Model Revision of Boolean Logical Models of Biological Regulatory Networks

Doctoral Program in Computer Science and Engineering

Filipe Gouveia

Advisor: Prof. Pedro Monteiro
Co-advisor: Prof. Inês Lynce

INESC-ID / Instituto Superior Técnico
Universidade de Lisboa
Lisbon, Portugal

September 2021

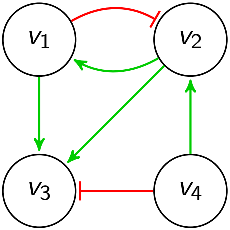
Table of Contents

Regulatory Networks

- Biological processes arise at the cellular level, governed by complex regulatory networks
- Regulatory network:
  - Collection of molecular compounds (e.g. proteins, genes)
  - Compounds interact with each other
- Computational modelling allows:
  - Functional understanding of the network
  - Test hypotheses
  - Identify predictions *in silico*
  - ...

Boolean Logical Model

- Different formalisms can be used [KS08]
  - We consider the Boolean logical formalism [Tho73]
- Compounds represented by a Boolean variable:
  - active/inactive
- Interactions defined as positive (activation) or negative (inhibition)
- Regulations defined as Boolean functions



$$f_{v_1} = v_2$$

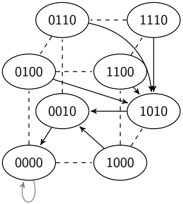
$$f_{v_2} = \neg v_1 \wedge v_4$$

$$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$$

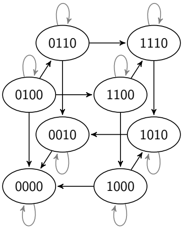
Dynamics

- The value of each node can change through time
  - Defining the state of the network
- The regulatory functions update the value of the corresponding node
- Different update schemes:

Synchronous

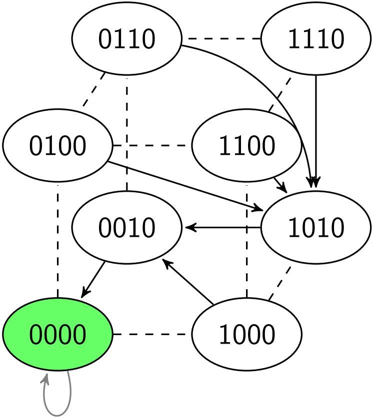


Asynchronous

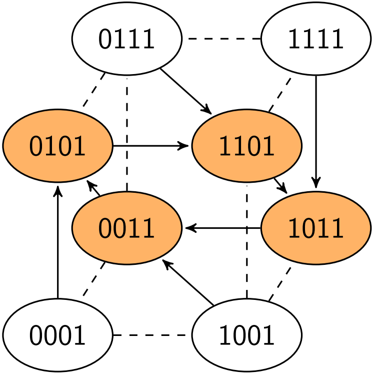


Dynamics

- State Transition Graph (STG) - synchronous update scheme



Stable State (Point Attractor)



Cycle Attractor

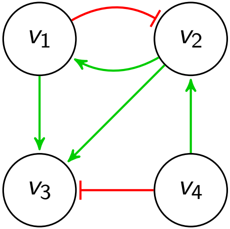
Motivation

- As new experimental data becomes available,
  models may become inconsistent
  - Models may not be able to reproduce the new information
  - Models need to be revised

- Model Revision is mainly a manual task
  - Performed by a modeller
  - Prone to error

Motivation

- How can we repair an inconsistent model?
  - Change a regulatory function?
    - ⋆ 2^{2^k} possibilities for each node!
  - Change the type of interaction?
  - Add or remove interactions?



$$f_{v_1} = v_2$$

$$f_{v_2} = \neg v_1 \wedge v_4$$

$$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$$

There are $\approx 10^{24}$ possible combinations!
(65536 Boolean functions with 4 regulators)

Contributions

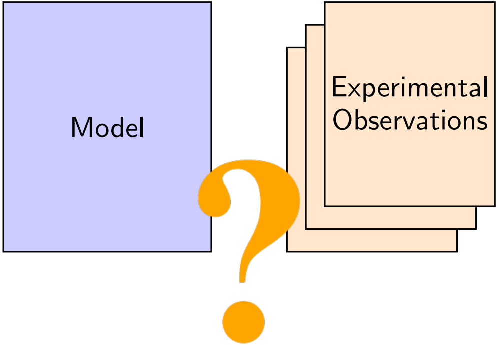
- Model revision approach for Boolean logical models of biological regulatory networks
  - Confronting a model with:
    - Stable state observations
    - Time-series observations
  - Consider different dynamics:
    - Synchronous update scheme
    - Asynchronous update scheme

- ModRev tool
  - Implements the model revision procedure
  - Produces all optimum repair sets to repair an inconsistent model
  - ModRev available at https://filipegouveia.github.io/ModelRevisionASP/

