

# Computing Attractors of Large-Scale Asynchronous Boolean Networks Using Minimal Trap Spaces

Van-Giang Trinh<sup>1</sup>, Kunihiro Hiraishi<sup>2</sup> and Belaid Benhamou<sup>1</sup>

<sup>1</sup>LIS, Aix-Marseille Université, Marseille, France

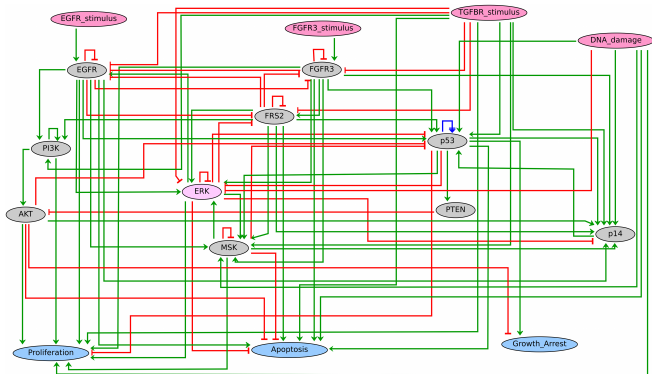
<sup>2</sup>School of Information Science, Japan Advanced Institute of Science and Technology, Japan

August 07, 2022



## Boolean modeling

Boolean network modeling of **gene regulation** but also of other biological systems has had great successes over the last ~20 years.



**Figure:** Boolean network model of the MAPK regulatory network, whose involvement in bladder cancer is well established [Grieco et al., 2013].

# Boolean networks

## Boolean network

A Boolean network  $\mathcal{M}$  is defined as a 2-tuple  $(V, F)$ , where  $V = \{x_1, \dots, x_n\}$  ( $n \geq 1$ ) is a set of nodes and  $F = \{f_1, \dots, f_n\}$  is a set of Boolean functions. Each node  $x_i$  is identified as a Boolean variable, and is associated with a Boolean function  $f_i : \mathbb{B}^{|IN(f_i)|} \rightarrow \mathbb{B}$ , where  $IN(f_i)$  is the set of input nodes of  $f_i$ .

A state  $s$  is a mapping  $s : V \mapsto \mathbb{B}$  that assigns either 0 (inactive) or 1 (active) to each node.

The state space of  $\mathcal{M}$  is  $\mathbb{B}^n$ .

# Dynamics of Boolean networks

At each time step  $t$ , node  $x_i$  can update its state by

$$x_i(t + 1) = f_i(\mathbf{x}(t)).$$

An update scheme specifies which node will be updated.

Based on the update scheme, the Boolean network can transit from a state to another state (possibly identical). This is the *state transition* (denoted by  $\longrightarrow$ ).

The dynamics of a Boolean network is captured by a *State Transition Graph* (STG) that is a directed graph whose nodes represent states and whose arcs represent the state transitions.

# Synchronous vs. Asynchronous

**Synchronous Boolean networks** [Garg et al., 2008]: The updating scheme is synchronous and deterministic, i.e., all the nodes are updated simultaneously at each time step.

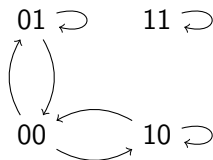
**Asynchronous Boolean networks** [Garg et al., 2008]: The updating scheme is fully asynchronous, i.e., only one node is non-deterministically selected to be updated at each time step.

Asynchronous Boolean networks are considered **more suitable** than synchronous ones in modeling biological systems [Saadatpour et al., 2010].

## Example asynchronous Boolean network

$$\begin{cases} f_1 = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \\ f_2 = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \end{cases}$$

Boolean functions



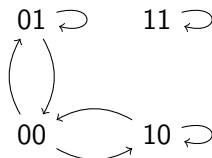
State transition graph

## Trap sets and attractors

A *trap set* is a non-empty set  $S$  of states s.t.  $\forall x \rightarrow y, x \in S \Rightarrow y \in S$ .

An *attractor* of an asynchronous Boolean network is defined as a minimal trap set that does not contain any other trap set as a subset.

$$\begin{cases} f_1 = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \\ f_2 = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \end{cases}$$



State set	Trap set?	Attractor?
{11}	yes	yes (fixed point)
{00, 01}	no	no
{00, 01, 10}	yes	yes (cyclic)
{00, 01, 10, 11}	yes	no

# Application

Besides simulation, the analysis of Boolean networks is mostly based on *attractor* computation, since those correspond roughly to observable biological *phenotypes*.

