Trap spaces of Boolean networks are conflict-free siphons of their Petri net encoding

Van-Giang Trinh[1], Belaid Benhamou[1], and Sylvain Soliman[2]

[1]LIS, Aix-Marseille University, Marseille, France
[2]Lifeware team, Inria Saclay center, Palaiseau, France

May 25, 2023

Van-Giang Trinh, Belaid Benhamou, Kunihiko Hiraishi, & Sylvain Soliman (2022, August). Minimal trap spaces of logical models are maximal siphons of their Petri net encoding. In *International Conference on Computational Methods in Systems Biology* (pp. 158–176). Springer.

Van-Giang Trinh, Belaid Benhamou, & Sylvain Soliman (2023). Trap spaces of Boolean networks are conflict-free siphons of their Petri net encoding. *Theoretical Computer Science*. (under review)

Boolean modeling

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

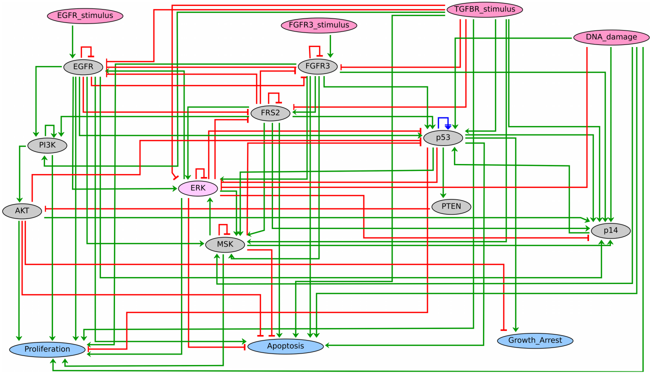


Figure: Boolean model of the MAPK regulatory network, whose involvement in bladder cancer is well established [Grieco et al., 2013].

Boolean networks

Boolean network

A Boolean network $\mathcal{N}$ is defined as a 2-tuple (V, F), where $V = \{x_1, ..., x_n\}$ $(n \geq 1)$ is a set of nodes and $F = \{f_1, ..., f_n\}$ is a set of Boolean functions. Each node x_i is identified as a Boolean variable, and is associated with a Boolean function $f_i : \{0, 1\}^{|IN(f_i)|} \rightarrow \{0, 1\}$, where $IN(f_i)$ is the set of input nodes of f_i.

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

The state space of $\mathcal{N}$ is $\{0, 1\}^n$.

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.

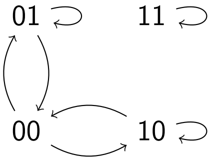
Based on the update scheme, the Boolean network can transit from a state to another state (possibly identical). This is the *state transition* (denoted by $\longrightarrow$).

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.

Example 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

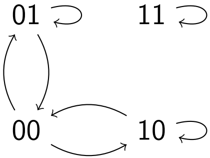
Fully asynchronous update scheme: only one node is non-deterministically selected in order to be updated at each time step.

Trap sets and attractors

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

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

$$\begin{cases} f_1 = (x_1 \land x_2) \lor (\neg x_1 \land \neg x_2) \\ f_2 = (x_1 \land x_2) \lor (\neg x_1 \land \neg x_2) \end{cases}$$



State set	Trap set?	Attractor?
$\{11\}$	yes	yes
$\{00, 01\}$	no	no
$\{00, 01, 10\}$	yes	yes
$\{00, 01, 10, 11\}$	yes	no

Application

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

Analysis of attractors could provide new insights into systems biology [Wang et al., 2012] (e.g., the origins of cancers [Montagud et al., 2021], SARS-CoV-2 [Ostaszewski et al., 2021], HIV [Oyeyemi et al., 2014]).

Attractor computation also gives a starting point for many control approaches for biological systems [Fontanals et al., 2020], which play an important role in the development of new drugs [Balbas-Martinez et al., 2018].

Motivation

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

The recent use of *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 et al., 2015].

