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

December 05, 2023

About me

Information

- Name: Van-Giang Trinh (Trịnh Văn Giang in Vietnamese)
- Email: trinh.van-giang@lis-lab.fr
- Website: `https://giang-trinh.github.io/`

Education and work

- 2018–2021: Ph.D., Information Science, Japan Advanced Institute of Science and Technology, Japan
- 2022–present: Postdoc, LIRICA team, LIS, Aix-Marseille University

Research interests

- theoretical computer science, artificial intelligence, and computational systems biology
- Boolean networks, Petri nets, ASP, and their applications to modeling, analysis, and control of biological systems

Van-Giang Trinh, Belaid Benhamou, Kunihiko Hiraishi, and Sylvain Soliman. "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. Cham: Springer International Publishing, 2022

Van-Giang Trinh, Belaid Benhamou, and Sylvain Soliman. "Trap spaces of Boolean networks are conflict-free siphons of their Petri net encoding." *Theoretical Computer Science* 971 (2023): 114073.

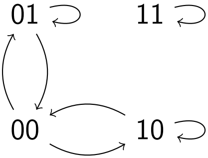
Boolean network

$\mathcal{N} = (V, F)$, where $V = \{v_1, ..., v_n\}$ is a set of nodes and $F = \{f_1, ..., f_n\}$ is a set of associated Boolean functions.

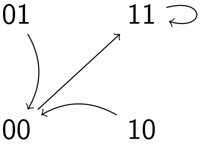
At time t, node $v_i \in V$ can update its state by $s_{t+1}(v_i) = f_i(s_t)$.

$$\begin{cases} f_1 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \\ f_2 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \end{cases}$$

Boolean network



ATG($\mathcal{N}$)



STG($\mathcal{N}$)

Update schemes:

- Fully asynchronous: only one node is non-deterministically selected to update at each time step.
- Synchronous: all nodes are selected to update at each time step.
- ...

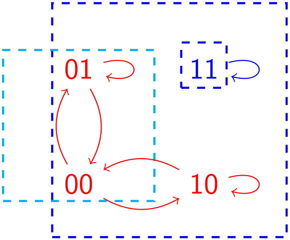
Attractors

A *trap set* is a non-empty set of states from which the system cannot escape once entered.

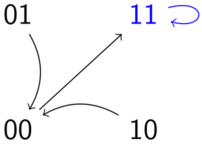
An *attractor* is a minimal trap set. Dependent of update schemes.

$$\begin{cases} f_1 = (v_1 \land v_2) \lor (\neg v_1 \land \neg v_2) \\ f_2 = (v_1 \land v_2) \lor (\neg v_1 \land \neg v_2) \end{cases}$$

Boolean network



ATG($\mathcal{N}$)



STG($\mathcal{N}$)

Two main types of attractors:

- fixed points
- cyclic attractors

Application

Many applications in systems biology, since attractors correspond to biological *phenotypes*:

- new insights into the origins of diseases: cancers, SARS-CoV-2, HIV
- aid the development of new drugs
- starting point for many control approaches for biological systems, which play an important role in systems medicine

Applications in many other fields:

- computer science
- mathematics
- theoretical physics
- complex systems
- ...

New focus

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

$\Longrightarrow$ 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].

$\Longrightarrow$ Separated from attractors, this concept itself goes further to play a crucial role in Boolean network analysis and control [Fontanals et al., 2020, Paulevé et al., 2020, Rozum et al., 2021].

Trap spaces

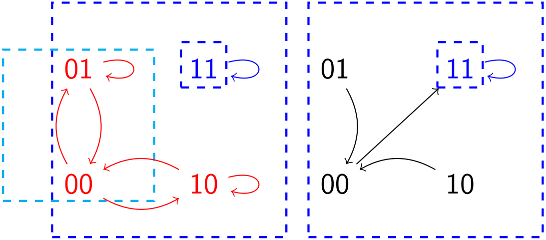
A *subspace* is a mapping $m : V \mapsto \{0, 1, \star\}$. Refer to as a set of states.

