HAL open science

Boolean Networks: Beyond Generalized Asynchronicity

Thomas Chatain, Stefan Haar, Loïc Paulevé

To cite this version:

Thomas Chatain, Stefan Haar, Loïc Paulevé. Boolean Networks: Beyond Generalized Asynchronicity. AUTOMATA 2018, Jun 2018, Ghent, Belgium. hal-01768359v1

HAL Id: hal-01768359

https://hal.science/hal-01768359v1

Submitted on 17 Apr 2018 (v1), last revised 21 Jun 2018 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Boolean Networks: Beyond Generalized Asynchronicity

Thomas Chatain[1], Stefan Haar[1], and Loïc Paulevé[2]

[1] LSV, ENS Paris-Saclay, INRIA, CNRS, France
[2] CNRS & LRI UMR 8623, Univ. Paris-Sud – CNRS, Université Paris-Saclay, 91405 Orsay, France

Abstract. Boolean networks are commonly used in systems biology to model dynamics of biochemical networks by abstracting away many (and often unknown) parameters related to speed and species activity thresholds. It is then expected that Boolean networks produce an over-approximation of behaviours (reachable configurations), and that subsequent refinements would only prune some impossible transitions.
However, we show that even generalized asynchronous updating of Boolean networks, which subsumes the usual updating modes including synchronous and fully asynchronous, does not capture all transitions doable in a multi-valued or timed refinement.
We define a structural model transformation which takes a Boolean network as input and outputs a new Boolean network whose asynchronous updating simulates both synchronous and asynchronous updating of the original network, and exhibits even more behaviours than the generalized asynchronous updating. We argue that these new behaviours should not be ignored when analyzing Boolean networks, unless some knowledge about the characteristics of the system explicitly allows one to restrict its behaviour.

1 Introduction

Boolean networks model dynamics of systems where several components (or nodes) interact. They specify for each node an update function to determine its next value according to the configuration (global state) of the network. Boolean networks are widely used to model dynamics of biological networks, such as gene networks and cellular signalling pathways.

The scheduling of nodes updates is known to have a strong influence on the reachable configurations of the networks. The relationships between different updating modes received a lot of attentions both in transition-centered models of networks such as Petri nets [14,6,8,27,28] (in particular when read arcs are used to model finely the update mechanisms), and function-centered models such as cellular automata [22,5] and Boolean networks [15,25,12,3,18,19], on which this article is focused. Notice that transformations exist from BNs to Petri nets [23,9,10] showing the strong relationship between the two formalisms.

For Boolean networks, the considered updating modes are usually the following: the *synchronous* updating, where all nodes are updated simultaneously, generating a deterministic dynamics; the (fully) *asynchronous* updating, where only one node can be updated at a time, this node being chosen non-deterministically. Asynchronous updating generates non-deterministic dynamics due to the different ordering of updates, which can be interpreted as considering in the same model different speed of updates. Then, the *generalized asynchronous* updating allows all the combinations of simultaneous updates subsets of nodes, ranging from single nodes (matching asynchronous transitions) to the full set of nodes (matching synchronous transitions). Other updating modes like sequential or block sequential have also been considered in the literature on cellular automata and Boolean networks [5,3], and usually lead to transitions allowed by the generalized asynchronous updating.

When a Boolean network aims at modelling a dynamical system having time features, as it is typically the case for biological systems, the choice of the update mode is crucial as it determines the set of configurations reachable from a given initial configuration. In applications, it is usual to assess the accordance of a Boolean network with the concrete system by checking if the observed configuration are indeed reachable in the Boolean network. Whenever it is not the case, it typically means that the designed Boolean functions do not model the system correctly, and thus should be modified before further model analysis.

Having very partial information on the actual velocity of different nodes and transitions in the concrete system, a common approach is to choose the most general updating mode, i.e., the one bringing the fewer constraints as possible regarding the unknown scheduling of node updates. In such a setting, and because we abstract away many parameters of the system dynamics, we expect that the Boolean network models an over-approximations of possible transitions, *i.e.*, that any reachable configuration in the concrete system should be reachable in the Boolean network.

In this paper, we show that the generalized asynchronous updating, subsuming synchronous and asynchronous updating, can miss transitions, hence reachable configurations, which correspond to particular, but plausible, behaviours. Thus, the resulting analysis can be misleading on the absence of some behaviours, notably regarding the reachability of attractors (configurations reachable on the long-run), and may lead to reject valid models.

We introduce a new updating mode for Boolean networks, so-called *interval semantics* which aims at enabling the reachability of configurations by considering further update scheduling policy. Essentially, the interval semantics considers the possibility of a delay between the trigger of the update of a node, and its actual completion: this models species for which value changes can be slow.

The interval semantics can be expressed as the asynchronous updating over a Boolean network which encodes the decoupling of update triggering and update application. Therefore, our approach allows the definition of an asynchronous Boolean network which simulates the general asynchronous dynamics of the original Boolean network, while including additional and plausible behaviours, and

still preserving important dynamical constraints on fixpoints and causality of transitions: the fixpoints of the interval semantics form a one-to-one relationship with the fixpoints of the generalized asynchronous updating, and it preserves the influence graph, notably its cycles and their signs.