However, with the continuing increase in model-size, the state-of-the-art computation of minimal trap spaces based on *prime-implicants* (e.g., PyBoolNet [Klarner et al., 2017]) shows its limits because there can be a huge number of implicants and their computation is a demanding task.

The recent method presented in [Chevalier et al., 2019] for computing minimal trap spaces avoids the prime-implicants computation. It is implemented in the tool mpbn [Paulevé et al., 2020] and can handle very large but only *locally-monotonic* Boolean networks.

Contribution

In this work, we make a connection between trap spaces of Boolean networks and siphons of Petri nets, which has not yet been explored before.

This connection can be a useful technique for studying properties of trap spaces in Boolean networks.

Based on the connection, we propose an alternative approach to compute minimal trap spaces of a Boolean network.

Note that, these results are applicable for general Boolean networks (i.e., both locally-monotonic and non-locally-monotonic ones).

Subspaces

A *subspace* m is defined as a mapping $m : V \mapsto \{0, 1, \star\}$.

For example, $m = 01\star$ means that

- x_1 and x_2 are fixed variables and $m(x_1) = 0$, $m(x_2) = 1$;
- x_3 is a free variable and $m(x_3) = \star$;
- m refers to the set of states $\{010, 011\}$.

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.

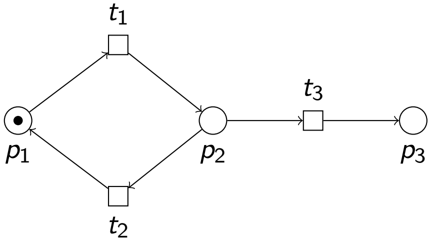
State set	Trap set?	Subspace?	Trap space?
$\{11\}$	yes	11	yes
$\{00, 01\}$	no	$0\star$	no
$\{00, 01, 10\}$	yes	no	no
$\{00, 01, 10, 11\}$	yes	$\star\star$	yes

Note that trap spaces of a Boolean network are independent of the update scheme of this model [Klarner et al., 2015].

Petri nets and their siphons

Bipartite graph

Places P
Transitions T
Weighted arcs W



Siphon

A *siphon* of a Petri net (P, T, W) is a set of places S such that:

$$\forall t \in T, S \cap succ(t) \neq \emptyset \Rightarrow S \cap pred(t) \neq \emptyset.$$

Here: $\emptyset$, $\{p_1, p_2\}$, $\{p_1, p_2, p_3\}$

Once a siphon is unmarked, it remains unmarked.

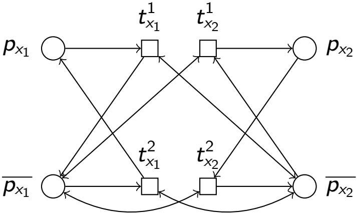
Petri net of a Boolean network

The original encoding was established in [Chaouiya et al., 2004].

Two places for each node: $v \rightsquigarrow p_v, \overline{p_v}$

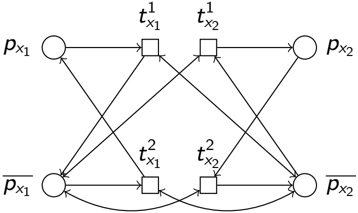
Solutions of $f_v \not\leftrightarrow v \rightsquigarrow$ transitions from p_v to $\overline{p_v}$ (and back)

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



Conflict-free siphons

A siphon is called **conflict-free** if it does not contain both p_v and $\overline{p_v}$ for all $v \in V$.



Siphon	Conflict-free?
$\emptyset$	yes
$\{\overline{p_{x_1}}, \overline{p_{x_2}}\}$	yes
$\{p_{x_1}, \overline{p_{x_1}}\}$	no
$\{p_{x_2}, \overline{p_{x_2}}\}$	no

A conflict-free siphon is *maximal* if it is not a subset of any other conflict-free siphon.

