Minimal trap spaces of Boolean models are maximal siphons of their Petri net encoding

Van-Giang Trinh[1], Belaid Benhamou[1], Kunihiko Hiraishi[2] and Sylvain Soliman[3]

[1]LIS, Aix-Marseille University, Marseille, France
[2]School of Information Science, Japan Advanced Institute of Science and Technology, Japan
[3]Lifeware team, Inria Saclay center, Palaiseau, France

September 14, 2022

Boolean modeling

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

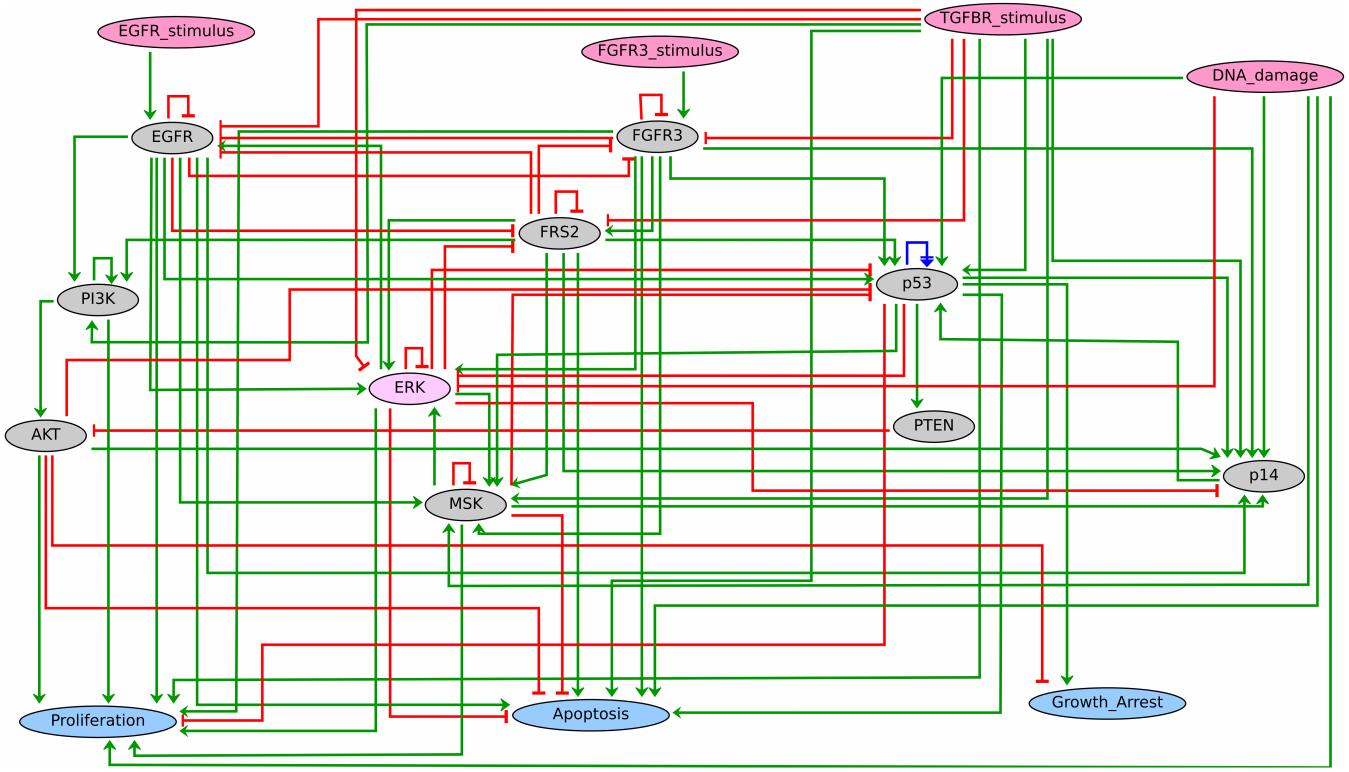


Figure: Boolean model of the MAPK regulatory network, whose involvement in bladder cancer is well established [Grieco et al., 2013].