- $0\star \sim \{00, 01\}$
- $11 \sim \{11\}$
- $\star\star \sim \{00, 01, 10, 11\}$

A *trap space* is a subspace that is a trap set.

$$\begin{cases} f_1 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \\ f_2 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \end{cases}$$

Boolean network

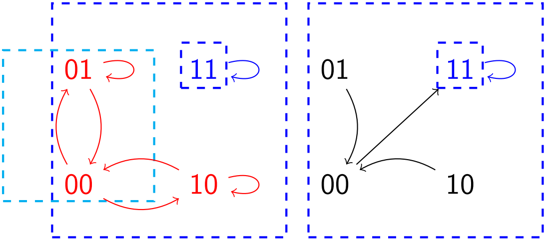


ATG($\mathcal{N}$) STG($\mathcal{N}$)

Trap spaces

$$\begin{cases} f_1 = (v_1 \land v_2) \lor (\neg v_1 \land \neg v_2) \\ f_2 = (v_1 \land v_2) \lor (\neg v_1 \land \neg v_2) \end{cases}$$

Boolean network



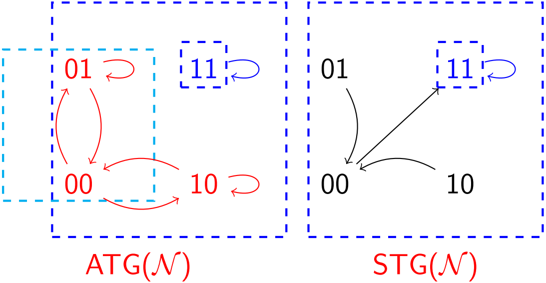
ATG($\mathcal{N}$) STG($\mathcal{N}$)

Not like attractors, trap spaces are independent of the employed update scheme [Klarner et al., 2015].

Trap spaces

$$\begin{cases} f_1 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \\ f_2 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \end{cases}$$

Boolean network



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

A minimal trap space contains at least one attractor $\Longrightarrow$ approximation of attractors regardless of the employed update scheme.

Limitations

The trap space enumeration problem has attracted researchers from various communities and many methods have been proposed.

With the constant increase in model size and complexity of Boolean update functions, the existing methods show their limitations.

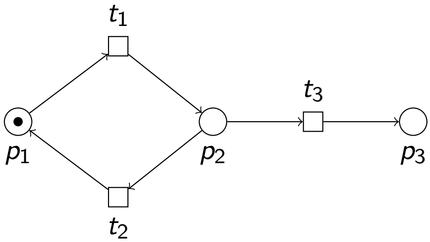
State-of-the-art	Bottleneck	Remark
[Klarner et al., 2015]	prime implicants	hard to obtain + large number
[Paulevé et al., 2020]	DNF + locally-monotonic	sometimes hard to obtain + not handle general models

- locally-monotonic: $x \wedge y$, $v_1 \vee (v_2 \wedge \neg v_3)$
- non-locally-monotonic: $(v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2)$

Petri nets

Bipartite graph

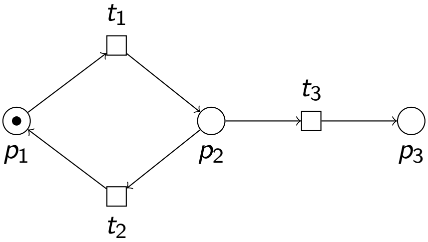
Places P
Transitions T
Weighted arcs W



Petri nets

Bipartite graph

Places P
Transitions T
Weighted arcs W

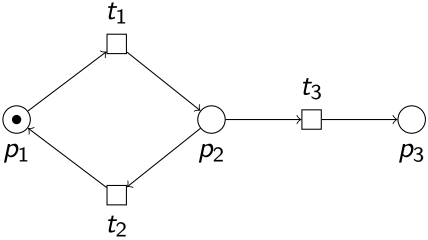


Petri nets have been widely applied to modeling and analysis of biological systems [Blätke et al., 2015]. $\Longrightarrow$ some preliminary but not deep connections between Petri nets and Boolean networks [Chaouiya et al., 2004].

