

Influence of update schemes on boolean networks

Tom Zuidberg
455969



Abstract—Boolean networks are a widely used framework for modeling the qualitative dynamics of biological regulatory systems. A central modeling decision is the choice of update scheme, which determines how and when individual nodes are evaluated at each time step and directly shapes the resulting dynamics. This paper introduces and compares four commonly used update schemes: synchronous, sequential, probabilistic, and asynchronous. We analyze how each scheme affects attractor structure, reachability, and network robustness, and discuss their practical implications for the modeling of gene regulatory networks. A central finding is that the same network can exhibit fundamentally different attractors depending on which scheme is used, demonstrating that the choice of update scheme is not a neutral implementation detail but directly shapes the dynamical behavior of the model and the biological conclusions that can be drawn from it.

1 INTRODUCTION

Boolean networks were first introduced by Stuart Kauffman in the late 1960s as a simplified model of biochemical systems [1]. In this framework, genes or regulatory components are represented as binary variables, each taking a value of either 0 (inactive) or 1 (active), whose states evolve according to logical update functions encoding regulatory interactions. Despite their simplicity, boolean networks have proven useful in capturing qualitative dynamical behavior in complex biochemical systems and have since become a standard tool of modeling in systems biology [2].

A key abstraction in boolean network modeling is the concept of attractors. Since a network of n binary nodes has exactly 2^n possible states and the update functions are deterministic, the system must eventually revisit a state it has already visited. The resulting recurring patterns, either fixed points or limit cycles, are called attractors and are often interpreted as stable cellular or functional states [1]. In the context of gene regulatory networks (GRNs), attractors have been proposed to correspond to distinct cell types or phenotypic states [2].

Gene regulatory networks describe the interactions between genes, transcription factors and other molecular entities that together control gene expression. Modeling GRNs as boolean networks provides a simplified view: instead of tracking continuous concentration levels of molecular species, each gene is treated as either expressed or silenced. This allows for qualitative insight into questions such as how different cell types arise from the same genetic

blueprint, or how mutations affect the stable states of the system [2], [3].

When it comes to modeling a system as a boolean network, the choice of update scheme plays an important role, as it defines how and when individual nodes are updated at each time step. This choice is not merely a technical detail, since it affects which attractors exist, which states are reachable from a given initial condition, and how robustly the network responds to perturbations [3], [4]. The most commonly used schemes include the synchronous scheme, where all nodes update simultaneously; the sequential scheme, where nodes are updated one after another in a fixed order; the probabilistic scheme, which introduces randomness into the update process; and the asynchronous scheme, where only one node is updated per time step.

The question of which update scheme best reflects biological reality has been an active area of research. Asynchronous schemes have often been considered more biologically plausible, since biochemical processes in living cells rarely occur simultaneously across all genes. However, as recent studies have shown, synchronous and asynchronous schemes can lead to the same meaningful behavior under the right conditions, and the practical choice of scheme often comes down to available biological knowledge and computational feasibility [3].

In the following, we will introduce the formal model of boolean networks in section 2, followed by a comparison of the different update schemes in section 3. Finally, in section 4, we will discuss the implications of these schemes for the modeling of gene regulatory networks.

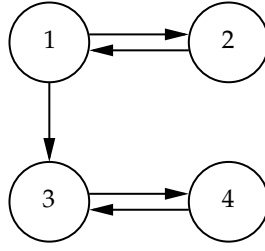
2 BOOLEAN NETWORKS

A boolean network consists of nodes $x_i(t)$ that have a boolean state, either 0 or 1, at a point in time t . Each node has a corresponding update function

$$x_i(t+1) = f_i(x_1(t), x_2(t), \dots, x_n(t))$$

expressing the new state of $x_i(t+1)$. The state of the boolean network can be described as a binary string $x(t) = x_1x_2 \dots x_n$ where each node is replaced with the corresponding state 0 or 1.