Boolean models

Boolean model

A Boolean model $\mathcal{M}$ 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 $f_i : \mathbb{B}^{|IN(f_i)|} \to \mathbb{B}$, where $IN(f_i)$ is the set of input nodes of f_i.

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

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

Dynamics of Boolean models

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.

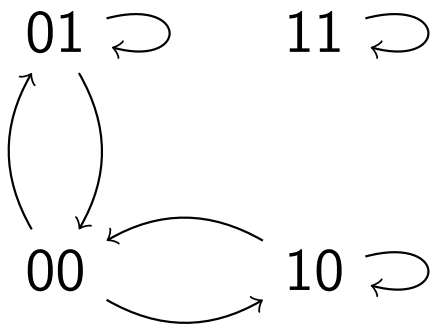
Based on the update scheme, the Boolean model can transit from a state to another state (possibly identical). This is the *state transition* (denoted by $\longrightarrow$).

The dynamics of a Boolean model 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 model

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

Boolean model



State transition graph

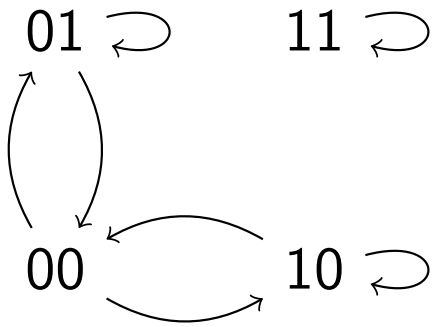
Fully asynchronous update scheme: only one node is nondeterministically selected in order to be updated at each time step.

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 model 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
$\{00, 01\}$	no	no
$\{00, 01, 10\}$	yes	yes
$\{00, 01, 10, 11\}$	yes	no

Application

Besides simulation, the analysis of Boolean 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 models 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 as there can be a huge number of implicants.

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 models.

Contribution

In this work, we

- make a connection between trap spaces of Boolean models and siphons of Petri nets, which has not yet been explored before;
- and then propose a novel method to compute minimal trap spaces of a Boolean model.

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

Subspaces

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

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.

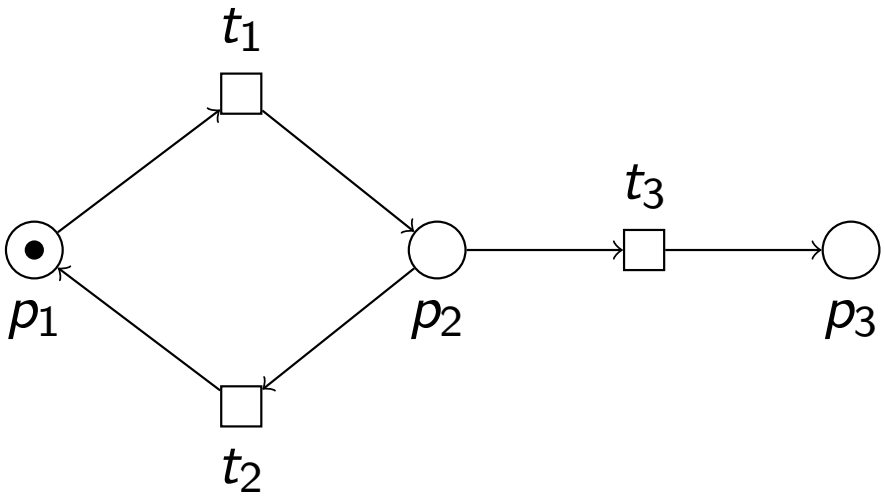
State set	Trap set?	Subspace?	Trap space?
$\{11\}$	yes	11	yes
$\{00, 01\}$	no	$0\star$	no
$\{00, 01, 10\}$	yes	no	no
$\{00, 01, 10, 11\}$	yes	$\star\star$	yes

Note that trap spaces of a Boolean model 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.