Petri nets

Bipartite graph

Places P
Transitions T
Weighted arcs W



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

Siphons are an important concept in the Petri net research, but they have not been used a lot for the study of biochemical systems.

Contribution

In this work, we make for the first time a connection between trap spaces of Boolean networks and siphons of Petri nets.

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 enumerate 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).

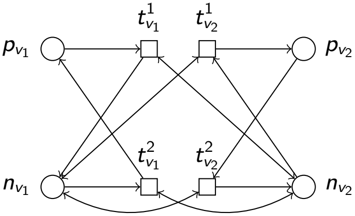
Petri net of a Boolean network

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

Two places for each node: $v \rightsquigarrow p_v, n_v$

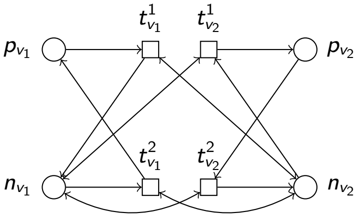
Solutions of $f_v \not\leftrightarrow v \rightsquigarrow$ transitions from p_v to n_v (and back)

$$\begin{cases} f_1 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \\ f_2 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \end{cases}$$



Conflict-free siphons

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



Siphon	Conflict-free?
$\emptyset$	yes
$\{n_{v_1}, n_{v_2}\}$	yes
$\{p_{v_1}, \overline{p_{v_1}}\}$	no
$\{p_{v_2}, \overline{p_{v_2}}\}$	no

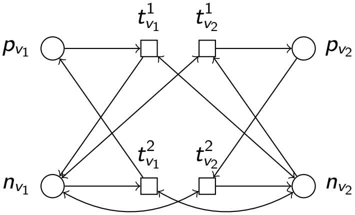
A conflict-free siphon is *maximal* if it is not a subset of any other conflict-free siphon.

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 $\mathcal{P}$.

$$\begin{cases} f_1 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \\ f_2 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \end{cases}$$



Trap space	Conflict-free siphon
$\star\star$	$\emptyset$
11	$\{n_{v_1}, n_{v_2}\}$

Once a siphon is unmarked, it remains unmarked $\sim$ closedness of trap spaces

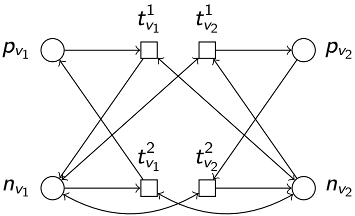
Conflict-free siphons are trap spaces

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 $\mathcal{P}$.

$$\begin{cases} f_1 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \\ f_2 = (v_1 \wedge v_2) \vee (\neg v_1 \wedge \neg v_2) \end{cases}$$

Trap space	Conflict-free siphon
$\star\star$	$\emptyset$
11	$\{n_{v_1}, n_{v_2}\}$



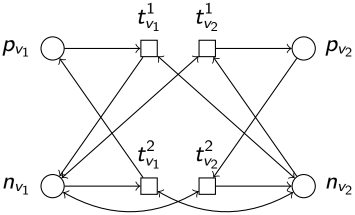
The proof of Theorem 1 can give another way to prove the independence of trap spaces on the update scheme.

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 $\mathcal{P}$.

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



Trap space	Conflict-free siphon
$\star\star$	$\emptyset$
11	$\{n_{v_1}, n_{v_2}\}$

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 $m_1 \cap m_2 = \emptyset$.

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.

We here emphasize 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.

$$(p_v \notin S) \vee (n_v \notin S)$$

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).

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., 2017a], 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^*