We illustrate the benefit of the interval semantics on a small example of Boolean network, which is actually embedded in many models of biological networks (e.g., [16,17,26]). Therefore, the analysis of dynamics of these biological models can be substantially impacted by considering the interval semantics.

Outline. Sect. 2 gives the definitions of Boolean networks and their synchronous, asynchronous, and generalized asynchronous updating, as well as their influence graph. Sect. 3 gives a motivating example showing the limit of the generalized asynchronous updating. Sect. 4 introduces the interval semantics for Boolean networks by providing an encoding as an asynchronous Boolean network and by establishing the relation with the generalized asynchronous updating and consistency criteria. Further extensions of the interval semantics are discussed in Sect. 5. Finally, Sect. 6 discusses the relevance of the results for the analysis of biological models, and suggests further work.

2 Definitions

We write $\mathbb{B} = \{0, 1\}$ and $[n] = \{1, \ldots, n\}$. Given a *configuration* $x \in \mathbb{B}^n$ and $i \in [n]$, we denote x_i the ith component of x, so that $x = x_1 \ldots x_n$. Given two configurations $x, y \in \mathbb{B}^n$, the components that differ are noted $\Delta(x, y) \stackrel{\Delta}{=} \{i \in [n] \mid x_i \neq y_i\}$.

Definition 1 (Boolean network). *A Boolean network (BN) of dimension n is a collection of functions $f = \langle f_1, \ldots, f_n \rangle$ where $\forall i \in [n], f_i : \mathbb{B}^n \to \mathbb{B}$.*

Given $x \in \mathbb{B}^n$, we write $f(x)$ for $f_1(x) \ldots f_n(x)$.

Fig. 1 (a) shows an example of BN of dimension 3.

When modelling biological systems, each node $i \in [n]$ usually represents a biochemical species, being either active (or present, value 1) or inactive (or absent, value 0). Each function f_i indicates how the evolution of the value of i is influenced by the current value of other components $j \in [n]$. However, this description can be interpreted in several ways, therefore several updating mode coexist for BNs, depending on the assumptions about the order in which the evolutions predicted by the f_i apply.

The *asynchronous updating* assumes that only one component is updated at each time step. The choice of the component to update is non deterministic.

Definition 2 (Asynchronous updating). *Given a BN f, the binary irreflexive relation $\xrightarrow[\text{async}]{f} \subseteq \mathbb{B}^n \times \mathbb{B}^n$ is defined as:*

$$x \xrightarrow[\text{async}]{f} y \stackrel{\Delta}{\Longleftrightarrow} \exists i \in [n], \Delta(x, y) = \{i\} \wedge y_i = f_i(x) \ .$$

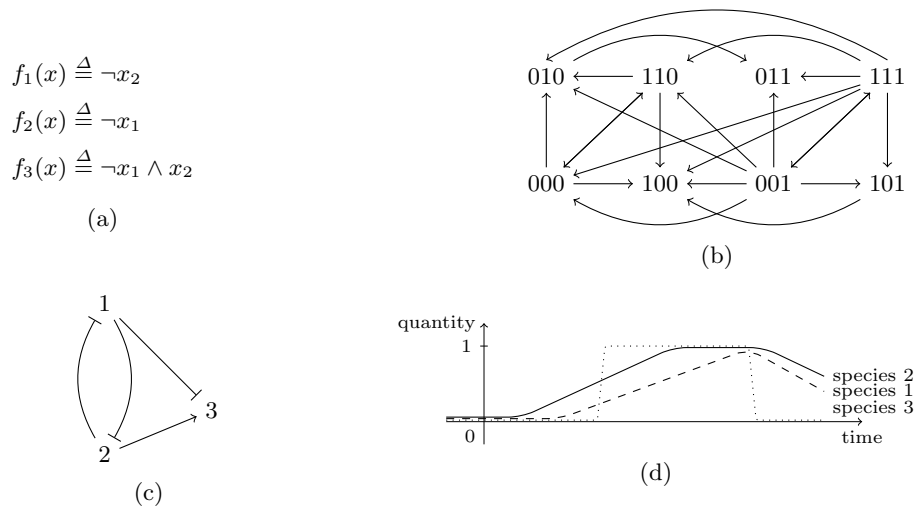

Fig. 1. (a) Example BN f of dimension 3; (b) Transition relations between configurations in $\mathbb{B}^3$ according to the generalized asynchronous updating of f; (c) Influence graph $G(f)$; positive edges are with normal tip; negative edges are with bar tip; (d) A possible evolution of the quantities of the species (species 1 in dashed line, species 2 plain, species 3 dotted).

We write $\xrightarrow[\text{async}]{f}{}^*$ *for the transitive closure of* $\xrightarrow[\text{async}]{f}$.

The *synchronous updating* can be seen as the opposite: *all* components are updated at each time step. This leads to a purely deterministic dynamics.