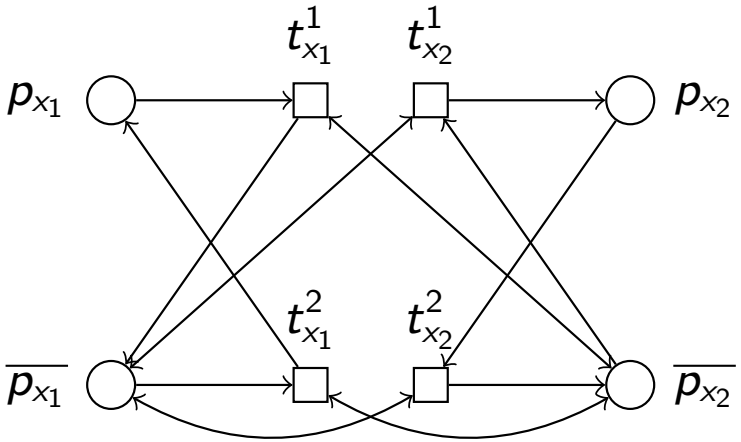
Petri net of a Boolean model

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

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

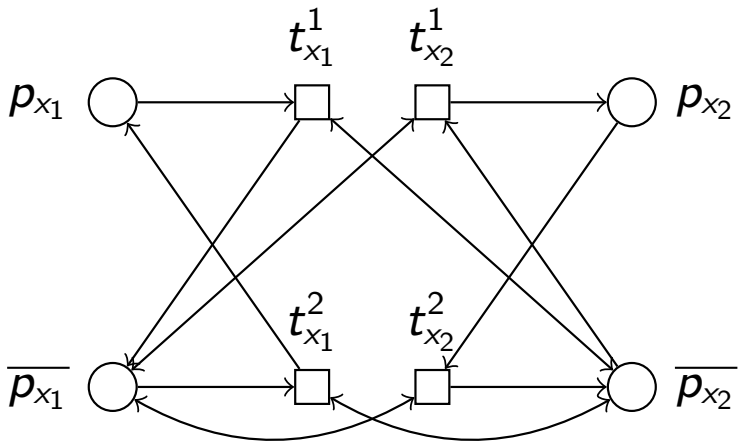
Solutions of $f_v \not\leftrightarrow v \rightsquigarrow$ transitions from p_v to $\overline{p_v}$ (and back)

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



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

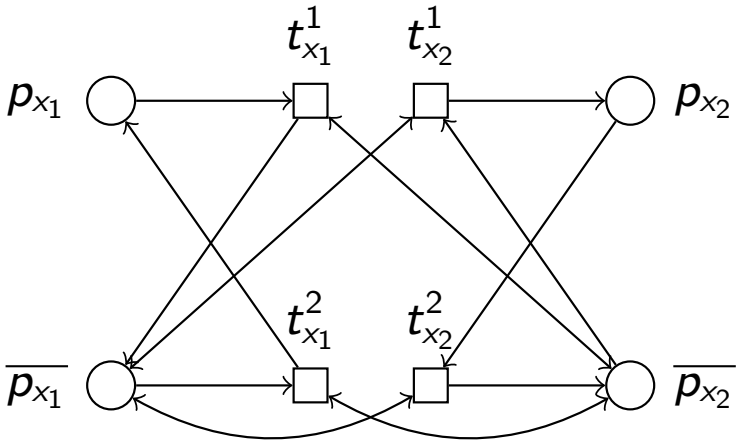
A conflict-free siphon is *maximal* if it is not a subset of any other conflict-free siphon.