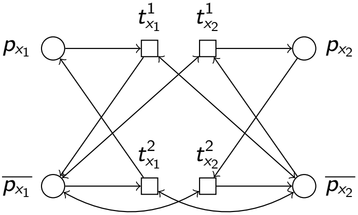
Conflict-free siphons are trap spaces

Theorem 1

Let $\mathcal{N}$ be a Boolean network and $\mathcal{P}$ be its Petri net encoding. There is a one-to-one correspondence between the set of **trap spaces** of $\mathcal{N}$ and the set of **conflict-free siphons** of $\mathcal{P}$.

$$\begin{cases} f_1 = (x_1 \land x_2) \lor (\neg x_1 \land \neg x_2) \\ f_2 = (x_1 \land x_2) \lor (\neg x_1 \land \neg x_2) \end{cases}$$

Trap space	Conflict-free siphon
⋆⋆	$\emptyset$
11	$\{\overline{p_{x_1}}, \overline{p_{x_2}}\}$



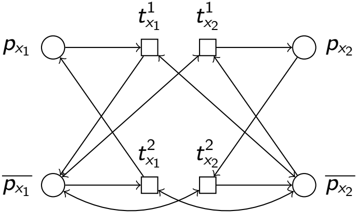
Note that the proof of Theorem 1 can give another way to prove the independence of trap spaces on the update scheme.

Maximal conflict-free siphons are minimal trap spaces

Theorem 2

Let $\mathcal{N}$ be a Boolean network and $\mathcal{P}$ be its Petri net encoding. There is a one-to-one correspondence between the set of **minimal trap spaces** of $\mathcal{N}$ and the set of **maximal conflict-free siphons** of $\mathcal{P}$.

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



Trap space	Conflict-free siphon
$\star\star$	$\emptyset$
11	$\{\overline{p_{x_1}}, \overline{p_{x_2}}\}$

A theoretical application

Theorem 3

Let $\mathcal{N}$ be a Boolean network. For any two distinct minimal trap spaces m_1 and m_2 of $\mathcal{N}$, we have that $\mathcal{S}_{\mathcal{N}}[m_1] \cap \mathcal{S}_{\mathcal{N}}[m_2] = \emptyset$.

The above property is important, but it has surprisingly not been formally proved.

We formally prove it by using the connection between trap spaces of Boolean networks and siphons of Petri nets.

Note that it would be not difficult to obtain a direct proof on trap spaces for this property. However, we emphasize here the potential of using the connection to explore and prove properties of trap spaces in Boolean networks.

A computational application

From Theorem 2, we propose an alternative approach for enumerating minimal trap spaces of a Boolean network $\mathcal{N}$.

- Build the Petri net encoding $\mathcal{P}$ of $\mathcal{N}$.
- Enumerate all maximal conflict-free siphons of $\mathcal{P}$.
- Convert the obtained maximal conflict-free siphons into the corresponding minimal trap spaces of $\mathcal{N}$.

Petri net transformation

Transforming a Boolean network into its Petri net encoding can be done via computing Disjunctive Normal Forms (DNF) of each Boolean function [Chatain et al., 2014].

Though this might appear quite computationally intensive in some cases it is important to remark first that contrary to the prime-implicants case, there is no need to find *minimal* DNFs.

We use the above transformation in our proposed approach. The implementation uses BDDs[1].

[1] `https://github.com/cjdrake/pyeda`

Characterization of conflict-free siphons

Characterize all generic siphons of the encoded Petri net as a system of Boolean rules. For each pair (p, t) where $p \in P, t \in T, t \in pred(p)$, we have

$$p \in S \Rightarrow \bigvee_{p' \in pred(t)} p' \in S$$

Add to the system the Boolean rules representing the conflict-freeness for every node v.

$$(p_v \notin S) \vee (\overline{p_v} \notin S)$$

The final system fully characterizes all conflict-free siphons of the encoded Petri net.

Four possible methods

Based on the above system of Boolean rules, we can have four possible implementations for the alternative approach:

- Answer Set Programming (ASP)
- MaxSAT
- Constraint Programming (CP)
- Integer Linear Programming (ILP)

For the ASP implementation, we compute all set-inclusion maximal answer sets (equivalent to maximal conflict-free siphons) of the ASP via using a built-in feature in clingo.

For other implementations, we need to use iterative procedures.

