Automatic Synthesis of Boolean Networks from Biological Knowledge and Data

Athénaïs Vaginay, Taha Boukhobza, Malika Smaïl-Tabbone

ASPOCP — July 31, 2022

ola 2021

Athénaïs Vaginay, Taha Boukhobza, Malika Smaïl-Tabbone.
Automatic Synthesis of Boolean Networks from Biological Knowledge and Data. *International Conference on Optimization and Learning, Jun 2021, online.*

Boolean Network (BN)

set of n **Boolean functions** (one per component)

$$\{f_i : \mathbb{B}^n \to \mathbb{B}, \forall i \in V\}$$

V: set of n **components** (= genes, proteins...)
$\mathbb{B} = \{0/\text{inactive}, 1/\text{active}\}$
configuration: a vector of $\mathbb{B}^n$

Boolean Network — an example

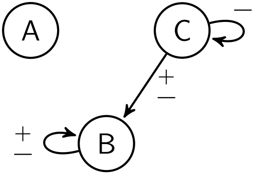
$$\mathscr{B} = \begin{cases} f_{\mathsf{A}} := 0 \\ f_{\mathsf{B}} := (\mathsf{B} \land \neg\mathsf{C}) \lor (\neg\mathsf{B} \land \mathsf{C}) \\ f_{\mathsf{C}} := \neg\mathsf{C} \end{cases}$$

Boolean functions in minimal disjonctive normal form (minDNF)

Boolean Network — an example, its structure

$$\mathscr{B} = \begin{cases} f_{\mathsf{A}} := 0 \\ f_{\mathsf{B}} := (\mathsf{B} \wedge \neg \mathsf{C}) \vee (\neg \mathsf{B} \wedge \mathsf{C}) \\ f_{\mathsf{C}} := \neg \mathsf{C} \end{cases}$$

interaction graph:



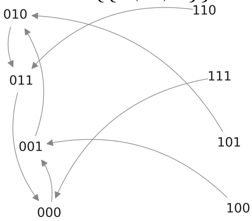
Boolean Network — an example, its dynamics

$$\mathscr{B} = \begin{cases} f_{\mathsf{A}} := 0 \\ f_{\mathsf{B}} := (\mathsf{B} \wedge \neg \mathsf{C}) \vee (\neg \mathsf{B} \wedge \mathsf{C}) \\ f_{\mathsf{C}} := \neg \mathsf{C} \end{cases}$$

state transition graph under an update scheme U:

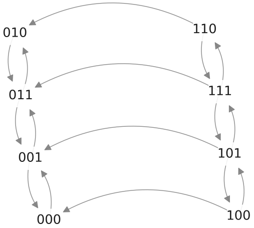
synchronous

$U = \{\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}\}$



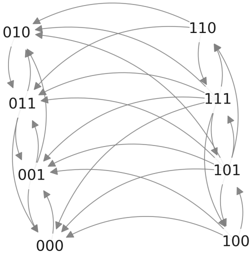
asynchronous

$U = \{\{\mathsf{A}\}, \{\mathsf{B}\}, \{\mathsf{C}\}\}$



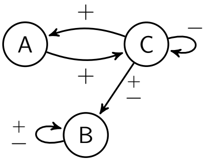
mixed

$U = P(\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}) \setminus \emptyset$



Structural Knowledge and Dynamical Data

structural knowledge: Prior Knowledge Network (PKN)
= putative interactions between the components

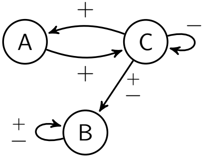


dynamical data: Time Series (TS) = concentrations of the components over time, sequence of configurations

t	1	2	3	4	5	6	7	8	9	10	11	12	13	...
A	0	3	7	13	20	30	49	61	100	63	36	25	2	...
B	100	86	64	57	54	53	51	49	45	37	33	28	22	...
C	0	27	36	42	60	75	54	44	38	48	60	72	88	...