Conflict-free siphons are trap spaces

Theorem 1

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

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



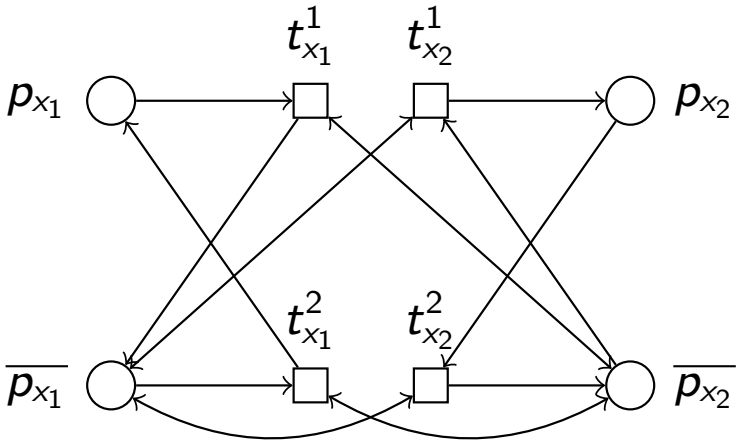
Trap space	Conflict-free siphon
$\star\star$	$\emptyset$
11	$\{\overline{p_{x_1}}, \overline{p_{x_2}}\}$

Maximal conflict-free siphons are minimal trap spaces

Theorem 2

Let $\mathcal{M}$ be a Boolean model and $\mathcal{P}$ be its Petri net encoding. There is a one-to-one correspondence between the set of **minimal trap spaces** of $\mathcal{M}$ 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	$\{\overline{p_{x_1}}, \overline{p_{x_2}}\}$

Proposed method for minimal trap space computation

From Theorem 2, we propose a new method for computing minimal trap spaces of a Boolean model $\mathcal{M}$.

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

Petri net transformation

Transforming a Boolean model 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 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 method. The implementation uses BDDs[1].

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

Maximal conflict-free siphon computation

Characterize all siphons of the encoded Petri net as a system of Boolean rules.

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

Add to the system the Boolean rules representing the conflict-freeness.

$$p_v \in S \Rightarrow \overline{p_v} \notin S \wedge \overline{p_v} \in S \Rightarrow p_v \notin S, v \in V$$

Encode the system as an ASP.

Use an ASP solver (e.g., clingo [Gebser et al., 2011]) to compute all set-inclusion maximal answer sets of the ASP.

Set-maximality through "heuristics"

```
clingo --heuristic=Domain --enum-mod=domRec --dom-mod=3
```

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 model 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 method	general

mpbn is specifically designed for exploiting the locally-monotonicity [Paulevé et al., 2020]. Hence, for locally-monotonic Boolean models, it has many advantages over other methods.

PyBoolNet repo, 1000 first solutions

our proposed method

model	n	$\|M\|$	PyBoolNet	mpbn	Trappist
arellano_rootstem	9	4	0.05	0.01	0.02
calzone_cellfate	28	27	0.03	0.02	0.03
dahlhaus_neuroplastoma	23	32	0.06	0.02	0.03
		...			
jaoude_thdiff	103	>1000	**1.92**	**1.32**	0.20
klamt_tcr	40	8	0.04	0.01	0.04
		...			
n5s3	4	3	0.03	**NM**	0.02
		...			
randomnet_n15k3	15	3	0.04	**NM**	0.04
randomnet_n7k3	7	10	0.03	**NM**	0.02
remy_tumorigenesis	34	25	**2.05**	0.04	0.06
saadatpour_guardcell	13	1	0.02	0.00	0.02
selvaggio_emt	56	>1000	**1.09**	**0.76**	0.18
tournier_apoptosis	12	3	0.03	0.01	0.02
xiao_wnt5a	7	4	0.03	0.00	0.02
zhang_tlgl	60	156	**0.22**	**0.22**	0.09
zhang_tlgl_v2	60	258	0.11	**0.24**	0.08

PyBoolNet repo, 1000 first solutions

model	n	$\|M\|$	PyBoolNet	mpbn	Trappist
arellano_rootstem	9	4	0.05	0.01	0.02
calzone_cellfate		7	0.03	0.02	0.03
dahlhaus_neuroplastoma	23				0.03
		...			
jaoude_thdiff	103	>1000	**1.92**	**1.32**	0.20
klamt_tcr	40	8	0.04	0.01	0.04
		...			
n5s3			0.03	**NM**	0.02
		...			
randomnet_n15k3	15	3	0.04	**NM**	0.04
randomnet_r			0.03	**NM**	0.02
remy_tumori			**2.05**	0.04	0.06
saadatpour_			0.02	0.00	0.02
selvaggio_emt	56	>1000	**1.09**	**0.76**	0.18
tournier_apoptosis	12	3	0.03	0.01	0.02
xiao_wnt5a	7	4	0.03	0.00	0.02
zhang_tlgl	60	156	**0.22**	**0.22**	0.09
zhang_tlgl_v2	60	258	0.11	**0.24**	0.08

number of nodes

number of minimal trap spaces

non-monotonic model

significant difference to the best running time (in seconds)

