), and dynamics of $f_{\{\mu\}}$ on configuration space $\mathbb{B}^3$ (right). One step is composed of two 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`.

Block-parallel update schedules have been introduced in [12], motivated by applications to gene regulatory networks, and their ability to generate new stable configurations (compared to block-sequential update schedules). A first theoretical study has been conducted in [36], providing counting formulas and enumeration algorithms, subject to equivalence relations on the produced dynamics.

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 $\forall i \in [\![p]\!] : f_{\{\mu\}}(x^i) = x^{i+1 \mod p}$. 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).

The computational complexity of finite discrete dynamical systems has been explored on the related models of finite cellular automata [40] and reaction networks [13]. Regarding automata networks, fixed points received early attention in [2] and [17], with existence problems complete for NP. Because of the fixed point invariance for block-sequential update schedules [38], the focus switched to limit cycles [6, 8], with problems reaching the second level of the polynomial hierarchy. The interplay of different update schedules has been investigated in [6]. Finaly, let us mention the general complexity lower bounds, established for any first-order question on the dynamics, under the parallel update schedule [18].

3 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 3.1). Then we start with results on computing images, preimages, fixed points and limit cycles (Subsection 3.2), before studying reachability and global properties of the function $f_{\{\mu\}}$ computed by an automata network f under block-parallel update schedule μ (Subsection 3.3).

3.1 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 [21].

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 1** (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 2.** *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 2, 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 4.** *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 2 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 2 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 5.** Given $n \geq 2$, consider g_n and μ_n given by Lemma 4. 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 2 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 $0^{q_{k_n}} 1^n$.

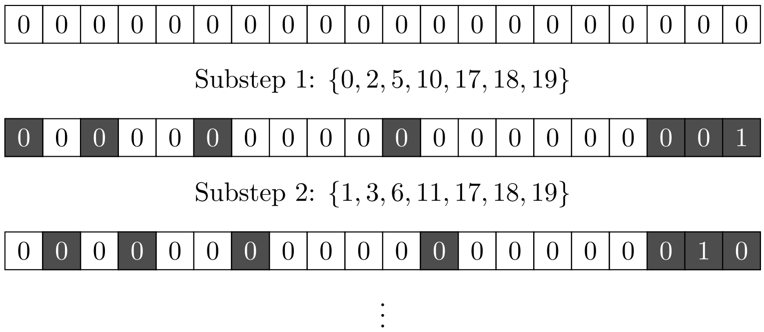

Figure 2 Substeps leading to the image of configuration $\mathtt{0}^{q_{k_n}}\mathtt{010}$ in $f_{\{\mu'\}}$ from Example 5 for $n = 3$ ($k_n = 4$ and $q_{k_n} = 2 + 3 + 5 + 7 = 17$). The last 3 bits implement a binary counter, freezing at 7 ($\mathtt{111}$). Above each substep the block of updated automata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

3.2 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 6.** **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 4 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 $\ell = |\varphi(\mu_n)| > 2^n$ 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 $\mathtt{1}$ 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 $x = \mathtt{0}^{q_{k_n}} \mathtt{0}^{\ell'} \tilde{x} \mathtt{0}$. 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 4), 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 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 7.** **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 6).

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 6, 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 $x = 0^{q_{k_n}} 0^{\ell'} \tilde{x} 0$ is a fixed point of $f_{\{\mu'\}}$. Indeed, during the ℓ-th and last substep, the primes P are still in state $0^{q_{k_n}}$, the counter B goes back to 0 (state $0^{\ell'}$), 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 $0^{q_{k_n}} 0^{\ell} \tilde{x} 0$ 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 $\mathtt{1}$ 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 $(\mathtt{0}^{q_{k_n}} B_i \tilde{x} \mathtt{0})_{i \in [\![k]\!]}$ 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 $x^0, \ldots, x^{p-1}$ where it is furthermore required that all configurations are different (this corresponds to having the minimum length p): the problem is still PSPACE-complete.

3.3 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 [18], 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 3.2, 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 6).

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 $G \sqsubset f_{\{\mu\}}$?

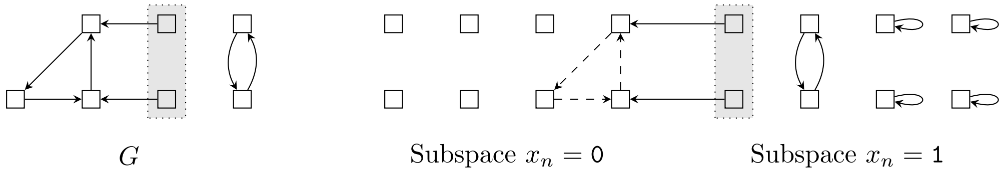

Figure 3 Construction of g in the proof of Theorem 8. Subspace $x_n = 0$ contains a copy of f with a potential limit cycle dashed. Subspace $x_n = 1$ implements G', and wires configurations of U (grey area) to the potential limit cycle in the copy of f (remaining configurations are fixed points).