Definition 3 (Synchronous updating). *Given a BN f, the binary irreflexive relation* $\xrightarrow[\text{sync}]{f} \subseteq \mathbb{B}^n \times \mathbb{B}^n$ *is defined as:*

$$x \xrightarrow[\text{sync}]{f} y \overset{\Delta}{\Longleftrightarrow} x \neq y \wedge \forall i \in [n], y_i = f_i(x) \ .$$

By forcing all the components to evolve synchronously, the synchronous updating makes a strong assumption on the dynamics of the system. In many concrete cases, for instance in systems biology, this assumption is clearly unrealistic, at least because the components model the quantity of some biochemical species which evolve at different speeds.

As a result, the synchronous updating fails to describe some behaviours, like the transition $010 \to 011$ represented in Fig. 1 (b) which represents the activation of species 3 when species 1 is inactive and species 2 is active ($f_3(010) = 1$). There are also transitions which are possible in the synchronous but not in the asynchronous updating, for instance $000 \to 110$. Remark that 110 is not even reachable from 000 in the asynchronous updating.

The *generalized asynchronous updating* generalizes both the asynchronous and the synchronous ones: it allows updating synchronously any nonempty subset of components.

Definition 4 (Generalized asynchronous updating). *Given a BN f, the binary irreflexive relation $\xrightarrow{f} \subseteq \mathbb{B}^n \times \mathbb{B}^n$ is defined as:*

$$x \xrightarrow{f} y \overset{\Delta}{\Longleftrightarrow} x \neq y \wedge \forall i \in \Delta(x,y) : y_i = f_i(x) \ .$$

Clearly, $x \xrightarrow[\text{async}]{f} y \Rightarrow x \xrightarrow{f} y$ and $x \xrightarrow[\text{sync}]{f} y \Rightarrow x \xrightarrow{f} y$. The converse propositions are false in general. It is even false that $x \xrightarrow{f} y$ implies $x \xrightarrow[\text{async}]{f} y \vee x \xrightarrow[\text{sync}]{f} y$. Note that we forbid "idle" transitions ($x \to x$) whatsoever the updating mode.

For each node $i \in [n]$ of the BN, f_i typically depends only on a subset of nodes of the network. The *influence graph* of a BN (also called interaction or causal graph) summarizes these dependencies by having an edge from node j to i if f_i depends on the value of j. Formally, f_i depends on x_j if there exists a configuration $x \in \mathbb{B}^n$ such that $f_i(x)$ is different from $f_i(x')$ where x' is x having solely the component j different ($x'_j = \neg x_j$). Moreover, assuming $x_j = 0$ (therefore $x'_j = 1$), we say that j has a positive influence on i (in configuration x) if $f_i(x) < f_i(x')$, and a negative influence if $f_i(x) > f_i(x')$. It is possible that a node has different signs of influence on i in different configurations (leading to non-monotonic f_i). It is worth noticing that different BNs can have the same influence graph.

Definition 5 (Influence graph). *Given a BN f, its* influence graph *$G(f)$ is a directed graph $([n], E_+, E_-)$ with* positives *and* negatives *edges such that*

$$(j,i) \in E_+ \overset{\Delta}{\Longleftrightarrow} \exists x, y \in \mathbb{B}^n : \Delta(x,y) = \{j\}, x_j < y_j, f_i(x) < f_i(y)$$
$$(j,i) \in E_- \overset{\Delta}{\Longleftrightarrow} \exists x, y \in \mathbb{B}^n : \Delta(x,y) = \{j\}, x_j < y_j, f_i(x) > f_i(y)$$

A (directed) cycle composed of edges in $E_+ \cup E_-$ is said positive *when it is composed by an even number of edges in E_- (and in number of edges in E_+), otherwise, it is* negative.

The influence graph is an important object in the literature of BNs [24,2]. For instance, many studies have shown that one can derive dynamical features of a BN f by the sole analysis of its influence graph $G(f)$. Importantly, the presence of negative and positive cycles in the influence graph, and the way they are intertwined can help to determine the nature of attractors (that are the smallest sets of configurations closed by the transition relationship) [21], and derive bounds on the number of fixpoints and attractors a BN having the same influence graph can have [20,1,4].

3 Motivating example

Fig. 1 shows an example of BN of dimension 3, its influence graph and $\xrightarrow{f}$ relation between configurations. The BN and its influence graph show that the quantity of 3 increases when 1 is absent and 2 is present. In any scenario starting from 000 where 3 eventually increases, 2 has to increase to trigger the increase of 3. Hence, according to the generalized asynchronous updating represented in Fig. 1 (c), the only transition which represents an increase of 3 is $010 \rightarrow 011$. After this, no transition is possible.

But, assuming the BN abstracts continuous evolution of quantities, the following scenario, pictured in Fig. 1(d), becomes possible: initially, the absence of species 1 causes an increase of the quantity of species 2, represented in plain line on the figure. Symmetrically, the absence of species 2 causes an increase of the quantity of species 1 (dashed line). This corresponds to the evolution described by the arrow $000 \rightarrow 110$ in Fig. 1(b) and leads to a (transient) configuration where species 1 and 2 are present.