Computation of special types of trap spaces

In the field of systems biology, biologists may want to compute more special types of trap spaces beyond minimal trap spaces [Klarner et al., 2017], which also play crucial roles in analysis and control of Boolean networks [Fontanals et al., 2020, Rozum et al., 2021].

We show that our proposed methods can be easily adjusted to compute several popular types of trap spaces.

- maximal trap spaces
- fixed points
- trap spaces *intersecting* a given subspace m^*
- trap spaces that are *inside* a given subspace m^*

Locally-monotonic vs. non-locally-monotonic

A Boolean function is *locally-monotonic* if it can be represented by a formula in disjunctive normal form in which all occurrences of any given literal are either negated or non-negated [Anthony, 2001].

A Boolean network is said to be locally-monotonic if all its Boolean functions are locally-monotonic. Otherwise, this model is said to be non-locally-monotonic.

Function	locally-monotonic?	non-locally-monotonic?
$x \wedge y$	yes	no
$(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$	no	yes

General models = locally-monotonic models + non-locally-monotonic models

Locally-monotonic vs. non-locally-monotonic (cont.)

Method	Applicable domain
`mpbn` [Paulevé et al., 2020]	locally-monotonic
`PyBoolNet` [Klarner et al., 2017]	general
our proposed methods	general

`mpbn` is specifically designed for exploiting the locally-monotonicity [Paulevé et al., 2020]. Hence, for locally-monotonic Boolean networks, it might have advantages over other methods.

Experiments

We implemented the proposed methods in a Python tool called trappist[2].

- trap-asp: ASP method
- trap-sat: MaxSAT method
- trap-cp: CP method
- trap-ilp: ILP method

We compared them with two state-of-the-art methods PyBoolNet and mpbn on both real-world and randomly generated models.

We searched for the first 1000 minimal trap spaces for each model.

[2] https://github.com/soli/trap-spaces-as-siphons

Results of the four proposed methods

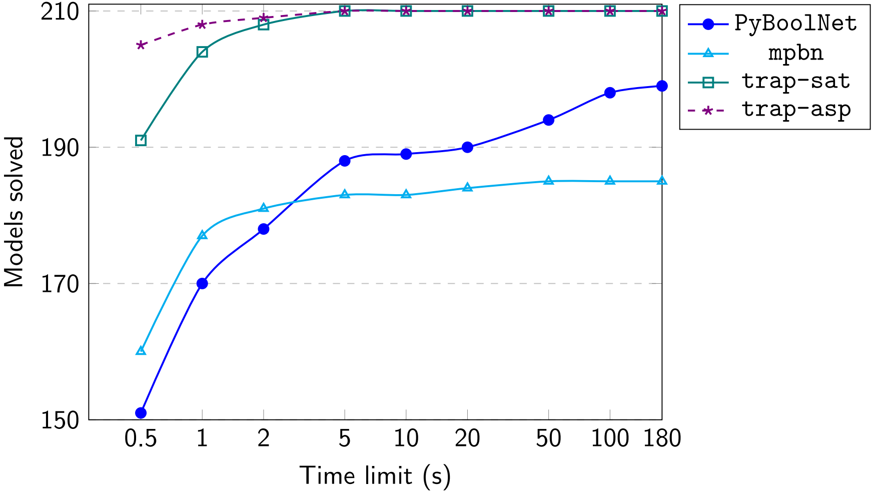
`trap-cp` and `trap-ilp` gave poor performance compared to `trap-sat` and `trap-asp`.

However, there are many models that `trap-cp` and `trap-ilp` could handle in time, whereas `PyBoolNet` and `mpbn` could not.

We will focus on `trap-sat` and `trap-asp` for further analysis.