Analysis of attractors could provide new insights into systems biology [Albert and Thakar, 2014] (e.g., the origins of **cancers** [Béal et al., 2021], **SARS-CoV-2** [Noël et al., 2020], **HIV** [Oyeyemi et al., 2015]).

Attractor computation also gives a **starting point** for many control approaches for biological systems [Biane and Delaplace, 2019], which play an important role in **the development of new drugs** [Irurzun-Arana et al., 2017].



## Motivation

Attractor computation of asynchronous Boolean networks is very challenging [Mizera et al., 2019].

Several methods have been proposed for computing attractors of asynchronous Boolean networks [Garg et al., 2008, Klarner et al., 2017, Mizera et al., 2019, Benes et al., 2021, Rozum et al., 2021b, Giang and Hiraishi, 2021].

To our best knowledge, all these methods do not satisfactorily handle large and complex models, i.e., the ones that have many nodes (e.g., hundreds of nodes and beyond) and many interactions among the nodes.

The recent use of *minimal trap spaces* as very good approximations of attractors made a **real breakthrough** in that field allowing to consider medium-sized models that used to be out of reach [Klarner and Siebert, 2015].

# Contribution

In this work, we propose a novel method named mtsNFVS for **exactly** computing all the attractors of an asynchronous Boolean network based on its minimal trap spaces.

The main advantage of mtsNFVS lies in opening the chance to reach **easy cases** for the attractor computation.

# Subspaces

A *subspace*  $m$  is defined as a mapping  $m : V \mapsto \mathbb{B} \cup \star$ .

The set of fixed variables of  $m$  (denoted by  $D_m$ ) is defined by  $D_m := \{v \mid v \in V, m(v) \neq \star\}$ .

The set of free variables of  $m$  is simply  $V \setminus D_m$ .

$m$  is equivalent to a set of states

$$S[m] := \{s \in \mathbb{B}^n \mid \forall v \in D_m : s(v) = m(v)\}.$$

For example,  $m = 01\star$  means that  $D_m = \{x_1, x_2\}$ ,  $m(x_1) = 0$ ,  $m(x_2) = 1$ ,  $m(x_3) = \star$ , and  $m$  refers to the set of states  $\{010, 011\}$ .

# Minimal trap spaces

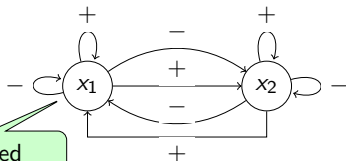
A *trap space* is a set  $S$  of states that is a subspace and also a trap set.

A trap space is *minimal* if it does not contain any smaller trap space.

Each minimal trap space contains at least one attractor, thus minimal trap spaces are good approximations for attractors of an asynchronous Boolean network [Klarner and Siebert, 2015].

# Interaction graph

The *interaction graph* of a Boolean network depicts the qualitative interactions between nodes and is usually represented as a signed directed graph on the set of nodes [Paulevé and Richard, 2012].



A **Negative Feedback Vertex Set (NFVS)** of a signed directed graph  $G$  is a set of vertices  $U$  such that  $G - U$  contains no negative cycle (cycle with an **odd** number of negative arcs). This graph has three NFVSs:

- $\{x_1\}$  (the minimum one),
- $\{x_2\}$  (the minimum one),
- $\{x_1, x_2\}$ .

## General approach of iFVS-ABN

iFVS-ABN [Giang and Hiraishi, 2021] uses an negative feedback vertex set to systematically remove arcs in the state transition graph to get a candidate set of states that covers all attractors.

Then, iFVS-ABN uses **reachability analysis** on the model to filter out this set.

The obtained result is a set of states such that there exists a **one-to-one correspondence between the set of states and the set of attractors**. This set is **sufficient** because starting from a state in an attractor, we can enumerate all other states in the attractor by listing all states reachable from this state [Garg et al., 2008].

The **correctness** of iFVS-ABN has been formally proved.

## Main issue of iFVS-ABN

The number of times iFVS-ABN need to proceed reachability analysis is equal to the number of candidate states.

iFVS-ABN tries to reduced that number of conducting Preprocessing SSF on the original candidate set.