Consider the boolean network shown in fig. 1. For example, if the state is $x(t) = 0011$ at time t , meaning that x_1, x_2 are 0 and x_3, x_4 are 1. After applying the synchronous



$$\begin{aligned} f_1 &= \neg x_2 \\ f_2 &= x_1 \\ f_3 &= x_1 \oplus x_4 \\ f_4 &= x_3 \end{aligned}$$

Fig. 1. Example of a boolean network with four nodes. Each vertex indicates that a node is part of the update function of the other node. For instance x_1 is part of f_2 and f_3 . Note that $\neg$ means the logical NOT and $\oplus$ means the logical XOR function.

update, the next state $x(t+1)$ is 1011. Updating the network multiple times creates a trajectory, which is the sequence of the states starting from the initial state. With our example, the trajectory is $0011 \rightarrow 1011 \rightarrow 1101 \rightarrow 0100 \rightarrow \dots$. Each trajectory in a reasonably small boolean network eventually reaches one of two scenarios. Either it falls into a state, called an attractor, that does not change when updated, or it falls into a cycle of states.

A state graph shows all possible states of the network with the corresponding trajectory. For example, the state graph in fig. 3 shows the trajectories from the boolean network in fig. 1. These state graphs help visualize the attractors and cycles, e.g. 2 cycles of length 8.

In boolean networks, robustness refers to the stability of the behavior with regard to perturbations, e.g. a single node state change. A boolean network with poor robustness means that a simple flip of a node can change the resulting attractor or cycle for that trajectory. With regard to our example from fig. 3, if either x_3 or x_4 (not both at once) is flipped, the cycle switches. A strong robustness however would mean that even with such perturbations, the resulting attractor or cycle does not change.

The set of all states that eventually lead to a particular attractor is called its *basin of attraction*. In the state graph in fig. 3, the two cycles each have a basin of eight states, meaning that starting from any state in a basin, the network will always converge to the corresponding attractor. Larger basins generally indicate more dominant or robust attractors. In biological terms, a large basin corresponds to a cell state that is easily reached and hard to escape from, which is a desirable property for modeling stable functional states such as cell types.

Not every state in a boolean network is reachable from another state through the dynamics. States that have no predecessor in the state graph are called *garden-of-Eden states*: they can only appear as initial conditions and are never produced by any update step. Their presence reduces the effective state space that the network can explore during normal operation. In the synchronous state graph of fig. 3,

$x(t)$	$x(t+1)$
0000	1000
0001	1010
0010	1001
0011	1011
0100	0000
0101	0010
0110	0001
0111	0011
1000	1110
1001	1100
1010	1111
1011	1101
1100	0110
1101	0100
1110	0111
1111	0101

Fig. 2. State transitions for all states of the boolean network from fig. 1 using synchronous update scheme once.

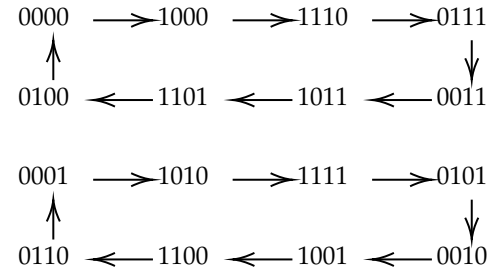


Fig. 3. State graph of the boolean network shown in fig. 1 using synchronous update scheme.

no garden-of-Eden states are present, since every state is reachable from at least one other state within the two cycles.

The structure of the state graph, including the number and size of attractors, the shape of their basins and the presence of garden-of-Eden states, depends critically on both the network topology and the choice of update scheme. The following section shows that changing the update scheme while keeping the network and its update functions fixed can produce a completely different state graph, with different attractors, different basins and different reachability properties.

3 UPDATE SCHEMES

The dynamics of a boolean network depend not only on its topology and update functions, but also on the choice of update scheme, which defines how and when nodes are updated at each time step. Different schemes can lead to notably different behavior, including different attractor structures, reachability properties and robustness characteristics [3], [4]. In general, update schemes can be divided into deterministic schemes, where the sequence of updates is fixed, and stochastic schemes, where randomness plays a role. In the following subsections, we introduce four commonly used update schemes and illustrate their effects on the boolean network from fig. 1.