Assume that 1 and 2 increase slowly. After some time, however, the quantity of 2 becomes sufficient for influencing positively the quantity of 3, while there is still too little of species 1 for influencing negatively the quantity of 3. Species 3 can then increase. In the scenario represented in the figure, 3 (dotted line) increases quickly, and then 1 and 2 continue to increase. In summary, the quantity of species 3 increased from 0 to 1 *during* the increase of 1 and 2, which was not predicted by the generalized asynchronous updating (Fig. 1(b)).

One could argue that in this case, one should better consider more fine-grained models, for instance by allowing more than binary values on nodes in order to reflect the different activation thresholds. However, the definition of the refined models would require additional parameters (the different activation thresholds) which are unknown in general. Our goal is to allow capturing these behaviours already in the Boolean abstraction, so that any refinement would remove possible transitions, and not create new ones.

4 Interval Semantics for Boolean Networks

Interval semantics has been proposed for Petri nets in [11] with the aim at generalizing the notion of steps [13], that are sets of transitions that can be simultaneously fired. The interval semantics adds the possibility to trigger, within a single step, transitions that become enabled by the firing transitions. The motivating example given in the previous section illustrates how this semantics can augment the set of reachable configurations.

In this section, we propose an encoding of the interval semantics for Boolean networks as an asynchronous Boolean network. Essentially, each node $i \in [n]$ is decoupled in two nodes: a "write" node storing the next value $(2i-1)$ and a "read" node for the current value $(2i)$. The decoupling is used to store an ongoing value change, while other nodes of the system still read the current (to be changed) value of the node. A value change is then performed according to

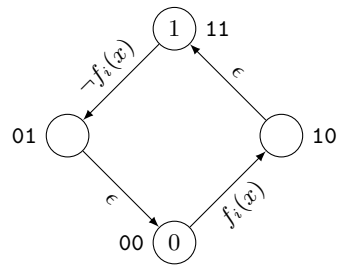

Fig. 2. Automaton of the value change of a node i in the interval semantics. The states marked 0 and 1 represents the value 0 and 1 of the node. The labels $f_i(x)$ and $\neg f_i(x)$ on edges are the conditions for firing the transitions; ϵ indicates that the transitions can be done without condition. The states are labeled by the corresponding values of nodes $(2i-1)(2i)$ in our encoding.

the automaton given in Fig. 2: assuming we start in both write and read node with value 0, if $f_i(x)$ is true, then the write node is updated to value 1. The read node is updated in a second step, leading to the value where both write and read nodes are 1. Then, if $f_i(x)$ is false, the write node is updated first, followed, in a second stage by the update of the read node.

Once the write node $(2i-1)$ has changed its value, it can no longer revert back until the read node has been updated. Hence, if $f_i(x)$ become false in the intermediate value 10, the read node will still go through value 1 (possibly enabling transitions) before the write node can be updated to 0, if still applicable.

4.1 Encoding

From the automaton given in Fig. 2, one can derive Boolean functions for the write $(2i-1)$ and read $(2i)$ nodes. It results in the following BN $\tilde{f}$, encoding the interval semantics for the BN f:

Definition 6 (Interval semantics for Boolean networks). *Given a BN f of dimension n, $\tilde{f}$ is a BN of dimension $2n$ where $\forall i \in [n]$,*

$$\tilde{f}_{2i-1}(z) \triangleq (f_i(\gamma(z)) \wedge (\neg z_{2i} \vee z_{2i-1})) \vee (\neg z_{2i} \wedge z_{2i-1})$$
$$\tilde{f}_{2i}(z) \triangleq z_{2i-1}$$

where $\gamma(z) \in \mathbb{B}^n$ is defined as $\gamma(z)_i \triangleq z_{2i}$ for every $i \in [n]$.
Given $x \in \mathbb{B}^n$, $\alpha(x) \in \mathbb{B}^{2n}$ is defined as $\alpha(x)_{2i-1} = \alpha(x)_{2i} \triangleq x_i$ for every $i \in [n]$.
A configuration $z \in \mathbb{B}^{2n}$ is called consistent *when $\alpha(\gamma(z)) = z$.*

The function $\gamma : \mathbb{B}^{2n} \to \mathbb{B}^n$ maps a configuration of the interval semantics to a configuration of the BN f by projecting on the read nodes. The function $\alpha : \mathbb{B}^n \to \mathbb{B}^{2n}$ gives the interval semantics configuration of a configuration of the Boolean network f, where the read and write nodes have a consistent value.

Example 1. Applied to the BN f of Fig. 1, we obtain the following possible sequence of asynchronous iterations of $\tilde{f}$:

$$00\,00\,00 \xrightarrow[\text{async}]{\tilde{f}} 10\,00\,00 \xrightarrow[\text{async}]{\tilde{f}} 10\,10\,00 \xrightarrow[\text{async}]{\tilde{f}} 10\,11\,00$$
$$\xrightarrow[\text{async}]{\tilde{f}} 10\,11\,10 \xrightarrow[\text{async}]{\tilde{f}} 10\,11\,11 \xrightarrow[\text{async}]{\tilde{f}} 11\,11\,11$$

Therefore, with the interval semantics, the configuration 111 of f is reachable from 000, contrary to the generalized asynchronous semantics. This is due to the decoupling of the update of node 1: the activation of 1 is delayed which allows activating node 3 beforehand.