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

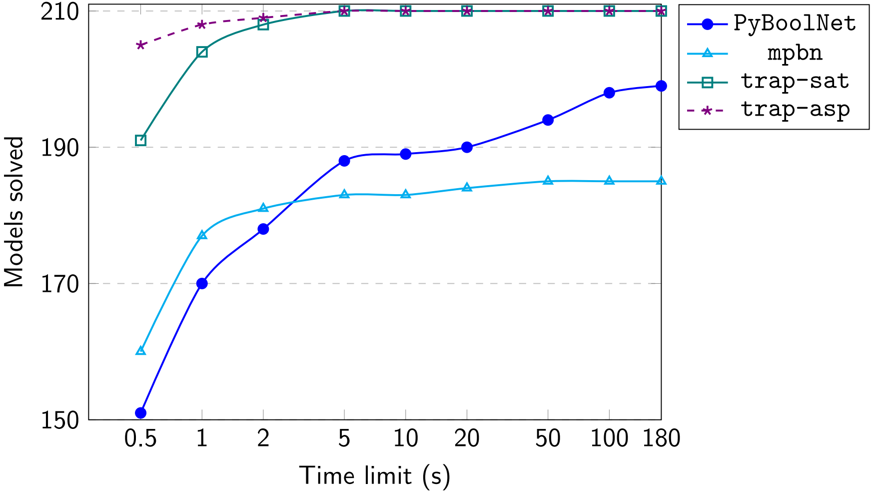
trap-cp and trap-ilp gave poor performance compared to trap-sat and trap-asp.

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

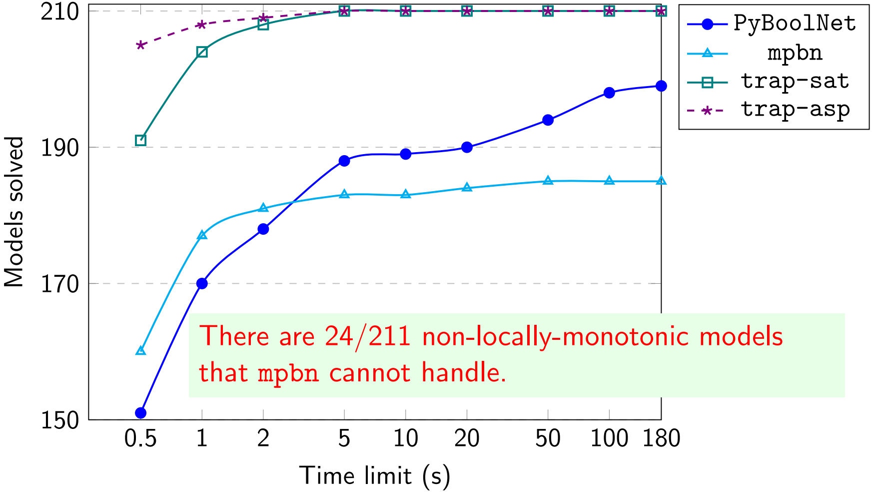
Tested on 211 real-worlds obtained from the BBM repository[3].



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

Results on real-world models

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

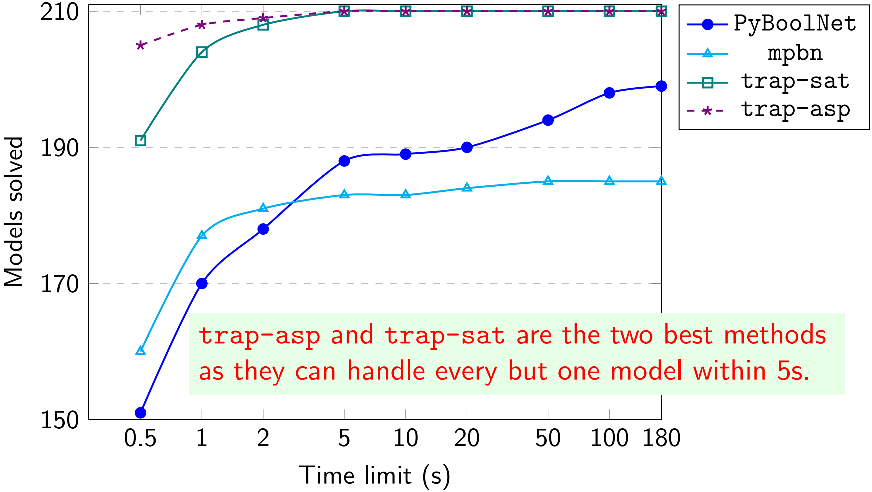


There are 24/211 non-locally-monotonic models that `mpbn` cannot handle.