3.1 Synchronous scheme

In section 2, we have used the synchronous update scheme to introduce the general dynamics of boolean networks. This scheme assumes that each update function takes the exact same amount of time and executes at the same time, which means that regardless of the outcome of the first function, any subsequent function will have the same result. Updating the network with this method is quite simple, always leads to a deterministic output and does not require a lot of computing power in order to simulate larger networks, as each state of the network results in exactly one updated state. With regard to gene regulatory networks however, this is an oversimplification as normal GRNs never execute fully synchronously. Furthermore, the change of one node has no effect on any other function, whereas larger biochemical systems often show dependencies between functions.

Despite this biological limitation, the synchronous scheme has important structural properties that make it the most widely used scheme in practice. Since every global state maps to exactly one successor state, the state space forms a functional graph in which every node has exactly one outgoing edge. This means that every trajectory will eventually converge to either a fixed-point attractor or a periodic cycle, and the full attractor landscape is completely determined by the update functions alone [1]. As a direct consequence, all attractors can be found by exhaustively simulating every possible initial state, which requires at most 2^n simulations of bounded length. For networks of moderate size this is computationally feasible, making synchronous update well suited for systematic attractor analysis.

The synchronous scheme was also the original formulation used by Kauffman when he introduced random boolean networks as a model of gene regulation [1]. In his analysis of networks where update functions are assigned randomly, Kauffman found that the average number and length of attractors depends strongly on the average node connectivity K . At low connectivity the network settles quickly into a small number of short attractors; at high connectivity the dynamics become chaotic with very long cycle lengths. At a critical connectivity of around $K = 2$ the network is poised between these two regimes, a property that has been proposed as a signature of real biological regulatory networks [2]. This critical behavior and the tools used to study it are defined with respect to the synchronous scheme, which illustrates why synchronous update remains central to the theoretical analysis of boolean networks even when other schemes may be more biologically appropriate.

3.2 Sequential scheme

In the sequential update scheme, nodes are updated one after another in a fixed order within each time step, which allows for side effects: the updated value of an earlier node in the sequence can influence the outcome of later ones. For example, take the sequence (x_1, x_2, x_3, x_4) . Updating the state 0000 means first using f_1 which results in an intermediate state 1000, then using f_2 , then f_3 , and lastly f_4 . This will lead to $0000 \rightarrow 1111$, a different state from that produced when using the synchronous update scheme. In

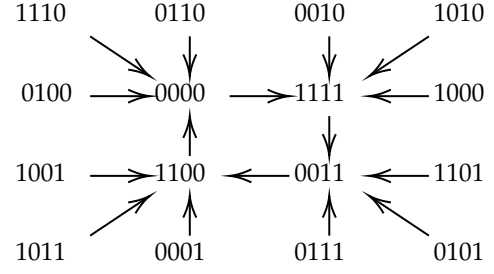


Fig. 4. State graph of the boolean network shown in fig. 1 using the sequential update with the sequence (x_1, x_2, x_3, x_4)

fact, the whole state graph changes depending on the update scheme used, namely every initial state will fall into a single 4-state cycle: $0000 \rightarrow 1111 \rightarrow 0011 \rightarrow 1100 \rightarrow 0000$, see fig. 4.

While the sequential update scheme produces different dynamics from the synchronous scheme, it is always possible to derive an equivalent synchronous function set that produces identical state transitions. The key observation is that when a node x_i is evaluated in a sequential update, all nodes x_j that appear earlier in the sequence have already been updated to their new values. The synchronous equivalent f'_i is therefore obtained from f_i by substituting, for each input variable x_j that precedes x_i in the sequence, the already-derived new function f'_j in place of x_j .