4.2 Asynchronous Weak Simulation of Generalized Asynchronous

The following theorem configurations that any transition of the generalized asynchronous semantics can be simulated by the interval semantics.

Theorem 1. *For all $x, y \in \mathbb{B}^n$,*

$$x \xrightarrow{f} y \Rightarrow \alpha(x) \xrightarrow[\text{async}]{\tilde{f}}{}^{*} \alpha(y) \ .$$

Proof. By decomposition along $\Delta(x, y)$: first, for each $i \in \Delta(x, y)$, update the $(2i-1)$-th component: we obtain after i asynchronous steps $z \in \mathbb{B}^{2n}$ where $z_{2i-1} = y_i$. Indeed, remark that $f_i(\gamma(z)) = f_i(x)$ and, as $y_i \neq x_i$, $f_i(\gamma(z)) = \neg z_{2i}$, therefore $\tilde{f}_{2i-1}(z) = \neg z_{2i} = f_i(x) = y_i$. Then, update all $(2i)$-th components, leading to $z' \in B^{2n}$ with $z'_{2i} = z'_{2i-1} = y_i$, thus $\alpha(y) = z'$. □

4.3 Consistency

The above theorem shows that the asynchronous semantics of the Boolean network encoding our interval semantics can reproduce any behaviour of the generalized asynchronous semantics. The aim of this section is to show that the interval semantics still preserves important constraints of the BN on its dynamics. In particular, we show the one-to-one relationship between the fixpoints of the BN and its encoding for interval semantics; and that the influences are preserved with their sign.

Lemma 1 states that from any configuration of encoded BN, one can always reach a configuration which corresponds to a configuration of the original BN (i.e., a configuration $z \in \mathbb{B}^{2n}$ such that $\alpha(\gamma(z)) = z$):

Lemma 1 (Reachability of consistent configurations). *For any $z \in \mathbb{B}^{2n}$ such that $\alpha(\gamma(z)) \neq z$, $\exists y \in \mathbb{B}^n : z \xrightarrow[\text{async}]{\tilde{f}}{}^{*} \alpha(y)$.*

Proof. For each $i \in [n]$ such that $z_{2i-1} \neq z_{2i}$, we update the $2i$ node, in whatever order. This leads to the configuration $z' \in \mathbb{B}^{2n}$ where $\forall i \in [n]$, $z'_{2i} = z'_{2i-1} = z_{2i-1}$. Hence, by picking $y = \gamma(z)$, we obtain $z \xrightarrow[\text{async}]{\tilde{f}}{}^{*} \alpha(y)$. □

The one-to-one relationship between fixpoints of f and fixpoints of $\tilde{f}$ is given by the following lemma:

Lemma 2 (Fixpoint equivalence). *$\forall x \in \mathbb{B}^n$, $f(x) = x \Rightarrow f(\alpha(x)) = \alpha(x)$; and $\forall z \in \mathbb{B}^{2n}$, $\tilde{f}(z) = z \Rightarrow \alpha(\gamma(z)) = z \wedge f(\gamma(z)) = \gamma(z)$.*

Proof. Let $x \in \mathbb{B}^n$ be such that $f(x) = x$. We have that $\alpha(x)_{2i-1} = \alpha(x)_{2i} = x_i = f_i(x)$. Hence, $\tilde{f}_{2i-1}(\alpha(x)) = f_i(\gamma(\alpha(x))) = f_i(x) = \alpha(x)_{2i-1}$; and $\tilde{f}_{2i}(\alpha(x)) = \alpha(x)_{2i-1} = \alpha(x)_{2i}$. Thus, $\tilde{f}(\alpha(x)) = \alpha(x)$.

Let $z \in \mathbb{B}^{2n}$ be such that $\tilde{f}(z) = z$. For each $i \in [n]$, because $\tilde{f}_{2i}(z) = z_{2i}$, by the definition of $\tilde{f}_{2i}$, we obtain that $z_{2i} = z_{2i-1}$. Thus, $\alpha(\gamma(z)) = z$. Moreover, as $(\neg z_{2i} \vee z_{2i-1})$ reduces to true and $(\neg z_{2i} \wedge z_{2i-1})$ reduces to false, $\tilde{f}_{2i-1}(z) = f_i(\gamma(z)) = z_{2i-1} = \gamma(z)_i$. Therefore, $f(\gamma(z)) = \gamma(z)$. □

Influence graph As defined in Sect. 2, the influence graph provides a summary of the causal dependencies between the value changes of nodes of the BN. We show that our encoding of interval semantics preserves the causal dependencies of the original network, and in particular, preserves the cycles and their signs.