At each iteration, Preprocessing SSF randomly chooses a node  $x_i$ , then updates the candidate set  $F$  by the set of next states of states in  $F$  by updating only node  $x_i$ . The number of iterations of Preprocessing SSF is specified by the parameter  $I\_MAX$ , which can be empirically set.

However, in the best case, iFVS-ABN must still call reachability analysis  $k$  times with  $k$  is the number of attractors. Since reachability in asynchronous Boolean networks is PSPACE-complete [Chatain et al., 2020], this is actually a **crucial issue** of iFVS-ABN.

## Main idea of mtsNFVS

mtsNFVS follows the general approach of iFVS-ABN but using minimal trap spaces of the asynchronous Boolean network to guide the attractor computation to some [easy cases](#).

First, mtsNFVS computes the set of minimal trap spaces  $M$  and the candidate set of states  $F$ .

Second, mtsNFVS computes attractors **inside** each minimal trap space  $m \in M$ .

Third, mtsNFVS computes attractors **outside** the minimal trap spaces.

Finally, mtsNFVS returns the set of states one-to-one corresponding to all attractors of the asynchronous Boolean network.



## Computing inside attractors

Each minimal trap space contains at least one attractor and minimal trap spaces are mutually disjoint.

Hence,  $F_m = F \cap S[m]$  covers all attractors inside a minimal trap space  $m$ .

If  $F_m$  contains many states, mtsNFVS uses Preprocessing SSF to shrink it.

Then, mtsNFVS performs the filtering process (using reachability analysis) on the candidate set  $F_m$  to get all the attractors contained in  $m$ .

In the best case of Preprocessing SSF, mtsNFVS can reach the case that  $F_m$  has only one state. In this case, **no reachability analysis** is needed and mtsNFVS simply adds this state into the set of inside attractors.

## Computing inside attractors (cont.)

Note that Preprocessing SSF and the filtering process can be performed on **the reduced model** of the original model.

The reduced model is obtained by fixing the fixed nodes of  $m$  in the original model, and then propagating the fixed values to the Boolean functions of the remaining nodes [Klarner and Siebert, 2015].

Since the reduced model is **much smaller** than the original model in most cases, this can reduce the computational burden significantly.

## Computing outside attractors

There may be some attractors that are **outside** of any minimal trap space of an asynchronous Boolean network.

Hence, mtsNFVS needs to process the remaining part  $F_r$  of the candidate set  $F$ .

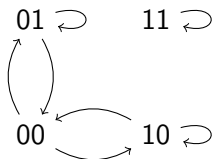
If this part contains many states, mtsNFVS uses Preprocessing SSF to shrink it.

Then, mtsNFVS performs the filtering process (using reachability analysis) on  $F_r$  to get all the attractors outside any minimal trap space.

In the best case of Preprocessing SSF, mtsNFVS can reach the case that  $F_r \subseteq S[M]$  where  $S[M] = \bigcup_{m \in M} S[m]$  is the set of states represented by all minimal trap spaces. In this case, **no reachability analysis** is needed.

## Illustrative example

$$\begin{cases} f_1 = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \\ f_2 = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \end{cases}$$



This asynchronous Boolean network has one minimal trap space  $m = 11$ . Suppose that the candidate set  $F = \{00, 11\}$ .

For minimal trap space  $m$ , we have  $F_m = F \cap S[m] = \{11\}$ . Since  $|F_m| = 1$ , mtsNFVS does not need to check reachability. mtsNFVS adds 11 to the set of inside attractors.

mtsNFVS needs to process the remaining candidate set (i.e.,  $F_r = F \setminus F_m = \{00\}$ ). 00 does not reach 11, hence mtsNFVS adds 00 to the set of outside attractors.

Finally, mtsNFVS returns  $A = \{11, 00\}$ .

## Several algorithmic improvements

By using minimal trap spaces, mtsNFVS can **open a chance to reach easy cases** for the reachability analysis in the filtering process, which are generally **unable** in iFVS-ABN.

We then propose **several algorithmic improvements** to several common constituent tasks between mtsNFVS and iFVS-ABN:

- The computation of a negative feedback vertex set
- The computation of the candidate set
- Preprocessing SSF

These algorithmic improvements are **key factors** for making the chance opened by mtsNFVS **effective**.