For the boolean network from fig. 1 with sequence (x_1, x_2, x_3, x_4) , the conversion proceeds as follows. Since x_1 is first in the sequence, its function is unchanged. For x_2 , the original function $f_2 = x_1$ depends on x_1 , which has already been updated; substituting $x_1 \mapsto f'_1$ gives $f'_2 = f'_1$. For x_3 , the function $f_3 = x_1 \oplus x_4$ depends on x_1 (already updated) but not x_2 ; substituting $x_1 \mapsto f'_1$ gives $f'_3 = f'_1 \oplus x_4$. Finally, $f_4 = x_3$ depends on x_3 , which has already been updated; substituting $x_3 \mapsto f'_3$ gives $f'_4 = f'_3$. Expanding these yields the synchronous functions

$$\begin{aligned}
 f'_1 &= \neg x_2, \\
 f'_2 &= f'_1 = \neg x_2, \\
 f'_3 &= f'_1 \oplus x_4 = \neg x_2 \oplus x_4, \\
 f'_4 &= f'_3 = \neg x_2 \oplus x_4.
 \end{aligned}$$

This means that a sequential scheme can always be expressed as an equivalent synchronous scheme with a different set of update functions. Furthermore, if the side-effects from the sequential scheme are desired, the synchronous function set would become larger and more complex.

An extension of the sequential scheme is the block-sequential scheme. This allows the modeler to divide the nodes into groups, called blocks, that run synchronously, while the blocks execute in sequence. This allows for further fine-tuning of the behavior and can simplify the modeling of realistic systems. The block-sequential scheme can also be converted to a synchronous scheme, thus producing the same behavior as previously described [5].

An important practical consideration is that for a network with n nodes, there are $n!$ possible sequential orderings, and different orderings can produce different state

graphs and attractors. This makes the choice of sequence both a useful modeling tool and a potential source of ambiguity, since the biologically correct ordering is often not known in advance [5]. The block-sequential scheme partially addresses this by only requiring a partition of nodes into blocks and an ordering of those blocks, rather than a complete linear ordering over all nodes.

3.3 Probabilistic scheme

The probabilistic scheme is an extension of the synchronous, sequential and non-random asynchronous schemes as it introduces randomness into the system. Instead of having one function per node, each node has multiple update functions with individual probabilities. Upon updating a node, one of these functions is chosen at random according to the probability distribution. This allows the introduction of noise and uncertainty into the system.

An example of how this can drastically change the outcome can be visualized when adding a second function to the node x_4 from the example in fig. 1. Let $f_{4,\text{switch}} = \neg f_4 = \neg x_3$ with an equal probability distribution. This simple addition means that on average, every second update the state switches from one cycle to the other. Thus there are no longer two individual cycles. Any initial state will eventually reach all possible states and will never fall into a repeating pattern.

More formally, a probabilistic boolean network assigns to each node x_i a set of update functions $\{f_i^{(1)}, \dots, f_i^{(k_i)}\}$ with corresponding probabilities $\{p_i^{(1)}, \dots, p_i^{(k_i)}\}$ that sum to 1. At each step, one function per node is selected at random according to these probabilities. The resulting dynamics can be analyzed as a Markov chain over the 2^n possible states, where the long-term behavior is described by a stationary distribution over states rather than a single fixed attractor [3].

This connection to GRN modeling is particularly relevant for oscillatory networks such as the Repressilator, discussed in detail in section 4. Under deterministic update schemes, the Repressilator produces a clean periodic oscillation. In real biological cells, however, gene expression is inherently noisy: a gene that should be repressed may occasionally still produce a small amount of protein due to stochastic effects such as transcriptional bursting or low molecule counts. By introducing a small probability for an alternative update function, this noise can be captured in the model. The result is an irregular oscillation whose period varies from cycle to cycle, which is considerably closer to the gene expression dynamics observed in living cells than the perfectly regular output of a deterministic scheme [3].