From the definition of $\tilde{f}$, one can derive that all the influences in f are preserved in $\tilde{f}$, and no additional influences between different variables i, j are created by the encoding. This latter fact is addressed by the following lemma:

Lemma 3. *For any $i, j \in [n]$, $i \neq j$, there is a positive (resp. negative) edge from j to i in $G(f)$ if and only if there is a positive (resp. negative) edge from $2j$ to $2i-1$ in $G(\tilde{f})$.*

Proof. Let us define $x, y \in \mathbb{B}^n$ such that $\Delta(x, y) = \{j\}$, and $z, z' \in \mathbb{B}^{2n}$ such that $z = \alpha(x)$ and $\Delta(z, z') = \{2j\}$, i.e., $z'_{2j} = y_j$. Because $z_{2i} = z_{2i-1}$ and, as $i \neq j$, $z'_{2i} = z'_{2i-1}$, we obtain that $\tilde{f}_{2i-1}(z) = f_i(x)$ and $\tilde{f}_{2i-1}(z') = f_i(y)$. □

Lemma 4. *For any $i \in [n]$,*

a. there is a positive self-loop on $2i-1$ in $G(\tilde{f})$ if and only if there exists $x \in \mathbb{B}^n$ such that $f_i(x) = x_i$;
b. there is never a negative self-loop on $2i-1$ in $G(\tilde{f})$;
c. there is never a positive edge from $2i$ to $2i-1$ in $G(\tilde{f})$;
d. there is a negative edge from $2i$ to $2i-1$ in $G(\tilde{f})$ if and only if there exists $x \in \mathbb{B}^n$ such that $f_i(x) \neq x_i$
e. there is always exactly one edge from $2i-1$ to $2i$ in $G(\tilde{f})$ and it is positive.

Proof. (a) Let us consider $z, z' \in \mathbb{B}^{2n}$ such that $\Delta(z, z') = \{2i-1\}$ with $z_{2i-1} = 0$: $\tilde{f}_{2i-1}(z) = 0 = \neg\tilde{f}_{2i-1}(z') \Leftrightarrow [(z_{2i} = 0 \wedge f_i(\gamma(z)) = 0) \vee (z_{2i} = 1 \wedge f_i(\gamma(z)) = 1)] \Leftrightarrow f_i(\gamma(z)) = z_{2i}$. (b) Let us consider $z, z' \in \mathbb{B}^{2n}$ such that $\Delta(z, z') = \{2i-1\}$ with $z_{2i-1} = 0$ and $\tilde{f}_{2i-1}(z) = 1 = \neg\tilde{f}_{2i-1}(z')$. Thus, $z_{2i} = 0$, therefore, $\tilde{f}_{2i-1}(z') = z'_{2i-1} = 1$, which is a contradiction. (c) Let us consider $z, z' \in \mathbb{B}^{2n}$ such that $\Delta(z, z') = \{2i\}$ with $z_{2i} = 0$: if $z_{2i-1} = z'_{2i-1} = 0$, then $\tilde{f}_{2i-1}(z) \geq \tilde{f}_{2i-1}(z')$; if $z_{2i-1} = z'_{2i-1} = 1$, then $\tilde{f}_{2i-1}(z) \geq \tilde{f}_{2i-1}(z')$; therefore there cannot

be a negative edge from $2i$ to $2i-1$ in $G(\tilde{f})$. (d) $\exists z, z' \in \mathbb{B}^{2n}$: $\Delta(z, z') = \{2i\}$, $z_{2i} = 0$, $\tilde{f}_{2i-1}(z) = 1 = \neg\tilde{f}_{2i-1}(z') \Leftrightarrow [(z_{2i-1} = z'_{2i-1} = 0 \wedge f_i(\gamma(z)) = 1) \vee (z_{2i-1} = z'_{2i-1} = 1 \wedge f_i(\gamma(z')) = 0)] \Leftrightarrow \exists x \in \mathbb{B}^n : f_i(x) = \neg x_i$. (e) By $\tilde{f}_{2i}$ definition.

From Lemma 4, one can deduce that if there is a positive self-loop on i in $G(f)$, then there is a positive self-loop on $2i-1$ in $G(\tilde{f})$; and if there is a negative self-loop on i in $G(f)$, then there is a negative edge from $2i$ to $2i-1$ in $G(\tilde{f})$.

We can then deduce that the positive and negative cycles of $G(f)$ are preserved in $G(\tilde{f})$. It is worth noting that the encoding may also introduce negative cycles between $2i-1$ and $2i$ and positive self-loops on $2i-1$, for some $i \in [n]$.