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

Results on real-world models

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

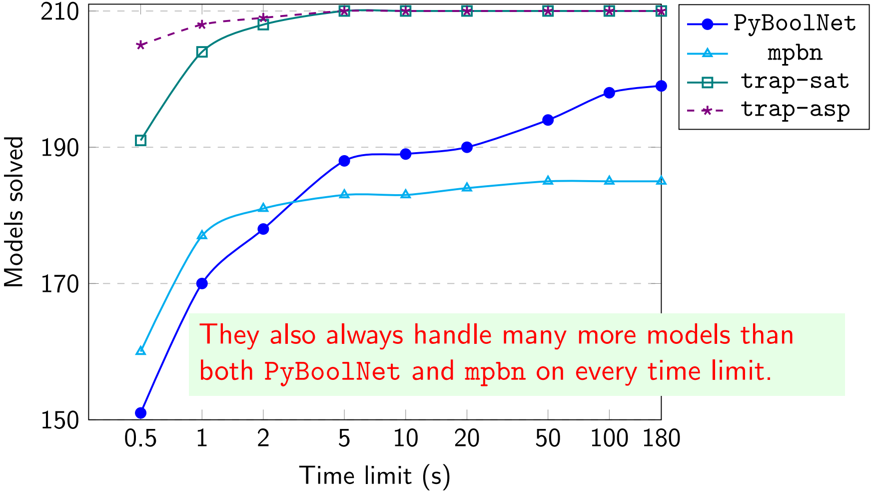


trap-asp and trap-sat are the two best methods as they can handle every but one model within 5s.

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

Results on real-world models

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

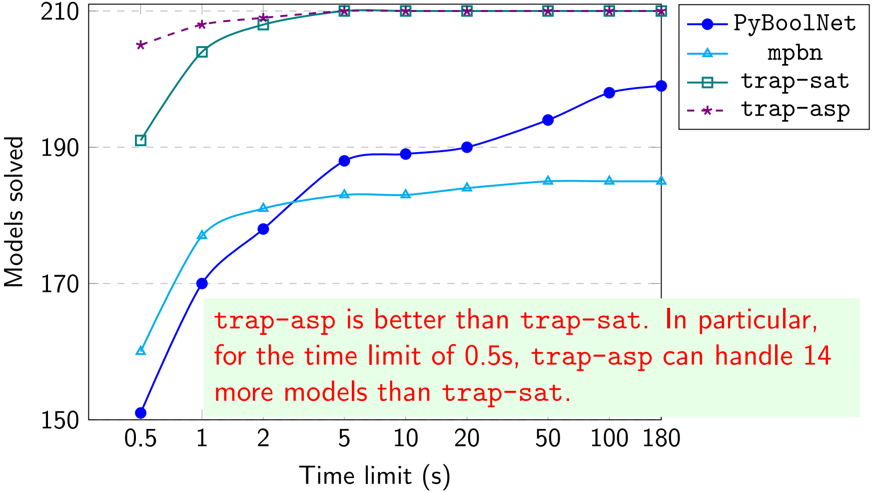


They also always handle many more models than both PyBoolNet and mpbn on every time limit.

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

Results on real-world models

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

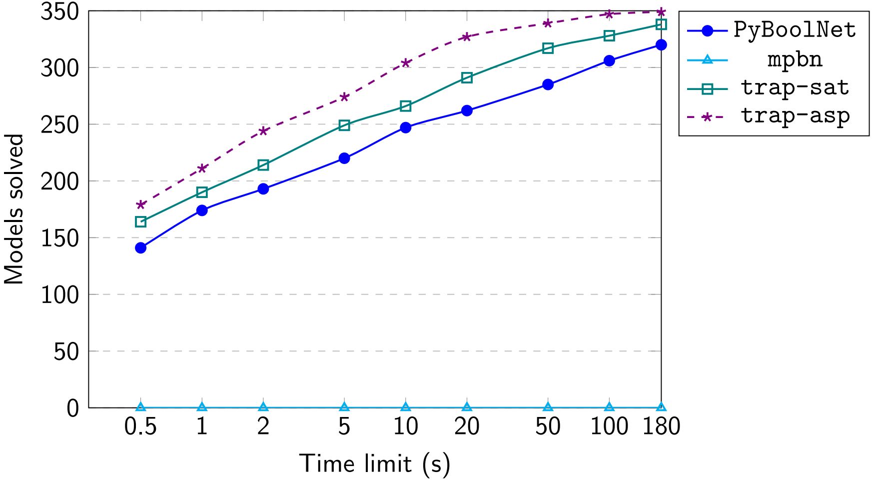


trap-asp is better than trap-sat. In particular, for the time limit of 0.5s, trap-asp can handle 14 more models than trap-sat.