3.4 Asynchronous scheme

While synchronous, sequential and probabilistic schemes update all the nodes in one time step, the asynchronous scheme only updates one node per time step. There are two main variants. In asynchronous deterministic update, a fixed predetermined sequence specifies which single node is updated at each time step. This differs from the sequential scheme in a fundamental way: in the sequential scheme, the entire sequence is executed within one time

step, updating all nodes once; in asynchronous deterministic update, each element of the sequence constitutes a separate time step that updates exactly one node. The same node can therefore appear at multiple positions across successive time steps, giving it a higher effective update frequency and allowing the modeler to encode that certain regulatory processes occur faster than others. In asynchronous random update, the node to be updated at each step is chosen uniformly at random. While the probabilistic scheme introduced stochasticity at the level of function selection, asynchronous random update introduces stochasticity at the level of node selection: there is no predetermined order, and from any given state, multiple different transitions are possible depending on which node happens to be selected.

During early stages of boolean network research, it was believed that asynchronous update would lead to more realistic models of biochemical systems [3]. However, as studies have shown, asynchronous and synchronous schemes can lead to the same meaningful behavior for certain networks, and the choice between them often comes down to practical considerations. A concrete limitation of the asynchronous approach is its computational cost: since each time step updates only one node, a single full round over all n nodes requires n sequential steps. Assuming each step takes seconds to minutes, this adds up to days for large networks [3].

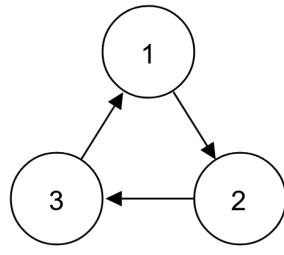
A key structural consequence of the random asynchronous variant specifically is that it introduces non-determinism into the state graph. Since the node to update is chosen at random, from any given state there are in general multiple possible successor states rather than a single deterministic one. This makes identifying attractors more complex than in the synchronous or sequential case, since a state can lead to different trajectories on different runs [3].

This difference has a concrete consequence in GRN models. In the Repressilator (see section 4 and fig. 6), the synchronous scheme produces two attractors: a biologically meaningful 6-cycle and an artificial 2-cycle that alternates between the all-off state 000 and the all-on state 111. Under random asynchronous update, neither 000 nor 111 are attractors. From state 000, for example, each update function evaluates to $\neg 0 = 1$, meaning any node selected for updating immediately moves the network into one of the oscillatory states. The artificial 2-cycle therefore disappears, and only the biologically relevant oscillation remains. This illustrates a broader finding: many cyclic attractors in synchronous boolean networks are artifacts of simultaneous updating that vanish under asynchronous dynamics [2].

Furthermore, the random asynchronous scheme is closely related to the probabilistic scheme. Rather than randomizing which update function is applied to each node, it randomizes which node is updated at each step, while keeping the update functions themselves fixed.

4 RELEVANCE FOR GENE REGULATORY NETWORKS

A gene regulatory network (GRN) is a system of interactions between genes, transcription factors and other regulatory molecules that together control gene expression in a cell.



$$\begin{aligned} f_1 &= \neg x_3 \\ f_2 &= \neg x_1 \\ f_3 &= \neg x_2 \end{aligned}$$

$x(t)$	$x(t+1)$
000	111
001	011
010	110
011	010
100	101
101	001
110	100
111	000

Fig. 5. The GRN Repressilator modeled as a boolean network and the update table using the synchronous scheme.

Since each gene can be treated as either active (expressed) or inactive (silenced), boolean networks are a natural fit for modeling GRNs. They have been applied to a variety of well-known GRNs, such as the cell-cycle network of *Saccharomyces cerevisiae* or the segment polarity network of *Drosophila melanogaster* [1], [2]. In these models, the attractors of the boolean network are interpreted as stable gene expression patterns corresponding to distinct cell types or functional states. A cell that has committed to a specific developmental fate, for example, would correspond to a fixed-point attractor in the GRN model, while transient oscillations in gene expression would appear as limit cycles.