Lemma 5. *To each positive (resp. negative) cycle in $G(f)$ of length $k > 1$, there exists a corresponding positive (resp. negative) cycle in $G(\tilde{f})$ of length $2k$. To each positive self-loop in $G(f)$ corresponds one positive self-loop in $G(\tilde{f})$; to each negative self-loop in $G(f)$ corresponds a negative cycle in $G(\tilde{f})$ of length 2.*

Proof. For cycle of length $k > 1$, by Lemma 3 and by the fact that there is a positive edge from $2i-1$ to $2i$ in $G(\tilde{f})$: each edge (i, j) in the cycle in $G(f)$ is mapped to the string $(2i, 2j-1)(2j-1, 2j)$, giving a cycle in $G(\tilde{f})$ of the same sign. Correspondence of self-loops is given by Lemma 4 □

5 Further Extensions

Our interval semantics decouples the update of a node in order to allow the interleaving of transitions during the interval when the next value has been computed (write node) but not applied yet (read node still with the before-update value). This also implies that, during this interval, the other nodes have access only to the before-update value. A third feature of the interval semantics is the enforcement of the update application: once an update is triggered (write node gets a different value than the read node), no further update on the same node is possible until the update has been applied. Thus, if for instance the update triggers a change of value from 0 to 1, the interval semantics guarantees that the read node will eventually have the value 1.

These two aspects, restricted access to the before-update value of nodes and enforcement of update application, were essentially motivated by our choice that our interval semantics should simulate the synchronous update of nodes used in the classical synchronous and generalized asynchronous semantics, as stated in Theorem 1. However, one could go further and consider extended interval semantics which relax either the restricted access to the before-update value of nodes, or the enforcement of update application, or both. We will see that these relaxations of our interval semantics still preserve the consistency properties stated in Sect. 4.3.

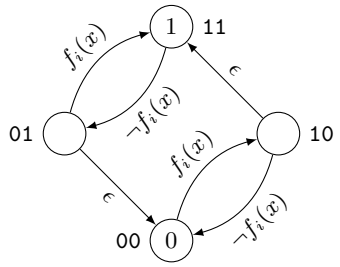

Fig. 3. Automaton of the value change of a node i in the extended interval semantics where the update can be canceled if $f_i(x)$ changes of value during the interval of update. Notations follow the ones of Fig. 2

5.1 Update cancellation

The relaxation of the enforcement of update application can be interpreted as the ability to cancel an ongoing update when f_i changes of value during the interval of update. This can be described by the automaton of Fig. 3, and encoded by removing $\neg z_{2i}$ and z_{2i-1} from the definition of $\tilde{f}_{2i-1}$ in Def. 6.

Theorem 1 and the lemmas in previous section are still verified with update cancellation. Moreover, this extension does not introduce any additional self-loop on $2i-1$ or negative edge from $2i$ to $2i-1$ in the influence graph.

5.2 Reading from either the before-update or after-update values

In terms of modeling, the restriction to before-update values can be seen as an asymmetry in the consideration of transitions: the resource modified by the transition is still available during the interval of update, whereas the result is only available once the transition finished. When modelling biological systems, it translates into considering only species which are slow to reach their activity threshold.

Actually, the choice of whether the before-update, after-update or both values are available during the update may be done according to the knowledge of the modeled system. Our construction can easily be adapted for giving access, depending on the node, to the after-update value instead of the before-update value. For instance, if the node i should follow closely value changes of node i, then node j should access the after-update value (write node) of i, whereas, as in our motivating example, if i is slow to update compared to j, node j should access the before-update value (read node) of i.

Finally, one could also consider a more permissive symmetric version which would allow the access of both before-update and after-update values. This choice may be very reasonable when not much is known about the system, for instance about the relative speed of the nodes.

5.3 Comparison with multi-valued networks

Multi-valued networks [7] are an extension of Boolean networks where the domain of each node $i \in [n]$ ranges over a finite discrete ordered domain $\mathbb{D}_i$. The value changes of the nodes are specified using a function $g_i : \mathbb{D}_1 \times \cdots \times \mathbb{D}_n \to \{-, 0, +\}$ which determines the direction of the value change.

Thus, a strong constraint of this semantics is that value changes are always unitary: a transition will either change the value to the smallest higher one, or the highest smaller one, if it exists. However, one can remark that the automaton modeling the value change with the interval semantics (Fig. 2) does not satisfy such a constraint, and hence cannot be encoded as a single multi-valued node.

6 Discussion

As shown in our motivating example in Sect. 3, the interval semantics can enable the reachability of configurations that are not allowed in other updating modes, notably asynchronous or generalized asynchronous. This can be problematic when expecting Boolean networks to produce an over-approximation of reachable configurations due to the abstraction of parameters related to speed and activity threshold of components, as it is usually assumed when modelling biological networks. It appears that the Boolean network in Sect. 3 is embedded in numerous actual models of biological networks (e.g., [16,17,26]). Therefore, the result of analysis of the transient dynamics of these models may be deeply impacted by using the interval semantics, which has never been considered so far.

The transitions enabled by the interval semantics are due to nodes which update slowly: whenever committed to a value change, in the meantime of the update application, the other nodes of the network still evolve subject to its before-update value. This time scale consideration brings an interesting feature when modeling biological networks which gathers processes of different nature and velocity. Our encoding allows the application of the interval semantics only to a subset of nodes, offering a flexible modelling approach.

Future work consider determining semantics of Boolean networks which guarantee the formal simulation of hybrid and continuous network dynamics.