# Evaluation

To evaluate the effectiveness of the proposed method mtsNFVS, we conducted experiments on both **real-world biological models** and **randomly generated models**.

We compared mtsNFVS<sup>1</sup> with **five previously notable methods** including CABEAN [Mizera et al., 2019, Su and Pang, 2021], AEON [Benes et al., 2021], PyBoolNet [Klarner and Siebert, 2015, Klarner et al., 2017], pystablemotifs [Rozum et al., 2021b, Rozum et al., 2021a], and iFVS-ABN [Giang and Hiraishi, 2021].

---

<sup>1</sup>Released at <https://github.com/giang-trinh/mtsNFVS.git>

## Experimental results on real-world models

We selected seven real-world models with crucial biological motivations, which are large and complex (i.e., high average connectivity), from the literature.

Model	# nodes	# edges	avg. con.	largest SCC size
T-LGL	61	193	3.16	43
CACC	70	153	2.19	65
AD	77	206	2.68	72
IL-6	86	164	1.91	38
CELL CYCLE 2019	87	375	4.31	57
SIPC	133	449	3.38	100
CASCADE 3.0	183	603	3.30	176

The time limit for each model is **10 hours**.

## Experimental results on real-world models (cont.)

CB: CABEAN

PBN: PyBoolNet

py-sm: pystablemotifs

iFVS: iFVS-ABN

DNF: Did Not Finish

Model	# atts	CB	AEON	PBN	py-sm	iFVS	mtsNFVS
T-LGL	318	N/A	80.93	DNF	2267.40	547.17	8.16
CACC	10	N/A	469.38	DNF	16459.35	2898.34	1.96
AD	2	N/A	5646.10	DNF	626.71	84.28	3.54
IL-6	32858	N/A	3625.36	DNF	DNF	2881.75	82.68
CELL CYCLE 2019	8	DNF	DNF	DNF	DNF	DNF	80.48
SIPC	2760	N/A	DNF	DNF	DNF	DNF	1768.35
CASCADE 3.0	1	N/A	DNF	DNF	DNF	969.20	10.10



## Experimental results on real-world models (cont.)

CB: CA In all the seven models, PyBoolNet failed to compute attractors within the time limit. For each model, PyBool-  
PBN: I Net quickly computed the approximations, but took long  
py-sm: i time for checking the correctness of the approximations via  
iFVS: i  
DNF: I model checking.

Model	# atts	CB	AEON	PBN	py-sm	iFVS	mtsNFVS
T-LGL	318	N/A	80.93	DNF	2267.40	547.17	8.16
CACC	10	N/A	469.38	DNF	16459.35	2898.34	1.96
AD	2	N/A	5646.10	DNF	626.71	84.28	3.54
IL-6	32858	N/A	3625.36	DNF	DNF	2881.75	82.68
CELL CYCLE 2019	8	DNF	DNF	DNF	DNF	DNF	80.48
SIPC	2760	N/A	DNF	DNF	DNF	DNF	1768.35
CASCADE 3.0	1	N/A	DNF	DNF	DNF	969.20	10.10

## Experimental results on real-world models (cont.)

CB In all the seven models, CABEAN failed to finish the computation.  
PB In six models (except the CELL CYCLE 2019 model), CABEAN  
py- terminated before exceeding the time limit and the segmentation  
iFV fault error was printed because the decomposition of CABEAN  
DN may not reduce the complexity of the model enough to continue  
with its attractor search.

Model	# atts	CB	AEON	PBN	py-sm	iFVS	mtsNFVS
T-LGL	318	N/A	80.93	DNF	2267.40	547.17	8.16
CACC	10	N/A	469.38	DNF	16459.35	2898.34	1.96
AD	2	N/A	5646.10	DNF	626.71	84.28	3.54
IL-6	32858	N/A	3625.36	DNF	DNF	2881.75	82.68
CELL CYCLE 2019	8	DNF	DNF	DNF	DNF	DNF	80.48
SIPC	2760	N/A	DNF	DNF	DNF	DNF	1768.35
CASCADE 3.0	1	N/A	DNF	DNF	DNF	969.20	10.10

## Experimental results on real-world models (cont.)

CB: CAEON failed to compute attractors within the time limit for the three models (SPIC, CELL CYCLE 2019, and CASCADE 3.0). In the four remaining models, it succeeded to finish the computation in reasonable time. As compared to PyBoolNet and CABEAN, AEON is more efficient.