PyBoolNet repo, 1000 first solutions

model	n	$\|M\|$	PyBoolNet	mpbn	Trappist
arellano_rootstem	9	4	0.05	0.01	0.02
calzone_cellfate	28	27	0.03	0.02	0.03
dahlhaus_neuroplastoma	23	32	0.06	0.02	0.03
		...			
jaoude_thdiff	103	>1000	**1.92**	**1.32**	0.20
klamt_tcr	40	8	0.04	0.01	0.04
		...			
n					2
randomnet_n15k3	15	3	0.04	**NM**	0.04
randomnet_n7k3	7	10	0.03	**NM**	0.02
remy_tumorigenesis	34	25	**2.05**	0.04	0.06
saadatpour_guardcell	13	1	0.02	0.00	0.02
selvaggio_emt	56	>1000	**1.09**	**0.76**	0.18
tournier_apoptosis	12	3	0.03	0.01	0.02
xiao_wnt5a	7	4	0.03	0.00	0.02
zhang_tlgl	60	156	**0.22**	**0.22**	0.09
zhang_tlgl_v2	60	258	0.11	**0.24**	0.08

All three methods are comparable with all minimal trap spaces found very fast because the models are quite small.

PyBoolNet repo, 1000 first solutions

model	n	$\lvert M \rvert$	PyBoolNet	mpbn	Trappist
arellano_rootstem	9	4	0.05	0.01	0.02
calzone_cellfate	28	27	0.03	0.02	0.03
dahlhaus_neuroplastoma	23	32	0.06	0.02	0.03
		...			
jaoude_thdiff	103	>1000	**1.02**	**1.32**	0.20
k					4
n5s3	4	3	0.03	**NM**	0.02
		...			
randomnet_n15k3	15	3	0.04	**NM**	0.04
randomnet_n7k3	7	10	0.03	**NM**	0.02
remy_tumorigenesis	34	25	**2.05**	0.04	0.06
saadatpour_guardcell	13	1	0.02	0.00	0.02
selvaggio_emt	56	>1000	**1.09**	**0.76**	0.18
tournier_apoptosis	12	3	0.03	0.01	0.02
xiao_wnt5a	7	4	0.03	0.00	0.02
zhang_tlgl	60	156	**0.22**	**0.22**	0.09
zhang_tlgl_v2	60	258	0.11	**0.24**	0.08

For 3 of the 29 models, mpbn did not give any answer because these models are non-locally-monotonic.

PyBoolNet repo, 1000 first solutions

model	n	$\|M\|$	PyBoolNet	mpbn	Trappist
arellano_rootstem	9	4	0.05	0.01	0.02
calzone_cellfate	28	27	0.03	0.02	0.03
dahlhaus_neuroplastoma	23	32	0.06	0.02	0.03
		...			
jaoude_thdiff	103	>1000	**1.92**	**1.32**	0.20
klamt_tcr	40	8	0.04	0.01	0.04
n					2
		...			
randomnet_n15k3	15	3	0.04	**NM**	0.04
randomnet_n7k3	7	10	0.03	**NM**	0.02
remy_tumorigenesis	34	25	**2.05**	0.04	0.06
saadatpour_guardcell	13	1	0.02	0.00	0.02
selvaggio_emt	56	>1000	**1.09**	**0.76**	0.18
tournier_apoptosis	12	3	0.03	0.01	0.02
xiao_wnt5a	7	4	0.03	0.00	0.02
zhang_tlgl	60	156	**0.22**	**0.22**	0.09
zhang_tlgl_v2	60	258	0.11	**0.24**	0.08

On every model that was a bit challenging for PyBoolNet or mpbn, the new method is more efficient with significant speedups.