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

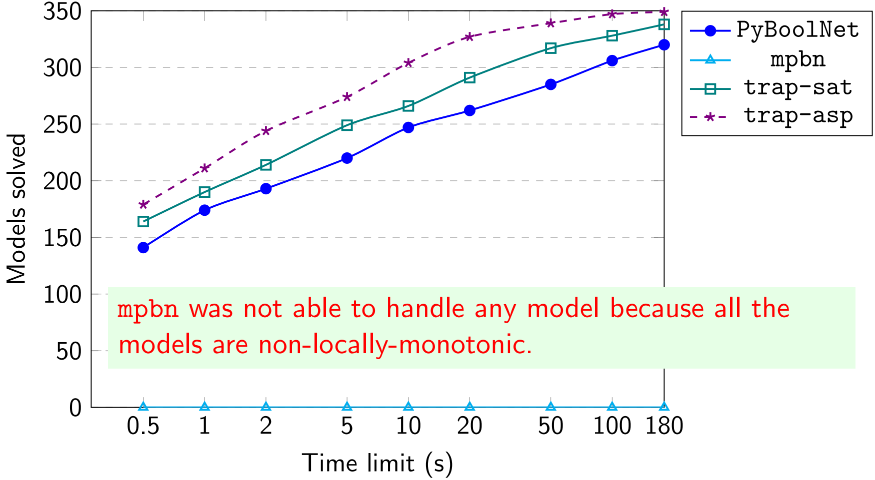
Results on randomly generated models

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



Results on randomly generated models

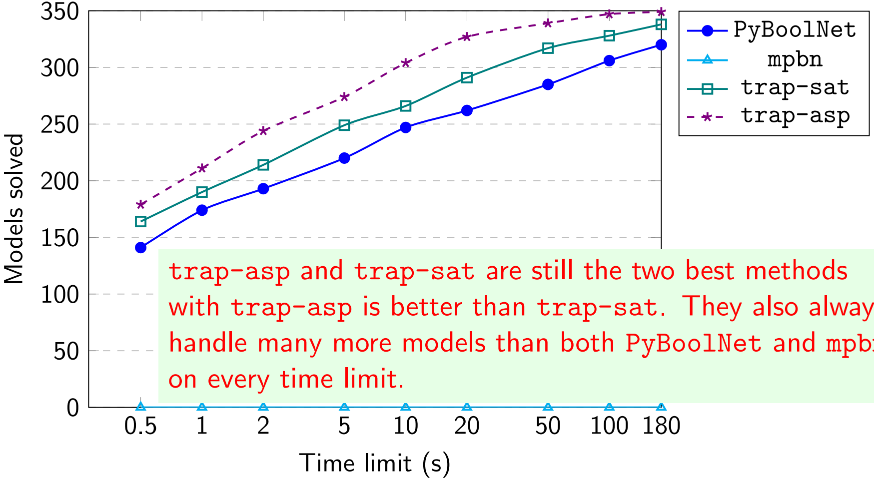
Tested on 350 random models generated by using the `BoolNet R` package [Müssel et al., 2010].



`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.

Future work

Limitations of the siphon-based approach.

- Petri net construction is sometimes expensive, even intractable (e.g., with complex Boolean functions)
- Too many transitions in the Petri net encoding $\rightarrow$ large problem encoding (SAT/ASP/CP/ILP) $\rightarrow$ long solving time, high memory consumption

$\Longrightarrow$ More efficient method for trap space enumeration in Boolean networks. Ongoing work with Samuel Pastva[4], Sylvain Soliman[5], and Belaid Benhamou[6].