Structural Knowledge and Dynamical Data

structural knowledge: Prior Knowledge Network (PKN)
= putative interactions between the components

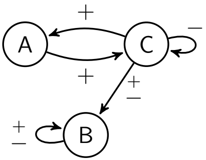


dynamical data: Time Series (TS) = concentrations of the components over time, sequence of configurations

t	1	2	3	4	5	6	7	8	9	10	11	12	13	...
A	0	3	7	13	20	30	49	61	100	63	36	25	2	...
B	100	86	64	57	54	53	51	49	45	37	33	28	22	...
C	0	27	36	42	60	75	54	44	38	48	60	72	88	...

Structural Knowledge and Dynamical Data

structural knowledge: Prior Knowledge Network (PKN)
= putative interactions between the components

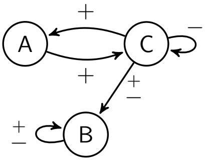


dynamical data: Time Series (TS) = concentrations of the components over time, sequence of configurations

t	1	2	3	4	5	6	7	8	9	10	11	12	13	...
A	0	3	7	13	20	30	49	61	100	63	36	25	2	...
B	100	86	64	57	54	53	51	49	45	37	33	28	22	...
C	0	27	36	42	60	75	54	44	38	48	60	72	88	...

Structural Knowledge and Dynamical Data

structural knowledge: Prior Knowledge Network (PKN)
= putative interactions between the components

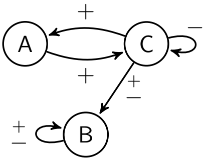


dynamical data: Time Series (TS) = concentrations of the components over time, sequence of configurations

t	1	2	3	4	5	6	7	8	9	10	11	12	13	...
A	0	3	7	13	20	30	49	61	100	63	36	25	2	...
B	100	86	64	57	54	53	51	49	45	37	33	28	22	...
C	0	27	36	42	60	75	54	44	38	48	60	72	88	...

Structural Knowledge and Dynamical Data

structural knowledge: Prior Knowledge Network (PKN)
= putative interactions between the components



dynamical data: Time Series (TS) = concentrations of the components over time, sequence of configurations

	010				→ 011			→ 100			→ 001			
t	1	2	3	4	5	6	7	8	9	10	11	12	13	...
A	0	3	7	13	20	30	49	61	100	63	36	25	2	...
B	100	86	64	57	54	53	51	49	45	37	33	28	22	...
C	0	27	36	42	60	75	54	44	38	48	60	72	88	...

Our Wishes VS Existing Approaches

- use a signed PKN + TS
- synthesise *all* the compatible BNs (with all the equivalent minDNFs)
- no assumption on the class of functions and on the underlying update scheme of the seq. of config.

Our Wishes VS Existing Approaches

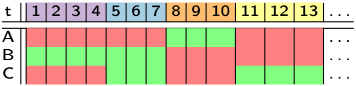
- use a signed PKN + TS
- synthesise *all* the compatible BNs (with all the equivalent minDNFs)
- no assumption on the class of functions and on the underlying update scheme of the seq. of config.

	signed PKN	all minDNF, all class		assumption on TS & config. seq.
REVEAL	✗	✗	✓	each timestep = sync. transition
Best-Fit	✗	✗	✓	each timestep = sync. transition
caspo-TS	✓	✓	monotonous	async. reachability

Our Wishes VS Existing Approaches

- use a signed PKN + TS
- synthesise *all* the compatible BNs (with all the equivalent minDNFs)
- no assumption on the class of functions and on the underlying update scheme of the seq. of config.

	signed PKN	all minDNF,	all class	assumption on TS & config. seq.
REVEAL	✗	✗	✓	each timestep = sync. transition
Best-Fit	✗	✗	✓	each timestep = sync. transition
caspo-TS	✓	✓	monotonous	async. reachability



010 → 011 → 100 → 001

010 → 010 → 010 → 010 → 011 → 011 → ...