Contributions

- "*ModRev - Model Revision Tool for Boolean Logical Models of Biological Regulatory Networks*", CMSB 2020 [GLM20a]
- "*Revision of Boolean Models of Regulatory Networks Using Stable State Observations*", Journal of Computational Biology [GLM20b]
- "*Model Revision of Boolean Regulatory Networks at Stable State*", ISBRA 2019 [GLM19]
- "*Model Revision of Logical Regulatory Networks Using Logic-Based Tools*", ICLP Doctoral Consortium 2018 [GLM18]

Model Revision



- A model is *consistent* if all of its nodes are consistent
  - Value of each node given by its regulatory function is equal to the observed value
- A model is *inconsistent* otherwise
  - Needs to be revised

Possible causes of inconsistency and repair operations:

Cause	**Repair Operation**
Wrong Regulatory Function	Function change
Wrong Interaction Type	Edge sign flip
Wrong Regulator	Edge removal
Missing Regulator	Edge addition

Model Revision

Possible causes of inconsistency and repair operations:

Cause	Repair Operation
Wrong Regulatory Function	Function change
Wrong Interaction Type	Edge sign flip
Wrong Regulator	Edge removal
Missing Regulator	Edge addition

Optimization Criteria:

1. Minimize interaction addition/removal
2. Minimize interaction type changes
3. Minimize Boolean function changes

Model Revision Approach

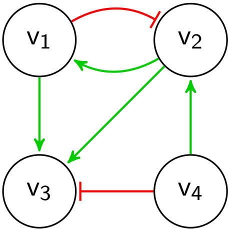
Iterate over each inconsistent node:

- Try to change the function
- Try to flip the sign of 1 edge
  - Consider changing the function again
  - Repeat this step for 2 edges, and so on
- Try to add or remove 1 edge
  - Consider changing the function and/or flip the sign of edges as previously
  - Repeat this step for adding or removing 2 edges and so on

Boolean Functions

Assumption: consider monotone non-degenerate Boolean functions

- **Monotone**: each regulator only has one role: positive/activation or negative/inhibition
  - each variable appears with the same sign in the function in Blake Canonical Form (BCF)



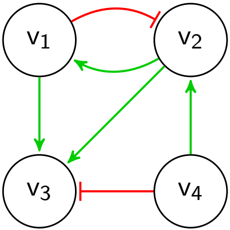
$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$ Monotone

$f_{v_3} = (v_1 \wedge \neg v_2) \vee (v_2 \wedge \neg v_4)$ Non-monotone

Boolean Functions

Assumption: consider monotone non-degenerate Boolean functions

- **Monotone**: each regulator only has one role: positive/activation or negative/inhibition
  - each variable appears with the same sign in the function in Blake Canonical Form (BCF)
- **Non-degenerate**: each regulator is essential in the regulatory function
  - each variable has an impact on the truth table



$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$ Non-degenerate

$f_{v_3} = (v_2 \wedge \neg v_4)$ (missing v_1) Degenerate

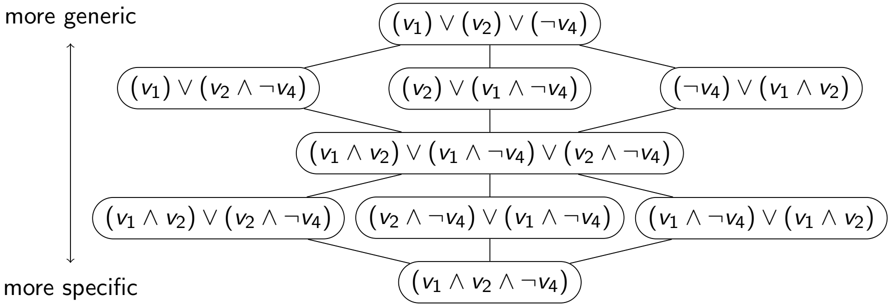
Boolean Functions

- Relation $\preceq$ between monotone non-degenerate Boolean functions [CMC19]

$$f \preceq f' \iff f(X) \Rightarrow f'(X).$$

where $f(X)$ denotes the entries where the function is **true** (or 1)

- f' is a **parent** of f iff $f \preceq f'$ and $\nexists f''$ such that $f \preceq f''$ and $f'' \preceq f'$
  - f is a **child** of f'

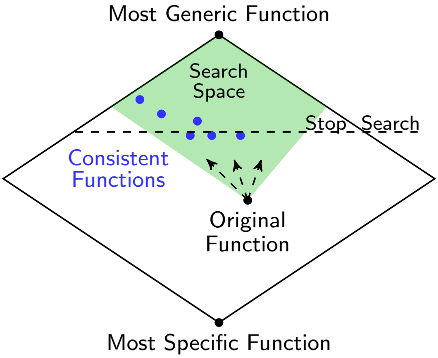

more generic
more specific
$(v_1) \vee (v_2) \vee (\neg v_4)$
$(v_1) \vee (v_2 \wedge \neg v_4)$
$(v_2) \vee (v_1 \wedge \neg v_4)$
$(\neg v_4) \vee (v_1 \wedge v_2)$
$(v_1 \wedge v_2) \vee (v_1 \wedge \neg v_4) \vee (v_2 \wedge \neg v_4)$
$(v_1 \wedge v_2) \vee (v_2 \wedge \neg v_4)$
$(v_2 \wedge \neg v_4) \vee (v_1 \wedge \neg v_4)$
$(v_1 \wedge \neg v_4) \vee (v_1 \wedge v_2)$
$(v_1 \wedge v_2 \wedge \neg v_4)$