References

1. J. Aracena. Maximum number of fixed points in regulatory boolean networks. *Bulletin of Mathematical Biology*, 70(5):1398–1409, 2008.
2. J. Aracena, J. Demongeot, and E. Goles. Positive and negative circuits in discrete neural networks. *IEEE Transactions of Neural Networks*, 15:77–83, 2004.
3. J. Aracena, E. Goles, A. Moreira, and L. Salinas. On the robustness of update schedules in Boolean networks. *Biosystems*, 97(1):1 – 8, 2009.
4. J. Aracena, A. Richard, and L. Salinas. Number of fixed points and disjoint cycles in monotone boolean networks. *SIAM Journal on Discrete Mathematics*, 31(3):1702–1725, 2017.

5. J. Baetens, P. V. der Weeën, and B. D. Baets. Effect of asynchronous updating on the stability of cellular automata. *Chaos, Solitons & Fractals*, 45(4):383 – 394, 2012.
6. P. Baldan, A. Corradini, and U. Montanari. Contextual Petri nets, asymmetric event structures, and processes. *Information and Computation*, 171(1):1–49, 2001.
7. G. Bernot, F. Cassez, J.-P. Comet, F. Delaplace, C. Müller, and O. Roux. Semantics of biological regulatory networks. *Electronic Notes in Theoretical Computer Science*, 180(3):3 – 14, 2007.
8. N. Busi and G. M. Pinna. Non sequential semantics for contextual P/T nets. In *Application and Theory of Petri Nets*, volume 1091 of *Lecture Notes in Computer Science*, pages 113–132. Springer, 1996.
9. C. Chaouiya, A. Naldi, E. Remy, and D. Thieffry. Petri net representation of multi-valued logical regulatory graphs. *Natural Computing*, 10(2):727–750, 2011.
10. T. Chatain, S. Haar, L. Jezequel, L. Paulevé, and S. Schwoon. Characterization of reachable attractors using Petri net unfoldings. In *Computational Methods in Systems Biology*, volume 8859 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2014.
11. T. Chatain, S. Haar, M. Koutny, and S. Schwoon. Non-atomic transition firing in contextual nets. In *Applications and Theory of Petri Nets*, volume 9115 of *Lecture Notes in Computer Science*, pages 117–136. Springer, 2015.
12. A. Garg, A. Di Cara, I. Xenarios, L. Mendoza, and G. De Micheli. Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinformatics*, 24(17):1917–1925, 2008.
13. R. Janicki and M. Koutny. Semantics of inhibitor nets. *Information and Computation*, 123(1):1–16, 1995.
14. R. Janicki and M. Koutny. Fundamentals of modelling concurrency using discrete relational structures. *Acta Inf.*, 34:367–388, 1997.
15. S. A. Kauffman. Metabolic stability and epigenesis in randomly connected nets. *Journal of Theoretical Biology*, 22:437–467, 1969.
16. Z. Mai and H. Liu. Boolean network-based analysis of the apoptosis network: Irreversible apoptosis and stable surviving. *Journal of Theoretical Biology*, 259(4):760 – 769, 2009.
17. P. Martínez-Sosa and L. Mendoza. The regulatory network that controls the differentiation of t lymphocytes. *Biosystems*, 113(2):96 – 103, 2013.
18. M. Noual and S. Sené. Synchronism versus asynchronism in monotonic boolean automata networks. *Natural Computing*, 2017.
19. E. Palma, L. Salinas, and J. Aracena. Enumeration and extension of non-equivalent deterministic update schedules in boolean networks. *Bioinformatics*, 32(5):722–729, 2016.
20. E. Remy, P. Ruet, and D. Thieffry. Graphic requirements for multistability and attractive cycles in a Boolean dynamical framework. *Advances in Applied Mathematics*, 41(3):335 – 350, 2008.
21. A. Richard. Negative circuits and sustained oscillations in asynchronous automata networks. *Advances in Applied Mathematics*, 44(4):378 – 392, 2010.
22. B. Schönfisch and A. de Roos. Synchronous and asynchronous updating in cellular automata. *Biosystems*, 51(3):123 – 143, 1999.
23. L. J. Steggles, R. Banks, O. Shaw, and A. Wipat. Qualitatively modelling and analysing genetic regulatory networks: a petri net approach. *Bioinformatics*, 23(3):336–343, 2007.

24. D. Thieffry and R. Thomas. Dynamical behaviour of biological regulatory networks – II. Immunity control in bacteriophage lambda. *Bulletin of Mathematical Biology*, 57:277–297, 1995.
25. R. Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42(3):563 – 585, 1973.
26. P. Traynard, A. Fauré, F. Fages, and D. Thieffry. Logical model specification aided by model-checking techniques: application to the mammalian cell cycle regulation. *Bioinformatics*, 32(17):i772–i780, 2016.
27. W. Vogler. Partial order semantics and read arcs. *Theoretical Computer Science*, 286(1):33–63, 2002.
28. J. Winkowski. Processes of contextual nets and their characteristics. *Fundamenta Informaticae*, 36(1), 1998.