A particularly instructive example for studying the effect of update schemes on GRN models is the *Repressilator*: a synthetic regulatory circuit in which three genes each repress the next in a cyclic fashion [2]. As shown in fig. 5, the network has update functions of the form $f_i = \neg x_{i-1}$ (indices modulo 3). Despite having only three nodes and eight possible states, the Repressilator shows a strikingly different attractor structure depending on which update scheme is used.

Under the **synchronous** scheme, the network has two attractors, shown in the upper part of fig. 6. The first is a period-6 oscillation cycle through the states $100 \rightarrow 101 \rightarrow 001 \rightarrow 011 \rightarrow 010 \rightarrow 110 \rightarrow 100$, which corresponds to the alternating gene expression observed in experimental implementations of the Repressilator. The second is an artificial period-2 cycle alternating between 000 (all genes off) and 111 (all genes on). This second attractor has no biological basis: it arises purely because updating all three repressors simultaneously maps 000 to 111 and back, a dynamic that would not occur if the genes were updated at slightly different times.

Under **random asynchronous** update, the picture changes. From state 000, any node selected for updating will have $f_i = \neg 0 = 1$, meaning the network is immediately pushed into one of the oscillatory states regardless of which node is chosen. The same applies to 111. Neither state can sustain itself, so they become transient states and the artificial 2-cycle disappears entirely. Only the biologically meaningful oscillation remains as an attractor. This confirms

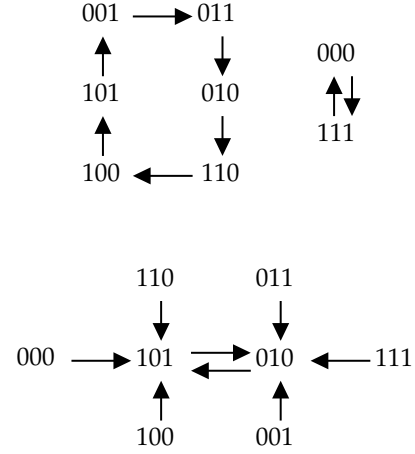


Fig. 6. State graphs of the repressilator as boolean network: The upper two cycles are the result of the use of synchronous scheme; the lower graph is the result of sequential scheme with sequence (x_1, x_2, x_3) .

that the period-2 cycle in the synchronous case was indeed an artifact of simultaneous updating [2], and it illustrates why asynchronous update is often considered more biologically faithful for oscillatory GRN models.

Under a **sequential** scheme with sequence (x_1, x_2, x_3) , the attractor structure changes even more dramatically, as shown in the lower part of fig. 6. Tracing through the state transitions reveals that the only attractor is now a period-2 cycle between states 010 and 101, a subset of the states from the synchronous 6-cycle. The synchronous 6-cycle itself no longer exists as an attractor, and all eight states converge to this single 2-cycle. This is a clear example that even among deterministic schemes, changing the update order can completely alter the attractor landscape. Under a **probabilistic** scheme, the clean periodic oscillation gives way to an irregular, noisy one. Introducing a small probability for an alternative update function on any of the three genes, modeling the stochastic fluctuations that occur in real gene expression, means the network occasionally deviates from the regular cycle. The long-term behavior is no longer a fixed attractor but a stationary distribution over states, reflecting how much time the system spends in each expression configuration on average [3]. This type of model is particularly useful for understanding phenotypic variability within a population of genetically identical cells.

The Repressilator example illustrates a key point for GRN modeling in general: the choice of update scheme is not a neutral decision. It directly affects which attractors exist, which states are biologically accessible, and how realistically the model captures the noise and timing characteristics of the underlying system. For large-scale GRNs where the biological timing of regulatory events is not known, synchronous update remains the practical standard due to its computational simplicity and the ease of attractor analysis. Sequential or block-sequential update is appropriate when relative timing information is available. Asynchronous update is biologically motivated but becomes slow for large networks, and probabilistic update is the method of choice when transcriptional noise is a central modeling goal.