010 → * → 011 → * → 100 → * → 001 → *

Our own approach: ASKeD-BN

For each component: use ASP to generate all the possible transition functions (in minDNF) compatible with a given PKN and TS. Then: use python to produce all the possible BNs.

Why ASP? Because…

- several tools are now developed with ASP in systems biology
- we can focus only on modeling the problem and not on the way to get the solutions
- we were told it is very fast and efficient, fun to learn, …

(and at the end we happy of this choice! :))))

ASKeD-BN— modeling a candidate DNF

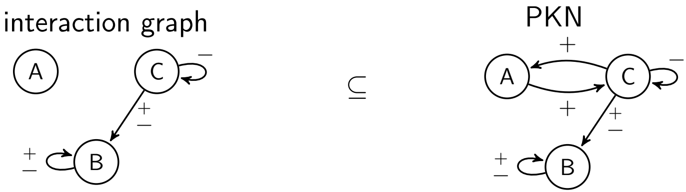
ASP picks a subset of conjunctions among all the possible ones (given)

```
1{ conjTakenID(0..maxNbPossibleConj)}.
conjTaken(I, N, V) :- conj(I, _, _); conjTakenID(I).
```

```
% GIVEN : conj(ID, Component, Sign}
conj(0, a,  0). conj(0, b, 0). conj(0, c,  0).
conj(1, a,  1). conj(1, b,-1). conj(1, c,  0). %  A ∧ ¬B
conj(2, a, -1). conj(1, b, 0). conj(1, c, -1). % ¬A ∧ ¬C
conj(3, a, -1). conj(3, b,-1). conj(3, c, -1). % ¬A ∧ ¬B ∧ ¬C
conj(4, a,  1). conj(4, b, 1). conj(4, c,  1). %  A ∧  B ∧  C
...
```

Example: taken = $\{1, 2\}$ → candidate = (A ∧ ¬B) ∨ (¬A ∧ ¬C)

ASKeD-BN— structural constraints



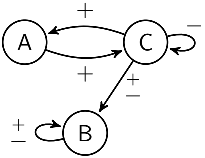
"it is false to select a conjunction that uses a literal that is not allowed by the PKN"

```
ig(ParentID, x, V):- conjTaken(ConjID, ParentID, V); V!=0.
:- ig(ParentID, x, V) ; not pkn(ParentID, x, V).
```

ASKeD-BN— dynamical constraints

(1) Use configurations sequence with the parcimonious update schema possible + the PKN to build partial truth tables

$$010 \xrightarrow{\{C\}} 011 \xrightarrow{\{A,B,C\}} 100 \xrightarrow{\{A,C\}} 001$$



	putative input	output
for A:	C	
0	0	0
1	1	1
for B:	B, C	
0	00	
1	01	
2	10	
3	11	0
for C:	A, C	
0	00	
1	01	0
2	10	1
3	11	

ASKeD-BN— dynamical constraints

(2) discard candidates that doesn't match the truth table

examples of eliminated candidates

for A:

0

$\neg C$

for B:

1

$B \vee C$

$B \wedge C$

$(A \wedge B) \vee (\neg A \wedge \neg B)$

for C:

0

1

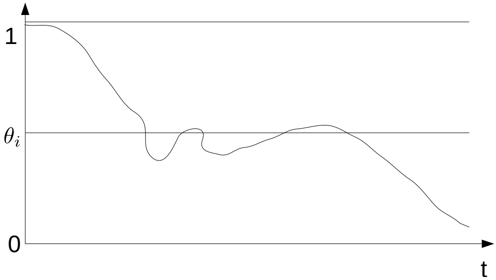
C

	putative input	output
for A:	C	
0	0	0
1	1	1
for B:	B, C	
0	00	
1	01	
2	10	
3	11	0
for C:	A, C	
0	00	
1	01	0
2	10	1
3	11	