Large models from literature, 2 min timeout

model	n	$\|M\|$	PyBoolNet	mpbn	Trappist
inflammatory-bowel	47	1	**DNF**	**NM**	1.71
T-LGL-survival	61	318	0.23	0.30	0.10
butanol-production	66	>1000	0.20	**0.88**	0.12
colon-cancer	70	10	**0.07**	**0.04**	0.06
mast-cell-activation	73	>1000	**0.16**	**0.89**	0.13
IL-6-signalling	86	>1000	0.17	**1.01**	0.13
Corral-ThIL-17-diff	92	>1000	**DNF**	**1.18**	0.18
Korkut-2015	99	>1000	**DNF**	1.36	0.39
adhesion-cip-migration	121	78	**28.77**	0.34	0.25
interferon-1	121	>1000	**2.82**	**1.42**	0.17
TCR-TLR5-signaling	130	48	**0.54**	0.18	0.13
influenza-replication	131	>1000	**8.75**	**1.54**	0.20
prostate-cancer	133	>1000	**DNF**	**2.71**	0.38
HIV-1	138	>1000	**DNF**	**11.77**	0.36
fibroblasts	139	>1000	**DNF**	**NM**	0.42
HMOX-1-pathway	145	>1000	**1.65**	**1.98**	0.20

Large models from literature, 2 min timeout (cont.)

model	n	$\|M\|$	PyBoolNet	mpbn	Trappist
		...			
MAPK	181	>1000	**86.81**	**2.64**	0.30
er-stress	182	>1000	**11.13**	**2.26**	0.24
cascade-3	183	1	**DNF**	0.33	0.24
CHO-2016	200	>1000	**DNF**	**3.36**	0.36
T-cell-check-point	218	>1000	**28.83**	**NM**	0.38
ErbB-receptor-signaling	247	>1000	**DNF**	**NM**	1.06
macrophage-activation	321	>1000	**10.47**	**3.86**	0.50
cholocystokinin	383	>1000	**2.83**	**4.46**	0.49
Alzheimer	762	>1000	**DNF**	**NM**	0.99
KEGG-network	1659	>1000	**DNF**	21.57	30.22
human-network	1953	>1000	**DNF**	25.19	21.91
SN-5	2746	>1000	**DNF**	37.54	45.57
turei-2016	4691	>1000	**DNF**	119.98	**DNF**

Large models from literature, 2 min timeout (cont.)

model	n	$\|M\|$	PyBoolNet	mpbn	Trappist
		...			
MAPK	181	>1000	**86.81**	**2.64**	0.30
er-stress	182	>1000	**11.13**	**2.26**	0.24
cascade-3	183	1	**DNF**	0.33	0.24
CHO-2016	200	>1000	**DNF**	**3.36**	0.36
T-cell-check-point	218	>1000	**28.83**	**NM**	0.38
These models are quite big (in size), complex (i.e., having high average in-degree) and most of them have never been fully analyzed.					
cholocystokinin	383	>1000	**2.83**	**4.46**	0.49
Alzheimer	762	>1000	**DNF**	**NM**	0.99
KEGG-network	1659	>1000	**DNF**	21.57	30.22
human-network	1953	>1000	**DNF**	25.19	21.91
SN-5	2746	>1000	**DNF**	37.54	45.57
turei-2016	4691	>1000	**DNF**	119.98	**DNF**

Large models from literature, 2 min timeout (cont.)