Results on real-world models

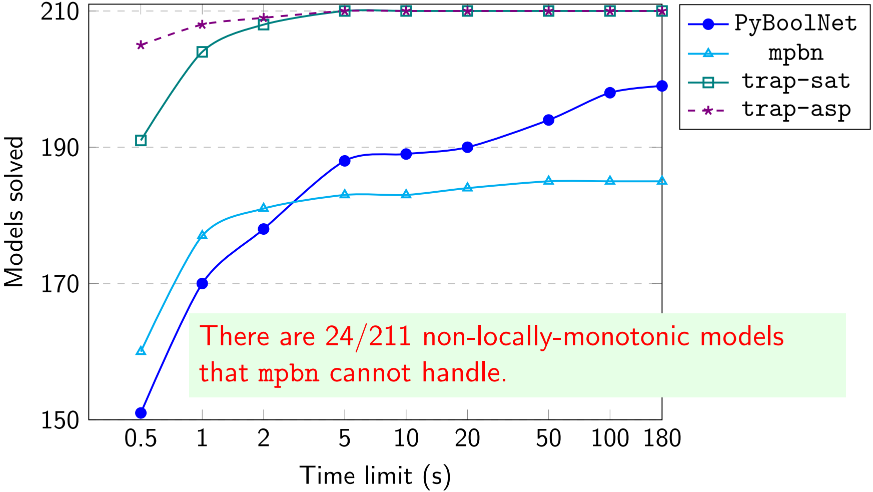
Tested on 211 real-worlds obtained from the BBM repository[3].



[3] https://github.com/sybila/biodivine-boolean-models

Results on real-world models

Tested on 211 real-worlds obtained from the BBM repository[3].

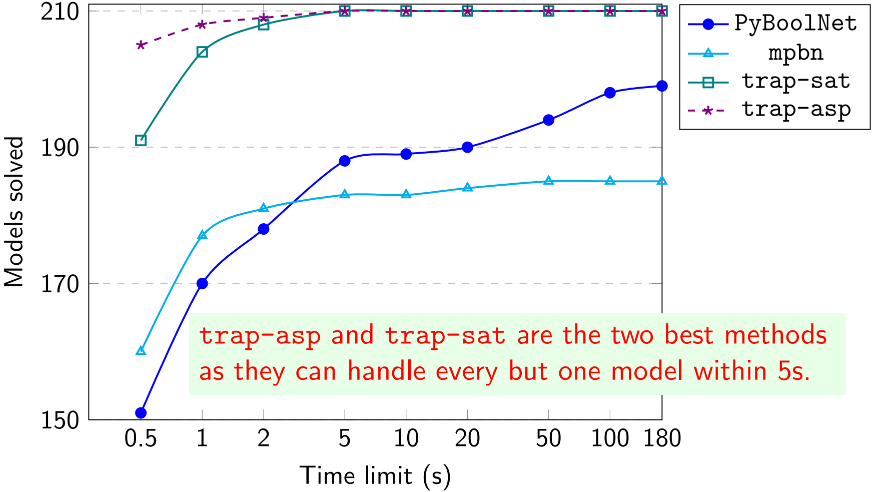


There are 24/211 non-locally-monotonic models that mpbn cannot handle.

[3] https://github.com/sybila/biodivine-boolean-models

Results on real-world models

Tested on 211 real-worlds obtained from the BBM repository[3].

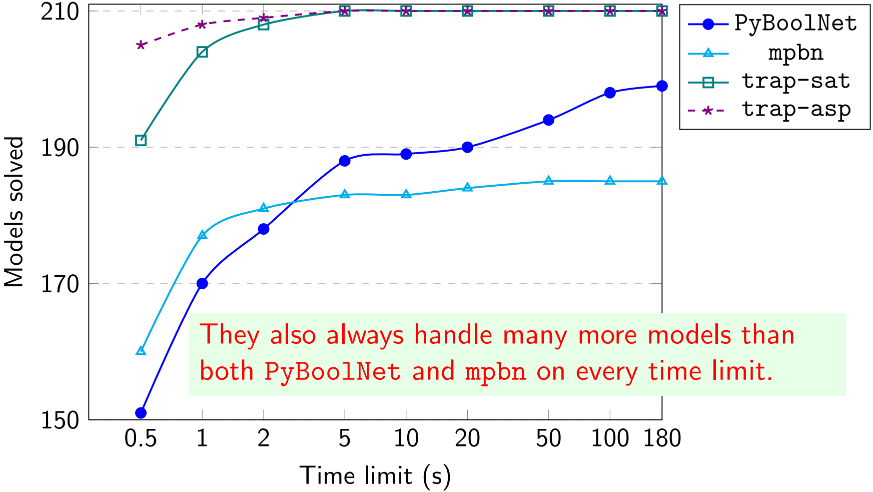


trap-asp and trap-sat are the two best methods as they can handle every but one model within 5s.