ASKeD-BN— dynamical constraints

(3) Optional: minimize the error (to avoid UNSAT)

```
#minimize{E@2 : error(E)}. % highest priority
```



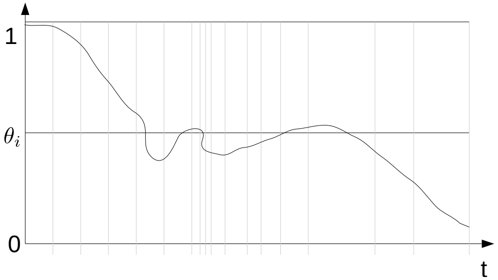
i_t: continuous value of i at time t

θ_i: binarisation threshold for i

ASKeD-BN— dynamical constraints

(3) Optional: minimize the error (to avoid UNSAT)

```
#minimize{E@2 : error(E)}. % highest priority
```



i_t: continuous value of i at time t

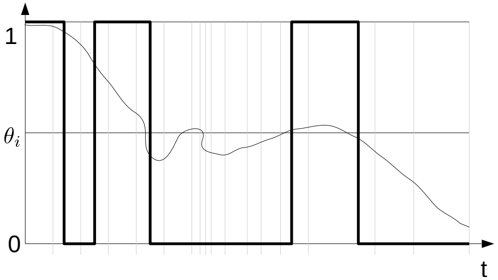
θ_i: binarisation threshold for i

T: # time steps

ASKeD-BN— dynamical constraints

(3) Optional: minimize the error (to avoid UNSAT)

```
#minimize{E@2 : error(E)}. % highest priority
```



i_t: continuous value of i at time t

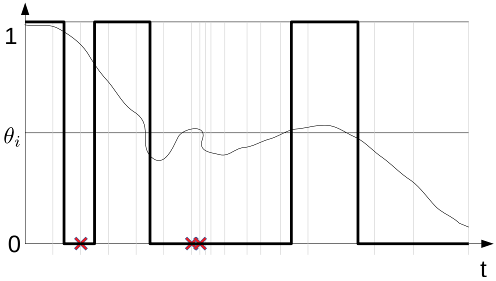
θ_i: binarisation threshold for i

T: # time steps

ASKeD-BN— dynamical constraints

(3) Optional: minimize the error (to avoid UNSAT)

```
#minimize{E@2 : error(E)}. % highest priority
```



i_t: continuous value of i at time t

θ_i: binarisation threshold for i

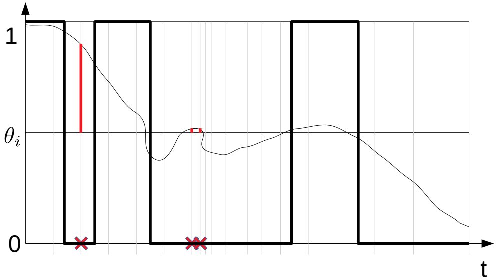
T: # time steps

$\mathscr{U}$: set of unexplained time steps

ASKeD-BN— dynamical constraints

(3) Optional: minimize the error (to avoid UNSAT)

```
#minimize{E@2 : error(E)}. % highest priority
```



i_t: continuous value of i at time t
θ_i: binarisation threshold for i
T: # time steps
$\mathscr{U}$: set of unexplained time steps

minimise the Mean Absolute Error (ideally 0)

$$\text{MAE}_{f_i} = \frac{\sum_{t \in \mathscr{U}_{f_i}} |\theta_i - i_t|}{T}$$

ASKeD-BN— minimality constraint

Find the smallest minDNF(s) among the minDNFs compatible with the (partial) truth table

	putative input	output	possible guess			
0	00		0	1	0	1
1	01	0	0	0	0	0
2	10	1	1	1	1	1
3	11		0	0	1	1
		minDNF	$\neg A \wedge B$	$\neg A$	B	$\neg A \vee B$
		size	2	1	1	2

```
sizeconj(C, S):-conjTakenID(C);S=#sum{|V|,N:conj(C, N, V)} .
sizeDNF(S):- S=#sum{N,C: sizeconj(C, N), conjTakenID(C)} .
% N elements in conjunction C
#minimize{S@1 : sizeDNF(S)}. % lower priority
```

`ASKeD-BN`— Evaluation and Results

Main goal: transforming existing ODE-like biological models to Boolean networks.