DNF: Did Not Finish

Model	# atts	CB	AEON	PBN	py-sm	iFVS	mtsNFVS
T-LGL	318	N/A	80.93	DNF	2267.40	547.17	8.16
CACC	10	N/A	469.38	DNF	16459.35	2898.34	1.96
AD	2	N/A	5646.10	DNF	626.71	84.28	3.54
IL-6	32858	N/A	3625.36	DNF	DNF	2881.75	82.68
CELL CYCLE 2019	8	DNF	DNF	DNF	DNF	DNF	80.48
SIPC	2760	N/A	DNF	DNF	DNF	DNF	1768.35
CASCADE 3.0	1	N/A	DNF	DNF	DNF	969.20	10.10

## Experimental results on real-world models (cont.)

We first note that this method computes exact fixed points and quasi-attractors, which correspond to but may not be identical to cyclic attractors of an asynchronous Boolean network. It only returned results for three models, but these results are not exact by comparing with the results of AEON. Hence, `pystablemotifs` is unable to fully analyze any model.

Model	# atts	CB	AEON	PBN	py-sm	iFVS	mtsNFVS
T-LGL	318	N/A	80.93	DNF	2267.40	547.17	8.16
CACC	10	N/A	469.38	DNF	16459.35	2898.34	1.96
AD	2	N/A	5646.10	DNF	626.71	84.28	3.54
IL-6	32858	N/A	3625.36	DNF	DNF	2881.75	82.68
CELL CYCLE 2019	8	DNF	DNF	DNF	DNF	DNF	80.48
SIPC	2760	N/A	DNF	DNF	DNF	DNF	1768.35
CASCADE 3.0	1	N/A	DNF	DNF	DNF	969.20	10.10

## Experimental results on real-world models (cont.)

In the five models (T-LGL, CACC, AD, IL-6, and CASCADE 3.0), iFVS-ABN succeeded to finish the computation in reasonable time. The running time of iFVS-ABN is comparable to that of AEON for the T-LGL, CACC, AD, and IL-6 models. It can handle CASCADE 3.0, whereas PyBoolNet, CABEAN, and AEON cannot.

Model	# atts	CB	AEON	PBN	py-sm	iFVS	mtsNFVS
T-LGL	318	N/A	80.93	DNF	2267.40	547.17	8.16
CACC	10	N/A	469.38	DNF	16459.35	2898.34	1.96
AD	2	N/A	5646.10	DNF	626.71	84.28	3.54
IL-6	32858	N/A	3625.36	DNF	DNF	2881.75	82.68
CELL CYCLE 2019	8	DNF	DNF	DNF	DNF	DNF	80.48
SIPC	2760	N/A	DNF	DNF	DNF	DNF	1768.35
CASCADE 3.0	1	N/A	DNF	DNF	DNF	969.20	10.10

## Experimental results on real-world models (cont.)

Our proposed method succeeded to finish the computation within the time limit for all the seven models. The running time for each model (except SIPC) is very short (less than two minutes).

CB: CA

PBN: I

py-sm: I

iFVS: iFVS-ABN

DNF: Did Not Finish

Model	# atts	CB	AEON	PBN	py-sm	iFVS	mtsNFVS
T-LGL	318	N/A	80.93	DNF	2267.40	547.17	8.16
CACC	10	N/A	469.38	DNF	16459.35	2898.34	1.96
AD	2	N/A	5646.10	DNF	626.71	84.28	3.54
IL-6	32858	N/A	3625.36	DNF	DNF	2881.75	82.68
CELL CYCLE 2019	8	DNF	DNF	DNF	DNF	DNF	80.48
SIPC	2760	N/A	DNF	DNF	DNF	DNF	1768.35
CASCADE 3.0	1	N/A	DNF	DNF	DNF	969.20	10.10

## Experimental results on random models

We randomly generated a set of models by using Bool Net R package [Müssel et al., 2010].

This set includes 40  $N$ - $K$  models [Kauffman, 1969] with network size  $n \in \{100, 150, 200, 250\}$  and  $K = 2$  (i.e., each node has exactly  $K = 2$  input nodes). 10 instances were generated for each  $n$ .

The time limit for each model is 10 hours.

We reported the number of failed models for each of the considered methods.