[3] https://github.com/sybila/biodivine-boolean-models

Results on real-world models

Tested on 211 real-worlds obtained from the BBM repository[3].

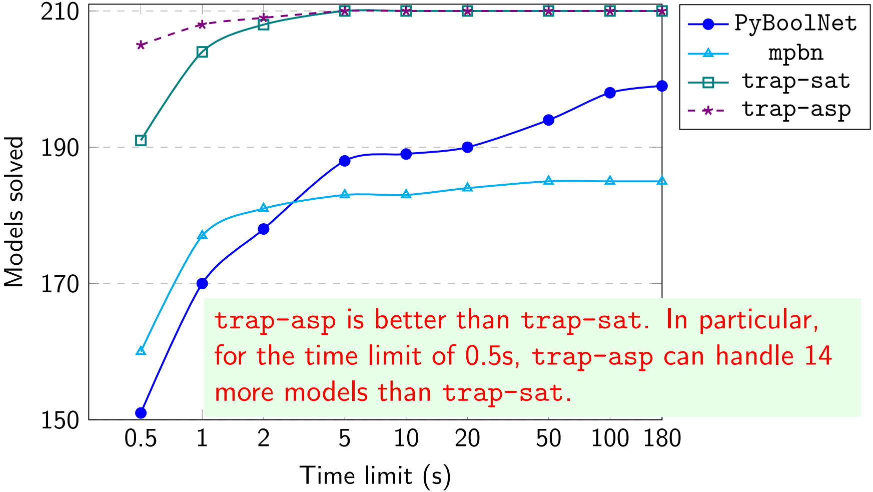


They also always handle many more models than both PyBoolNet and mpbn on every time limit.

[3] https://github.com/sybila/biodivine-boolean-models

Results on real-world models

Tested on 211 real-worlds obtained from the BBM repository[3].

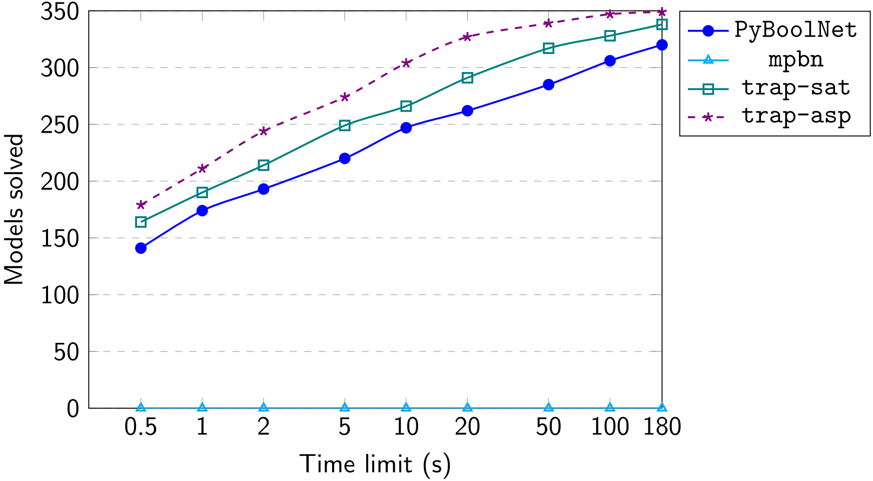


trap-asp is better than trap-sat. In particular, for the time limit of 0.5s, trap-asp can handle 14 more models than trap-sat.