[4]Institute of Science and Technology Austria

[5]Lifeware team, Inria Saclay center, Palaiseau, France

[6]LIRICA team, LIS, Aix-Marseille University, Marseille, France

Future work

Reduction techniques in terms of attractors are useful and have been well studied.

$\Longrightarrow$ Use the connection between trap spaces in Boolean networks and siphons in Petri nets to study reductions of Boolean networks in terms of trap spaces.

$\triangle$ $\mathcal{N}' = \mathcal{N} -$ some nodes. What are relations between trap spaces of $\mathcal{N}'$ and $\mathcal{N}$?

Future work

Node $v_i \in V$ is called a *source* node if and only if $f_i = v_i$.

The number of minimal trap spaces $\geq 2^k$ where k is the number of source nodes.

One answer set = one minimal trap space $\rightarrow$ long computational time, high memory consumption

Future work

Boolean network models of biological systems usually contain many source nodes, which might be hard to avoid in the modeling process [Aghamiri et al., 2020].

However, systems biologists usually do not want to obtain many solutions (i.e., minimal trap spaces), less is more preferred.

⟹ Need a new method to overcome this problem. This method may return a symbolic representation of the set of minimal trap spaces, which is useful for further biologically meaningful analysis (in collaboration with Domenico Sgariglia[4] on Boolean modeling of breast cancer).

[4] Engenharia de Sistemas e Computação, COPPE-UFRJ, Rio de Janeiro, Brazil

Future work

In logical modeling, having only two levels of activation is sometimes not enough to fully capture the dynamics of a real-world biological systems.

$\Longrightarrow$ Need for multi-valued networks

$\Longrightarrow$ We define the concept of trap spaces for multi-valued networks, prove several properties, and propose a siphon-based approach to enumerate trap spaces[4].

$\Longrightarrow$ More efficient method for trap space enumeration in multi-valued networks. Ongoing work with Samuel Pastva[5], Sylvain Soliman[6], and Belaid Benhamou[7].

[4] Trinh, V.-G., Benhamou, B., Henzinger, T., & Pastva, S. (2023). Trap spaces of multi-valued networks: Definition, computation, and applications. ISMB/ECCB 2023.

[5] Institute of Science and Technology Austria

[6] Lifeware team, Inria Saclay center, Palaiseau, France

[7] LIRICA team, LIS, Aix-Marseille University, Marseille, France

Future work

It is interesting and helpful to define the notion of trap spaces in Petri nets.

⟹ Study in depth properties of trap spaces in Petri nets.

⟹ Establish the link between siphons of a Petri net and its trap spaces or attractors.

⟹ Propose efficient enumeration method.

⟹ Application to analysis and control of Petri nets. In collaboration with Professor Koichi Kobayashi[4]

[4] Hokkaido University, Sapporo, Hokkaido, Japan

Future work

Continuing with Petri nets, it would be also interesting to develop efficient methods for computing some static properties such as minimal P-semiflows [Soliman, 2012] and siphons [Nabli et al., 2016].

Existing methods work not very well, especially for dense networks.

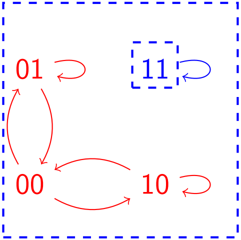
Can be useful for studying Chemical Reaction Networks[4].

[4] Degrand, Elisabeth, François Fages, and Sylvain Soliman. "Graphical conditions for rate independence in chemical reaction networks." In International Conference on Computational Methods in Systems Biology, pp. 61-78. Cham: Springer International Publishing, 2020.

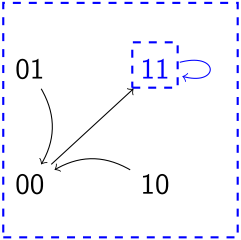
Future work

$$\begin{cases} f_1 = (v_1 \land v_2) \lor (\neg v_1 \land \neg v_2) \\ f_2 = (v_1 \land v_2) \lor (\neg v_1 \land \neg v_2) \end{cases}$$

Boolean network



ATG($\mathcal{N}$)



STG($\mathcal{N}$)

Back to attractor enumeration: more difficult in general and dependent of the employed update scheme.