2 papers so far.
Nice results on our criteria, on this (**very specific**) application case
$\rightarrow$ better than `REVEAL`, `Best-Fit` and `caspo-TS`

- IG $\subseteq$ PKN (by construction)
- the mixed STG BNs recovers a good proportion of the transitions of the sequence.
- small number of BNs syntesised (thanks to mincard minDNF)

Remaining Things to Investigate

- overfitting to the given seq. of configurations? (drawback of mincard minDNF)
- choice binarisation procedure and error measure
- long solving time & a lot of memory ($>$ 30h, $>$ 700 Go RAM)
  $\rightarrow$ is there a better encoding possible? better clingo options?

Thanks for your attention.

Enjoy the workshop and conference, and come try the "Bergamotes de Nancy" I brought (available starting tomorrow)

athenais.vaginay@loria.fr

(looking for "write a PhD thesis" and "find a post-doc" advice :)))

$$\mathscr{R} = \begin{cases} r_{\text{on}} = e_{\text{on}} & : \mathsf{S} + \mathsf{E} \quad \rightarrow \mathsf{C} \\ r_{\text{off}} = e_{\text{off}} & : \mathsf{C} \qquad\quad\; \rightarrow \mathsf{S} + \mathsf{E} \\ r_{\text{cat}} = e_{\text{cat}} & : \mathsf{C} \qquad\quad\; \rightarrow \mathsf{E} + 2 \times \mathsf{P} \end{cases}$$

$$\mathscr{R} = \begin{cases} r_{\text{on}} = e_{\text{on}} & : \mathsf{S} + \mathsf{E} \rightarrow \mathsf{C} \\ r_{\text{off}} = e_{\text{off}} & : \mathsf{C} \rightarrow \mathsf{S} + \mathsf{E} \\ r_{\text{cat}} = e_{\text{cat}} & : \mathsf{C} \rightarrow \mathsf{E} + 2 \times \mathsf{P} \end{cases}$$

$$\mathscr{R} = \begin{cases} r_{\text{on}} = e_{\text{on}} & : \mathsf{S} + \mathsf{E} \rightarrow \mathsf{C} \\ r_{\text{off}} = e_{\text{off}} & : \mathsf{C} \rightarrow \mathsf{S} + \mathsf{E} \\ r_{\text{cat}} = e_{\text{cat}} & : \mathsf{C} \rightarrow \mathsf{E} + 2 \times \mathsf{P} \end{cases}$$

$$\mathscr{R} = \begin{cases} r_{\text{on}} = e_{\text{on}} & : \mathsf{S} + \mathsf{E} \quad \rightarrow \mathsf{C} \\ r_{\text{off}} = e_{\text{off}} & : \mathsf{C} \quad \rightarrow \mathsf{S} + \mathsf{E} \\ r_{\text{cat}} = e_{\text{cat}} & : \mathsf{C} \quad \rightarrow \mathsf{E} + 2 \times \mathsf{P} \end{cases}$$

$$
\mathscr{R} = \begin{cases} r_{\text{on}} = e_{\text{on}} & : \mathsf{S} + \mathsf{E} \rightarrow \mathsf{C} \\ r_{\text{off}} = e_{\text{off}} & : \mathsf{C} \rightarrow \mathsf{S} + \mathsf{E} \\ r_{\text{cat}} = e_{\text{cat}} & : \mathsf{C} \rightarrow \mathsf{E} + 2 \times \mathsf{P} \end{cases}
$$