[3] https://github.com/sybila/biodivine-boolean-models

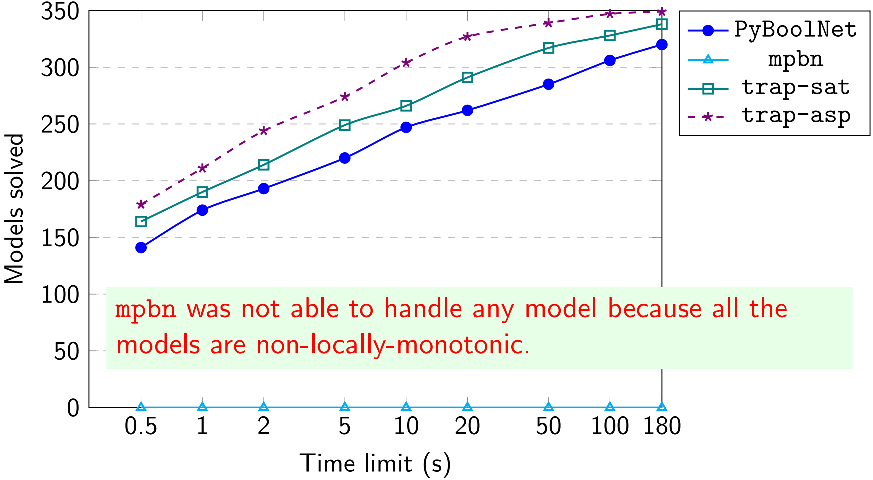
Results on randomly generated models

Tested on 350 random models generated by using the BoolNet R package [Müssel et al., 2010].



Results on randomly generated models

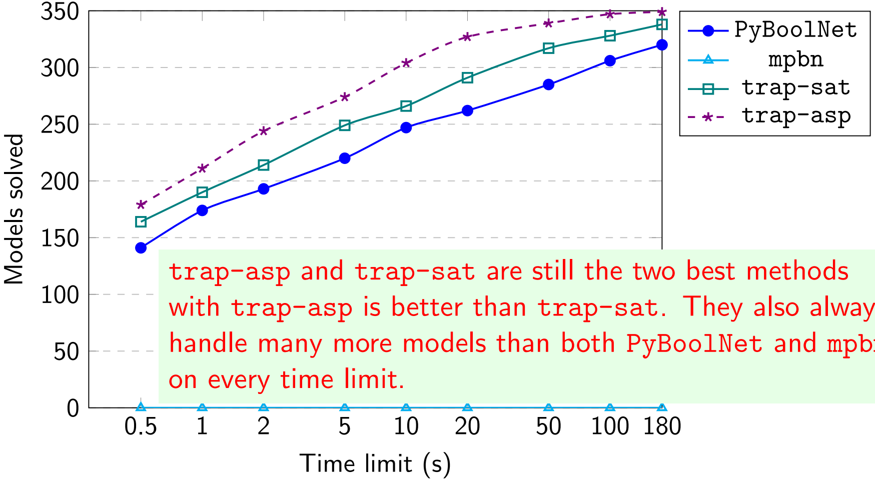
Tested on 350 random models generated by using the BoolNet R package [Müssel et al., 2010].



mpbn was not able to handle any model because all the models are non-locally-monotonic.

Results on randomly generated models

Tested on 350 random models generated by using the `BoolNet R` package [Müssel et al., 2010].



`trap-asp` and `trap-sat` are still the two best methods with `trap-asp` is better than `trap-sat`. They also always handle many more models than both `PyBoolNet` and `mpbn` on every time limit.

Conclusion

Trap spaces are important in Boolean network analysis.

We linked the concept of trap spaces in the Boolean networks field and the concept of siphons on the Petri nets field.

The connection can be a useful technique for studying properties of trap spaces in Boolean networks.

We proposed a new approach for the enumeration of minimal trap spaces (also other types of trap spaces) in Boolean networks.

The evaluation on real-world and randomly generated models shows that our approach can scale up much better than the state-of-the-art methods.

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

Future work

Improve our approach to deal with larger and more complex models.

Extend our approach to that for multi-level logical models, which can model biological systems better.