Boolean networks ⇒ Multi-valued networks ⇒ Petri nets

Future work

Fully asynchronous update: existing methods [Klarner et al., 2017b, Abdallah et al., 2017, Mizera et al., 2019, Giang et al., 2022, Benes et al., 2021, Rozum et al., 2022, Trinh et al., 2022] have their own bottlenecks.

Synchronous update: existing methods [Zhang et al., 2007, Dubrova and Teslenko, 2011, Zheng et al., 2013, Yuan et al., 2019, Mori and Akutsu, 2022] have their own bottlenecks.

A real challenge! $\Longrightarrow$ Ongoing work with Jordan Rozum[4], Samuel Pastva[5], and Kyu Hyong Park[6].

[4] Binghamton University, United States
[5] Institute of Science and Technology Austria
[6] Pennsylvania State University, United States

Thank you for your attention!

References I

Abdallah, E. B., Folschette, M., Roux, O. F., and Magnin, M. (2017).
ASP-based method for the enumeration of attractors in non-deterministic synchronous and asynchronous multi-valued networks.
Algorithms Mol. Biol., 12(1):20:1–20:23.

Aghamiri, S. S., Singh, V., Naldi, A., Helikar, T., Soliman, S., Niarakis, A., and Xu, J. (2020).
Automated inference of Boolean models from molecular interaction maps using CaSQ.
Bioinform., 36(16):4473–4482.

References II

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.

Blätke, M. A., Heiner, M., and Marwan, W. (2015).
Biomodel engineering with Petri nets.
In *Algebraic and Discrete Mathematical Methods for Modern Biology*, pages 141–192. Elsevier.

References III

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, 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 IV

Dubrova, E. and Teslenko, M. (2011).
A sat-based algorithm for finding attractors in synchronous boolean networks.
IEEE/ACM transactions on computational biology and bioinformatics, 8(5):1393–1399.

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 V

Giang, T. V., Akutsu, T., and Hiraishi, K. (2022).
An FVS-based approach to attractor detection in asynchronous random Boolean networks.
IEEE ACM Trans. Comput. Biol. Bioinform., 19(2):806–818.

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. (2017a).
PyBoolNet: a python package for the generation, analysis and visualization of Boolean networks.
Bioinform., 33(5):770–772.

References VI

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

Mori, T. and Akutsu, T. (2022).
Attractor detection and enumeration algorithms for Boolean networks.
Comput. Struct. Biotechnol. J., 20:2512–2520.

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 VII

Nabli, F., Martinez, T., Fages, F., and Soliman, S. (2016).
On enumerating minimal siphons in petri nets using clp and sat solvers: theoretical and practical complexity.
Constraints, 21:251–276.

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., Deritei, D., Park, K. H., Gómez Tejeda Zañudo, J., and Albert, R. (2022).
pystablemotifs: Python library for attractor identification and control in boolean networks.
Bioinformatics, 38(5):1465–1466.

References VIII

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.

Soliman, S. (2012).
Invariants and other structural properties of biochemical models as a constraint satisfaction problem.
Algorithms for Molecular Biology, 7(1):1–9.

Trinh, V., Hiraishi, K., and Benhamou, B. (2022).
Computing attractors of large-scale asynchronous Boolean networks using minimal trap spaces.
In *ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 13:1–13:10. ACM.

References IX

Yuan, Q., Mizera, A., Pang, J., and Qu, H. (2019).
A new decomposition-based method for detecting attractors in synchronous boolean networks.
Science of Computer Programming, 180:18–35.

Zhang, S.-Q., Hayashida, M., Akutsu, T., Ching, W.-K., and Ng, M. K. (2007).
Algorithms for finding small attractors in boolean networks.
EURASIP Journal on Bioinformatics and Systems Biology, 2007:1–13.

Zheng, D., Yang, G., Li, X., Wang, Z., Liu, F., and He, L. (2013).
An efficient algorithm for computing attractors of synchronous and asynchronous boolean networks.
PloS one, 8(4):e60593.