Chemical Reactions Network

$$\mathscr{R} = \overbrace{\mathsf{E} + \mathsf{S} \underbrace{\overset{e_{\text{on}}}{\underset{e_{\text{off}}}{\longleftrightarrow}} \mathsf{C}}_{r_{\text{off}}}}^{r_{\text{on}}} \overbrace{\xrightarrow{e_{\text{cat}}} \mathsf{E} + 2 \times \mathsf{P}}^{r_{\text{cat}}}$$

A reaction transforms some reactants to products at a given speed

Chemical Reactions Network

$$\mathscr{R} = \overbrace{\mathsf{E} + \mathsf{S} \xleftrightarrow[e_{\text{off}}]{e_{\text{on}}} \mathsf{C}}^{r_{\text{on}}}_{\underbrace{}_{r_{\text{off}}}} \overbrace{\xrightarrow{e_{\text{cat}}} \mathsf{E} + 2 \times \mathsf{P}}^{r_{\text{cat}}}$$

A reaction transforms some reactants to products at a given speed

Chemical Reactions Network

$$\mathscr{R} = \overbrace{\mathsf{E} + \mathsf{S} \underbrace{\xrightleftharpoons[e_{\text{off}}]{e_{\text{on}}}}_{} \mathsf{C}}^{r_{\text{on}}} \xrightarrow{e_{\text{cat}}} \mathsf{E} + 2 \times \mathsf{P}$$

$$\underbrace{\mathsf{E} + \mathsf{S} \longleftrightarrow \mathsf{C}}_{r_{\text{off}}} \qquad \overbrace{\mathsf{C} \longrightarrow \mathsf{E} + 2 \times \mathsf{P}}^{r_{\text{cat}}}$$

A reaction transforms some reactants to products at a given speed

Chemical Reactions Network

$$\mathscr{R} = \overbrace{\mathsf{E} + \mathsf{S} \underbrace{\xleftrightarrow[e_{\text{off}}]{e_{\text{on}}} \mathsf{C}}_{}}^{r_{\text{on}}} \overbrace{\xrightarrow{e_{\text{cat}}} \mathsf{E} + 2 \times \mathsf{P}}^{r_{\text{cat}}}$$

$$\underbrace{\mathsf{E} + \mathsf{S} \xleftrightarrow[e_{\text{off}}]{e_{\text{on}}} \textcolor{blue}{\mathsf{C}}}_{r_{\text{off}}}$$

A reaction transforms some reactants to products at a given speed

Chemical Reactions Network

$$\mathscr{R} = \overbrace{\mathsf{E} + \mathsf{S} \underbrace{\overset{\color{blue}{e_{\text{on}}}}{\underset{e_{\text{off}}}{\longleftrightarrow}} \mathsf{C}}_{}}^{r_{\text{on}}} \overbrace{\xrightarrow{e_{\text{cat}}} \mathsf{E} + 2 \times \mathsf{P}}^{r_{\text{cat}}}$$

$$\underbrace{\mathsf{E} + \mathsf{S} \longleftrightarrow \mathsf{C}}_{r_{\text{off}}}$$

A reaction transforms some reactants to products at a given speed

Chemical Reaction Network — ODEs

$$\mathscr{R} = \overbrace{\mathsf{E} + \mathsf{S} \underbrace{\xleftrightarrow[e_{\text{off}}]{e_{\text{on}}} \mathsf{C}}_{}}^{r_{\text{on}}}$$

$$\mathscr{R} = \underbrace{\overbrace{\mathsf{E} + \mathsf{S} \xleftrightarrow[e_{\text{off}}]{e_{\text{on}}} \mathsf{C}}^{r_{\text{on}}}}_{r_{\text{off}}} \overbrace{\xrightarrow{e_{\text{cat}}} \mathsf{E} + 2 \times \mathsf{P}}^{r_{\text{cat}}}$$