## Experimental results on random models (cont.)

Method	$n = 100$	$n = 150$	$n = 200$	$n = 250$
CABEAN	10	10	10	10
PyBoolNet	10	10	10	10
AEON	2	10	10	10
pystablemotifs	9	9	10	10
iFVS-ABN	1	6	9	10
mtsNFVS	0	0	0	0



## Experimental results on random models (cont.)

Method	$n = 100$	$n = 150$	$n = 200$	$n = 250$
CABEAN	10	10	10	10
PyBoolNet	10	10	10	10
AEON	2	10	10	10
pystablemotifs	9	9	10	10
iFVS-ABN	1	6	9	10
mtsNFVS	0	0	0	0

CABEAN and PyBoolNet failed to compute attractors within the time limit for all the models.

## Experimental results on random models (cont.)

Method	$n = 100$	$n = 150$	$n = 200$	$n = 250$
CABEAN	10	10	10	10
PyBoolNet	10	10	10	10
AEON	2	10	10	10
pystablemotifs	9	9	10	10
iFVS-ABN	1	6	9	10
mtsNFVS	0	0	0	0

AEON and pystablemotif can handle only a few models for  $n = 100$  and their numbers of failures rapidly approach 10 for larger  $n$ .

## Experimental results on random models (cont.)

Method	$n = 100$	$n = 150$	$n = 200$	$n = 250$
CABEAN	10	10	10	10
PyBoolNet	10	10	10	10
AEON	2	10	10	10
pystablemotifs	9	9	10	10
iFVS-ABN	1	6	9	10
mtsNFVS	0	0	0	0

Although iFVS-ABN is better, its number of failures drastically increases.

## Experimental results on random models (cont.)

Method	$n = 100$	$n = 150$	$n = 200$	$n = 250$
CABEAN	10	10	10	10
PyBoolNet	10	10	10	10
AEON	2	10	10	10
pystablemotifs	9	9	10	10
iFVS-ABN	1	6	9	10
mtsNFVS	0	0	0	0

In the case of mtsNFVS, it can handle all the models. We also reported that the running time of mtsNFVS in each model is always less than 800 seconds.

## Experimental results on random models (cont.)

Method	$n = 100$	$n = 150$	$n = 200$	$n = 250$
CABEAN	10	10	10	10
PyBoolNet	10	10	10	10
AEON	2	10	10	10
pystablemotifs	9	9	10	10
iFVS-ABN	1	6	9	10
mtsNFVS	0	0	0	0

In summary, mtsNFVS completely outperforms all the other methods and it can handle large-scale models.

# Conclusion

**Computing attractors of asynchronous Boolean networks** is crucial but challenging.

We proposed the novel method mtsNFVS for computing all the attractors of an asynchronous Boolean network.

The evaluation on large models from the literature and randomly generated models shows that mtsNFVS completely outperforms the state-of-the-art method CABEAN and other previously notable methods.

We believe that mtsNFVS opens up the way to much better analysis of large asynchronous Boolean networks, hence **biological systems**.

## Future work

In the future, we plan to conduct a more comprehensive comparison among the existing methods for attractor detection in asynchronous Boolean networks (i.e., CABEAN, AEON, PyBoolNet, pystablemotifs, iFVS-ABN, and mtsNFVS). To do this, we shall need to run experiments on more real-world models as well as randomly generated models of larger size.




In addition, the processes of mtsNFVS for the set of minimal trap spaces seem potentially to be paralleled. Hence, we plan to investigate deeply the parallelization capability of mtsNFVS.

Finally, proposing heuristics to help Preprocessing SSF converge more quickly is also a potential improvement to mtsNFVS.




Thank you for your attention!



# References I

-  Albert, R. and Thakar, J. (2014).  
Boolean modeling: a logic-based dynamic approach for understanding signaling and regulatory networks and for making useful predictions.  
*Wiley Interdiscip. Rev. Syst. Biol. Med.*, 6(5):353–369.
-  Béal, J., Pantolini, L., Noël, V., Barillot, E., and Calzone, L. (2021).  
Personalized logical models to investigate cancer response to BRAF treatments in melanomas and colorectal cancers.  
*PLoS Comput. Biol.*, 17(1).
-  Benes, N., Brim, L., Pastva, S., and Safránek, D. (2021).  
Computing bottom SCCs symbolically using transition guided reduction.  
In *International Conference on Computer Aided Verification*, pages 505–528. Springer.