5 CONCLUSION

This paper gave an overview of four update schemes for boolean networks and analyzed how they affect the dynamics of a model. The synchronous scheme, where all nodes update simultaneously, produces fully deterministic and easily enumerable dynamics, making it the most widely used approach in practice despite its biological simplifications. Sequential and block-sequential schemes allow for a more realistic encoding of temporal ordering between regulatory events while retaining determinism. The probabilistic scheme introduces stochastic variation at the level of the update functions, capturing noise and fluctuations in the system. The asynchronous scheme updates one node at a time and, in its random variant, introduces non-determinism at the level of node selection rather than function selection.

The Repressilator example from section 4 makes the practical stakes of this choice concrete. Even in a network with only three nodes, as illustrated in fig. 6, the synchronous scheme produces a spurious 2-cycle between 000 and 111 alongside the biologically relevant 6-cycle. The asynchronous scheme eliminates this artifact. The sequential scheme with sequence (x_1, x_2, x_3) replaces the 6-cycle with an entirely different 2-cycle through $010 \leftrightarrow 101$. The probabilistic scheme turns the regular oscillation into a noisy, variable one. The update scheme is therefore not just a computational choice, it determines which attractors the model has and what biological conclusions can be drawn from them.

The question of which scheme best captures the biology of a given system remains an active area of research. A notable recent contribution is the introduction of Most Permissive Boolean Networks [6], a new update semantics that generalizes both synchronous and asynchronous approaches and carries a formal guarantee that no behavior achievable by a compatible quantitative model is missed. Unlike standard asynchronous update, the most permissive semantics also avoids the state-space explosion that makes large-scale asynchronous analysis intractable, enabling attractor analysis of genome-scale regulatory networks. On the tooling side, software packages such as BoolNet [7] and GINsim [8] bring multiple update schemes and attractor analysis algorithms to a broader research community.

Open questions include how to choose a biologically grounded update scheme when experimental timing data is sparse, how to parameterize probabilistic models from single-cell RNA sequencing data, and how to efficiently compute attractor landscapes in asynchronous networks with thousands of nodes. Boolean networks, despite their simplicity, remain a powerful framework for understanding the dynamical logic of gene regulation, and the update scheme is central to using them well.

REFERENCES

- [1] Tomas Helikar, Naomi Kochi, John Konvalina, and Jim A Rogers. Boolean modeling of biochemical networks. *The Open Bioinformatics Journal*, 5(1), 2011.
- [2] Stefan Bornholdt. Boolean network models of cellular regulation: prospects and limitations. *Journal of the Royal Society interface*, 5(suppl_1):S85–S94, 2008.
- [3] Julian D Schwab, Silke D Kühlwein, Nensi Ikonomi, Michael Kühl, and Hans A Kestler. Concepts in boolean network modeling: What do they all mean? *Computational and structural biotechnology journal*, 18:571–582, 2020.
- [4] Julio Aracena, Eric Goles, Andrés Moreira, and Luis Salinas. On the robustness of update schedules in boolean networks. *Biosystems*, 97(1):1–8, 2009.
- [5] Eric Goles and Mathilde Noual. Block-sequential update schedules and boolean automata circuits. *Discrete Mathematics & Theoretical Computer Science*, (Proceedings), 2010.
- [6] Loïc Paulevé, Juri Kolčák, Thomas Chatain, and Stefan Haar. Reconciling qualitative, abstract, and scalable modeling of biological networks. *Nature communications*, 11(1):4256, 2020.
- [7] Christoph Müssel, Martin Hopfensitz, and Hans A Kestler. Boolnet—an r package for generation, reconstruction and analysis of boolean networks. *Bioinformatics*, 26(10):1378–1380, 2010.
- [8] Aurélien Naldi, Céline Hernandez, Nicolas Levy, Gautier Stoll, Pedro T Monteiro, Claudine Chaouiya, Tomáš Helikar, Andrei Zinovyev, Laurence Calzone, Sarah Cohen-Boulakia, et al. The colomoto interactive notebook: accessible and reproducible computational analyses for qualitative biological networks. *Frontiers in physiology*, 9:680, 2018.