model	n	$\|M\|$	PyBoolNet	mpbn	Trappist
		...			
MAPK	181	>1000	**86.81**	**2.64**	0.30
er-stress	182	>1000	**11.13**	**2.26**	0.24
cascade-3	183	1	**DNF**	0.33	0.24
CHO-2016	200	>1000	**DNF**	**3.36**	0.36
T-cell-check-point	218	>1000	**28.83**	**NM**	0.38
For 6 of the 33 models, mpbn did not give any answer because these models are non-locally-monotonic.					
cholocystokinin	383	>1000	**2.83**	**4.46**	0.49
Alzheimer	762	>1000	**DNF**	**NM**	0.99
KEGG-network	1659	>1000	**DNF**	21.57	30.22
human-network	1953	>1000	**DNF**	25.19	21.91
SN-5	2746	>1000	**DNF**	37.54	45.57
turei-2016	4691	>1000	**DNF**	119.98	**DNF**

Large models from literature, 2 min timeout (cont.)

model	n	$\|M\|$	PyBoolNet	mpbn	Trappist
		...			
MAPK	181	>1000	**86.81**	**2.64**	0.30
er-stress	182	>1000	**11.13**	**2.26**	0.24
cascade-3	183	1	**DNF**	0.33	0.24
CHO-2016	200	>1000	**DNF**	**3.36**	0.36
T-cell-check-point	218	>1000	**28.83**	**NM**	0.38
Alzheimer	762	>1000	**DNF**	**NM**	0.99
KEGG-network	1659	>1000	**DNF**	21.57	30.22
human-network	1953	>1000	**DNF**	25.19	21.91
SN-5	2746	>1000	**DNF**	37.54	45.57
turei-2016	4691	>1000	**DNF**	119.98	**DNF**

For 26 of the 33 models where both mpbn and Trappist returned the answers, they are comparable in computation time, though surprisingly mpbn appears a bit slower on average.

Large models from literature, 2 min timeout (cont.)

model	n	$\|M\|$	PyBoolNet	mpbn	Trappist
		...			
MAPK	181	>1000	**86.81**	**2.64**	0.30
er-stress	182	>1000	**11.13**	**2.26**	0.24
cascade-3	183	1	**DNF**	0.33	0.24
CHO-2016	200	>1000	**DNF**	**3.36**	0.36
T-cell-check-point	218	>1000	**28.83**	**NM**	0.38

Note however that mpbn was the only tool to provide a solution for the turei model within two minutes, thus confirming its advantages for locally-monotonic models.

Alzheimer	762	>1000	**DNF**	**NM**	0.99
KEGG-network	1659	>1000	**DNF**	21.57	30.22
human-network	1953	>1000	**DNF**	25.19	21.91
SN-5	2746	>1000	**DNF**	37.54	45.57
turei-2016	4691	>1000	**DNF**	119.98	**DNF**

Large models from literature, 2 min timeout (cont.)

model	n	$\|M\|$	PyBoolNet	mpbn	Trappist
		...			
MAPK	181	>1000	**86.81**	**2.64**	0.30
er-stress	182	>1000	**11.13**	**2.26**	0.24
cascade-3	183	1	**DNF**	0.33	0.24
CHO-2016	200	>1000	**DNF**	**3.36**	0.36
T-cell-check-point	218	>1000	**28.83**	**NM**	0.38
Alzheimer	762	>1000	**DNF**	**NM**	0.99
KEGG-network	1659	>1000	**DNF**	21.57	30.22
human-network	1953	>1000	**DNF**	25.19	21.91
SN-5	2746	>1000	**DNF**	37.54	45.57
turei-2016	4691	>1000	**DNF**	119.98	**DNF**

The proposed method vastly outperforms PyBoolNet in computational time, on each and every model, and sometimes with orders of magnitude of difference.

Conclusion

Minimal trap spaces are important in Boolean model analysis.

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

We proposed a new method for the computation of minimal trap spaces of Boolean models.

The evaluation on large models from the literature shows that

- our method can scale up much better than the state-of-the-art prime-implicants based techniques for non-locally-monotonic models;
- our method is comparable to mpbn for locally-monotonic models.

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

Future work

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

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

Finally, we think that the links between Petri nets and Boolean models that we stumbled upon in this method 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 models.

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

Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., and Schneider, M. (2011).
Potassco: The Potsdam answer set solving collection.
AI Commun., 24(2):107–124.

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.

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.

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.

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).

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.