## References II

-  Biane, C. and Delaplace, F. (2019).  
Causal reasoning on Boolean control networks based on abduction:  
Theory and application to cancer drug discovery.  
*IEEE ACM Trans. Comput. Biol. Bioinform.*, 16(5):1574–1585.
-  Chatain, T., Haar, S., Kolcák, J., Paulevé, L., and Thakkar, A.  
(2020).  
Concurrency in Boolean networks.  
*Nat. Comput.*, 19(1):91–109.
-  Garg, A., Cara, A. D., Xenarios, I., Mendoza, L., and Micheli, G. D.  
(2008).  
Synchronous versus asynchronous modeling of gene regulatory  
networks.  
*Bioinform.*, 24(17):1917–1925.

## References III



Giang, T. V. and Hiraishi, K. (2021).

An improved method for finding attractors of large-scale asynchronous Boolean networks.

In *2021 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–9. IEEE.






Grieco, L., Calzone, L., Bernard-Pierrot, I., Radvanyi, F., Kahn-Perlès, B., and Thieffry, D. (2013).

Integrative modelling of the influence of MAPK network on cancer cell fate decision.

*PLoS Computational Biology*, 9(10):e1003286.

## References IV

-  Irurzun-Arana, I., Pastor, J. M., Trocóniz, I. F., and Gómez-Mantilla, J. D. (2017).  
Advanced Boolean modeling of biological networks applied to systems pharmacology.  
*Bioinform.*, 33(7):1040–1048.
-  Kauffman, S. (1969).  
Metabolic stability and epigenesis in randomly constructed genetic nets.  
*J. Theor. Biol.*, 22(3):437–467.
-  Klarner, H. and Siebert, H. (2015).  
Approximating attractors of Boolean networks by iterative CTL model checking.  
*Front. Bioeng. Biotechnol.*, 3.

## References V



Klarner, H., Streck, A., and Siebert, H. (2017).

Pyboolnet: a python package for the generation, analysis and visualization of Boolean networks.

*Bioinform.*, 33(5):770–772.



Mizera, A., Pang, J., Qu, H., and Yuan, Q. (2019).

Taming asynchrony for attractor detection in large Boolean networks.

*IEEE ACM Trans. Comput. Biol. Bioinform.*, 16(1):31–42.






Müssel, C., Hopfensitz, M., and Kestler, H. A. (2010).




BoolNet—an R package for generation, reconstruction and analysis of Boolean networks.

*Bioinform.*, 26(10):1378–1380.

## References VI

-  Noël, V., Carbonell, J., Ponce de Leon, M., Soliman, S., Niarakis, A., Calzone, L., Barillot, E., Valencia, A., and Montagud, A. (2020). PhysiBoSS-COVID: the Boolean modelling of COVID-19 signalling pathways in a multicellular simulation framework allows for the uncovering of mechanistic insights.
-  Oyeyemi, O. J., Davies, O., Robertson, D. L., and Schwartz, J. (2015).  
A logical model of HIV-1 interactions with the T-cell activation signalling pathway.  
*Bioinform.*, 31(7):1075–1083.
-  Paulevé, L. and Richard, A. (2012).  
Static analysis of Boolean networks based on interaction graphs: A survey.  
*Electron. Notes Theor. Comput. Sci.*, 284:93–104.

## References VII

-  Rozum, J. C., Deritei, D., Park, K. H., Gómez Tejeda Zañudo, J., and Albert, R. (2021a).  
pystablemotifs: Python library for attractor identification and control in Boolean networks.  
*Bioinform.*
-  Rozum, J. C., Zañudo, J. G. T., Gan, X., Deritei, D., and Albert, R. (2021b).  
Parity and time reversal elucidate both decision-making in empirical models and attractor scaling in critical Boolean networks.  
*Sci. Adv.*, 7(29).
-  Saadatpour, A., Albert, I., and Albert, R. (2010).  
Attractor analysis of asynchronous Boolean models of signal transduction networks.  
*J. Theor. Biol.*, 266(4):641–656.

## References VIII



Su, C. and Pang, J. (2021).

Cabean 2.0: Efficient and efficacious control of asynchronous Boolean networks.

In *International Symposium on Formal Methods*, pages 581–598.

Springer.