Finally, we think that the links between Petri nets and Boolean networks that we presented here might have deeper roots. Exploring those connections might lead both to interesting topics of research for Petri nets, like a notion of trap spaces, and for Boolean networks.

Thank you for your attention!

References I

Anthony, M. (2001).
Discrete Mathematics of Neural Networks.
Society for Industrial and Applied Mathematics.

Balbas-Martinez, V., Ruiz-Cerdá, L., Irurzun-Arana, I., González-García, I., Vermeulen, A., Gómez-Mantilla, J. D., and Trocóniz, I. F. (2018).
A systems pharmacology model for inflammatory bowel disease.
PloS One, 13(3):e0192949.

Chaouiya, C., Remy, E., Ruet, P., and Thieffry, D. (2004).
Qualitative modelling of genetic networks: From logical regulatory graphs to standard Petri nets.
In Cortadella, J. and Reisig, W., editors, *Applications and Theory of Petri Nets 2004, 25th International Conference, ICATPN 2004,*

References II

Bologna, Italy, June 21-25, 2004, Proceedings, volume 3099 of *Lecture Notes in Computer Science*, pages 137–156. Springer.

Chatain, T., Haar, S., Jezequel, L., Paulevé, L., and Schwoon, S. (2014).
Characterization of reachable attractors using Petri net unfoldings.
In Mendes, P., Dada, J. O., and Smallbone, K., editors, *Computational Methods in Systems Biology - 12th International Conference, CMSB 2014, Manchester, UK, November 17-19, 2014, Proceedings*, volume 8859 of *Lecture Notes in Computer Science*, pages 129–142. Springer.

References III

Chevalier, S., Froidevaux, C., Paulevé, L., and Zinovyev, A. Y. (2019).
Synthesis of Boolean networks from biological dynamical constraints using answer-set programming.
In *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019*, pages 34–41. IEEE.

Fontanals, L. C., Tonello, E., and Siebert, H. (2020).
Control strategy identification via trap spaces in Boolean networks.
In Abate, A., Petrov, T., and Wolf, V., editors, *Computational Methods in Systems Biology - 18th International Conference, CMSB 2020, Konstanz, Germany, September 23-25, 2020, Proceedings*, volume 12314 of *Lecture Notes in Computer Science*, pages 159–175. Springer.

References IV

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.

Klarner, H., Bockmayr, A., and Siebert, H. (2015).
Computing maximal and minimal trap spaces of Boolean networks.
Nat. Comput., 14(4):535–544.

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.

References V

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.

Montagud, A., Béal, J., Tobalina, L., Traynard, P., Subramanian, V., Szalai, B., Alföldi, R., Puskás, L., Valencia, A., Barillot, E., Saez-Rodriguez, J., and Calzone, L. (2021).
Patient-specific Boolean models of signaling networks guide personalized treatments.
bioRxiv.

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

Ostaszewski, M., Niarakis, A., Mazein, A., Kuperstein, I., Phair, R., Orta-Resendiz, A., Singh, V., Aghamiri, S. S., Acencio, M. L., Glaab, E., et al. (2021).
COVID19 Disease Map, a computational knowledge repository of virus–host interaction mechanisms.
Mol Syst Biol., 17(10):e10387.

Oyeyemi, O. J., Davies, O., Robertson, D. L., and Schwartz, J.-M. (2014).
A logical model of HIV-1 interactions with the T-cell activation signalling pathway.
Bioinformatics, 31(7):1075–1083.

References VII

Paulevé, L., Kolčák, J., Chatain, T., and Haar, S. (2020).
Reconciling qualitative, abstract, and scalable modeling of biological networks.
Nat. Commun., 11(1).

Rozum, J. C., Zañudo, J. G. T., Gan, X., Deritei, D., and Albert, R. (2021).
Parity and time reversal elucidate both decision-making in empirical models and attractor scaling in critical Boolean networks.
Sci. Adv., 7(29):eabf8124.

Wang, R.-S., Saadatpour, A., and Albert, R. (2012).
Boolean modeling in systems biology: an overview of methodology and applications.
Phys. Biol., 9(5):055001.