Chemical Reaction Network — ODEs

$$\mathscr{R} = \overbrace{\underbrace{\mathsf{E} + \mathsf{S} \xleftrightarrow[e_{\text{off}}]{e_{\text{on}}} \mathsf{C}}_{r_{\text{off}}}}^{r_{\text{on}}}\!\!\overbrace{\xrightarrow{e_{\text{cat}}} \mathsf{E} + 2 \times \mathsf{P}}^{r_{\text{cat}}}$$

$$\forall i \in V : \frac{\mathrm{d}i}{\mathrm{d}t} = \sum_{r \in \mathscr{R}} e_r \times \delta_r(i)$$

Chemical Reaction Network — ODEs

$$\mathscr{R} = \overbrace{\underbrace{\mathsf{E} + \mathsf{S} \xrightleftharpoons[e_{\text{off}}]{e_{\text{on}}} \mathsf{C}}_{r_{\text{off}}}}^{r_{\text{on}}} \overbrace{\xrightarrow{e_{\text{cat}}} \mathsf{E} + 2 \times \mathsf{P}}^{r_{\text{cat}}}$$

$$\forall i \in V : \frac{\mathrm{d}i}{\mathrm{d}t} = \sum_{r \in \mathscr{R}} e_r \times \delta_r(i)$$

$$\frac{\mathrm{d}[\mathsf{C}]}{\mathrm{d}t} = \underbrace{e_{\text{on}} \times 1}_{r_{\text{on}}} + \underbrace{e_{\text{off}} \times -1}_{r_{\text{off}}} + \underbrace{e_{\text{cat}} \times -1}_{r_{\text{cat}}}$$

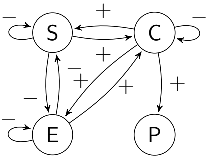
Reaction Network — Structure and Dynamics

$$\mathscr{R} = \overbrace{\underbrace{\mathsf{E} + \mathsf{S} \xleftrightarrow[e_{\text{off}}]{e_{\text{on}}} \mathsf{C}}_{r_{\text{off}}}}^{r_{\text{on}}} \overbrace{\xrightarrow{e_{\text{cat}}} \mathsf{E} + 2 \times \mathsf{P}}^{r_{\text{cat}}}$$

Structure:

1. **If** Y is a reactant and X disappears
   then $\mathsf{Y} \xrightarrow{-} \mathsf{X}$
2. **If** Y is a reactant and X appears
   then $\mathsf{Y} \xrightarrow{+} \mathsf{X}$

[Fages et al. 2008]



Dynamics:
numerical simulation of the ODEs
+ binarisation

	putative input	output	candidate functions							
for B:	B, C									
0	00		0	1	0	1	0	1	0	1
1	01		0	0	1	1	0	0	1	1
2	10		0	0	0	0	1	1	1	1
3	11	0	0	0	0	0	0	0	0	0
			rota	rota	rota	rota	rota	rota	rota	rota

	putative input	output	with guess			
for C:	A, C					
0	00		0	1	0	1
1	01	0	0	0	0	0
2	10	1	1	1	1	1
3	11		0	0	1	1
			$\neg A \wedge B$			

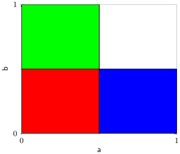
ASKeD-BN— minimality constraints

→ For finding the minDNF(s) given a truth table

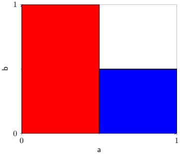
	putative input	output
0	00	1
1	01	1
2	10	1
3	11	0

Several candidate DNFs, but only one minimal

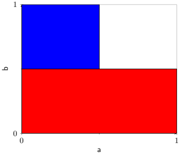
$\neg A \wedge \neg B$
$\vee (A \wedge \neg B)$
$\vee (\neg A \wedge B)$



$\neg A$
$\vee (A \wedge \neg B)$



$\neg B$
$\vee (\neg A \wedge B)$



$\neg A \vee \neg B$