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 a functional graph: it is necessarily induced since there is no arc to delete.

▶ **Theorem 8.** **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 [10]).

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 7 for the problem **BP-Limit-Cycle-*k***. We have that $f_{\{\mu\}}$ has a limit cycle of length k on configurations $(0^{q_{k_n}} B_i \tilde{x} 0)_{i \in [\![k]\!]}$ (or configuration $0^{q_{k_n}} 0^{\ell} \tilde{x} 0$ 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 $x_n = 0$, and a copy of G' on the subspace $x_n = 1$ 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 $x_n = 0$). Other configurations in the subspace $x_n = 1$ will be fixed points. Figure 3 illustrates the construction. Recall that G is fixed, and consider a mapping $\alpha : V(G) \to \{0, 1\}^n$ such that vertices of the limit cycle of length k are sent to the configurations $(0^{q_{k_n}} B_i \tilde{x} 0)_{i \in [\![k]\!]}$ respectively (or $0^{q_{k_n}} 0^{\ell} \tilde{x} 0$ 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 $(0^{q_{k_n}} B_i \tilde{x} 0)_{i \in [\![k]\!]}$ (or

$0^{q_{k_n}} 0^\ell \tilde{x} 0$ 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 8, 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 9.** *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 9 shows that bijectivity can be decided at the local level of circuits (not iterated), which can be checked in coNP and gives Theorem 10.

▶ **Theorem 10.** **BP-Bijectivity** *is* coNP*-complete.*

Proof. A coNP algorithm can be established from Lemma 9, 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 [35, 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 [35, 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 11.** **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 10, "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 [29].

▶ **Theorem 12.** **BP-Identity** *is* ModP*-hard (for polytime many-one reduction).*

Proof. Given a formula ψ on n variables, m and i in unary, we apply Lemma 4 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 $B = \{q_{k_n}, \dots, q_{k_n} + \ell' - 1\}$, 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). ◀

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** [31]. 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 [31] and [34] 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 [27, 41], 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 13** ([15, 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 14.** *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* $\mathsf{FP}^{\mathsf{PSPACE}}$*.*

Proof. 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 4; 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 13. ◀

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 (image through $f_{\{\mu\}}$), in order to reach **BP-Identity**, 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 $x \in \mathbb{B}^n$?

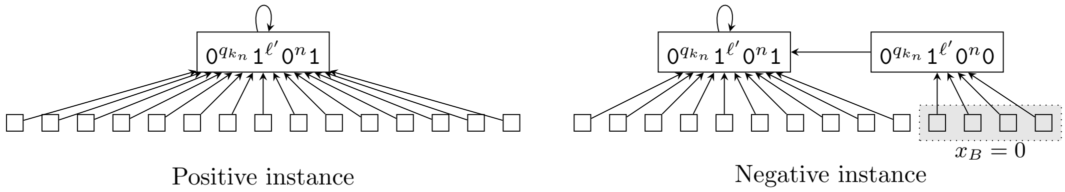

Figure 4 Illustration of the dynamics obtained for the reduction to **BP-Constant** in the proof of Theorem 15. Configurations x with the counter automata B initialized to $x_B = 0$ either go to $0^{q_{k_n}}1^{\ell'}0^n1$ (left, positive instance), or to $0^{q_{k_n}}1^{\ell'}0^n0$ (right, negative instance). Only the bit of automata R changes.

▶ **Theorem 15.** **BP-Constant** *is* PSPACE*-complete.*

Proof. To decide **BP-Constant**, one can simply enumerate all configurations and compute their image (Theorem 6) 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 4 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 $\ell = |\varphi(\mu_n)| > 2^n$. We add (Figure 4 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 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.

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

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

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. ◀

4 Conclusion and perspectives

Block-sequential update schedules have a number of substeps limited by the fact that every automaton is updated only once. Block-parallel update schedules overcome this restriction, thus significantly raising the n (number of automata) upper bound for the number of substeps (Lemma 4 gives a backbone construction with more than 2^n substeps). This greatly increases the expressiveness of block-parallel dynamics, and we have demonstrated that this gain in computational power comes along with higher complexity costs. A fundamental point is that computing a single transition becomes PSPACE-hard in this context (Theorem 6), whereas it is feasible in polynomial time for all block-sequential update schedules [37]. We derive multiple consequences on the PSPACE-completeness of classical decision problems related to the existence of preimages, fixed points, limit cycles, and the recognition of constant dynamics. These problems are NP-complete (existence problems), or coNP-complete (global dynamical properties) for block-sequential modes (see [35]), hence one might be tempted to extrapolate to the following conjecture, which is false (unless a drastic complexity collapse).

▶ **Conjecture 16** (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.*

The recognition of bijective dynamics disproves Conjecture 16: according to Lemma 9, a single substep is necessary and sufficient to break the bijectivity of the automata network's dynamics, hence bringing the question to the circuit level (of substeps), in coNP. It also prevents to level the Rice-like complexity lower bound theorem presented in [18], to PSPACE-hardness. Recognition problems are nonetheless still NP-hard or coNP-hard for non-trivial first order questions, because parallel is a particular case of block-parallel.

The reachability problem, which is PSPACE-complete for block-sequential modes, remains PSPACE-complete for block-parallel modes (Theorem 18). Intuitively, on the one hand the idea of reachability can be embedded in a single transition step of block-parallel update, because it may have an exponential number of substeps. On the other hand, the sequence of reachability problems at the level of substeps combines into a reachability problem at the level of steps which is still in PSPACE.

The recognition of identity dynamics is not fully characterized (Open problem 11 and Theorem 12). A fine interplay between computing in a reversible setting (since non-bijective dynamics can be identified in NP) and the length of limit cycles in the dynamics of substeps (to loop back to the starting configuration and be the identity map) is still to be discovered. Computing the interaction graph (feasible in DP, just above NP and coNP) may give some insights but, contrary to block-sequential modes having identity dynamics if and only if the interaction graph is made of n positive loops, it is possible to design more complex identity dynamics under block-parallel update schedules.

After determining the complexity of recognizing preimages, image points or fixed points in Subsection 3.2, the next logical step would be the complexity of counting them. This is not an easy step to make from the constructions presented in the present work, which are not parcimonious (for the definition of #PSPACE, see [30]).

An important remark for the community is that, while all the proofs in this paper were written with Boolean automata networks in mind, the results also hold for non-Boolean automata networks.

Another avenue of research could be questions about the existence of a block-parallel update schedules verifying a certain property, as in [8, 6] for block-sequential update schedules. Given that the fixed point invariance is broken under block-parallel update schedules, it opens the way for more questions. The ability to create new fixed points (how and when does it happen?) is in itself a meaningful track of research.

References

1 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.

2 N. Alon. Asynchronous threshold networks. *Graphs and Combinatorics*, 1:305–310, 1985. doi:10.1007/BF02582959.

3 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.

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

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

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

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

8 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.

9 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. doi:10.1007/978-3-319-55911-7_9.

10 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.

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

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

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

14 B. Elspas. The theory of autonomous linear sequential networks. *IRE Transactions on Circuit Theory*, 6:45–60, 1959.

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

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

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

18 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.

19 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.

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

21 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.

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

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

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

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

26 D. A. Huffman. Canonical forms for information-lossless finite-state logical machines. *IRE Transactions on Information Theory*, 5:41–59, 1959.

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

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

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

30 R. E. Ladner. Polynomial Space Counting Problems. *SIAM Journal on Computing*, 18(6):1087–1097, 1989. doi:10.1137/0218073.

31 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.

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

33 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.

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

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

36 K. Perrot, S. Sené, and L. Tapin. Combinatorics of block-parallel automata networks. In *Proceedings of SOFSEM'24*, LNCS. Springer, 2024. To appear.

37 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.

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

39 F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

40 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.

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

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

43 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.

A Omitted proofs

Proof of Lemma 2. 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 17.** **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 6), 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 6, and modify the local functions of automata D and R as follows:

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

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 1 (including the initial substep). We set $y = 0^{q_{k_n}} 0^\ell 0^n 1$.

If the **Iter-CVP** instance is positive, then we have $f_{\{\mu'\}}(0^{q_{k_n}} 0^\ell 0^n 0) = y$ (automata B go back to $0^{q_{k_n}}$, automata D iterate circuit C from $\tilde{x}$ and end in state 0^n, and automaton R has recorded that the i-th bit of D has been to state 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 $x_B = 0^{q_{k_n}}$, 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 0^n, and that automaton R records the answer to the **Iter-CVP** instance. Since it it ends in state $y_R = 1$ 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 18.** **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 [40]. 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 [13]. 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 $\mu_{\texttt{par}}$) is in direct correspondance with the evolution of the reaction system. ◀