Function Repair

If a function is inconsistent:

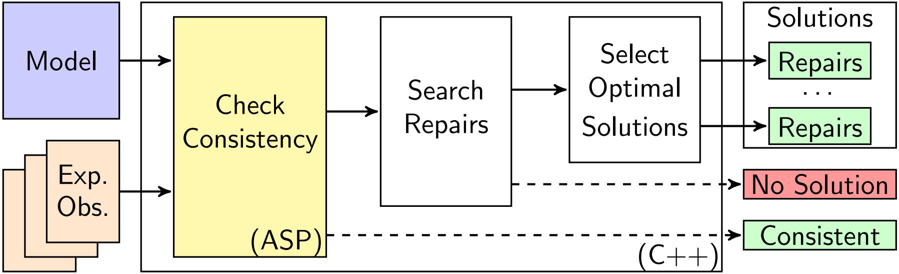
- Determine if is necessary to generalize or specify the function
- Compute set of parents (children) to go up (down) the diagram
- Continue to do so until a consistent function is found
  - or no function is found

If a function is found, is closest to the original function



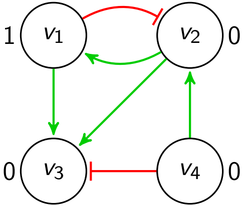
Model Revision

Architecture



Model Revision

Stable State Example



$f_{v_1} = v_2$

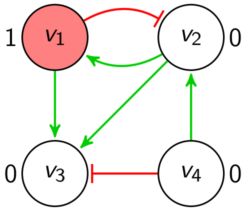
$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

- Inconsistent Model
- Node v_1 is inconsistent
- Change the interaction between v_2 and v_1
- Node v_3 is inconsistent
- Consistent Model

Model Revision

Stable State Example



$f_{v_1} = v_2$

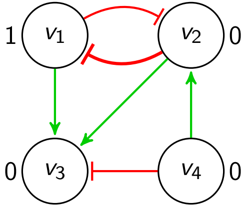
$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

- Inconsistent Model
- Node v_1 is inconsistent
- Change the interaction between v_2 and v_1
- Node v_3 is inconsistent
- Consistent Model

Model Revision

Stable State Example

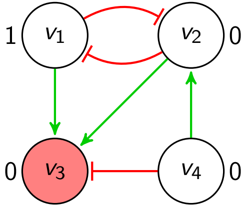


$f_{v_1} = \neg v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

- Inconsistent Model
- Node v_1 is inconsistent
- Change the interaction between v_2 and v_1
- Node v_3 is inconsistent
- Consistent Model



$f_{v_1} = \neg v_2$

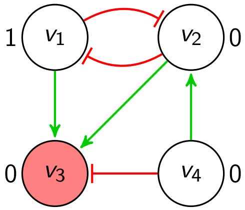
$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

- Inconsistent Model
- Node v_1 is inconsistent
- Change the interaction between v_2 and v_1
- Node v_3 is inconsistent
- Consistent Model

Model Revision

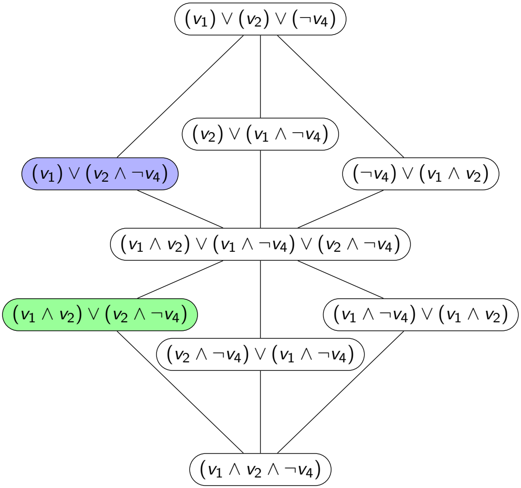
Stable State Example



$f_{v_1} = \neg v_2$

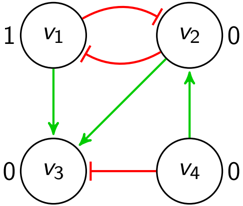
$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$



Model Revision

Stable State Example



$f_{v_1} = \neg v_2$

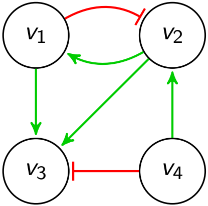
$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = (v_1 \wedge v_2) \vee (v_2 \wedge \neg v_4)$

- Inconsistent Model
- Node v_1 is inconsistent
- Change the interaction between v_2 and v_1
- Node v_3 is inconsistent
- Consistent Model

Model Revision

Time-series Example - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

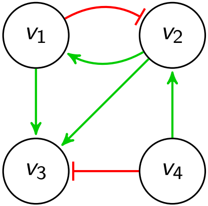
$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	T_0	T_1	T_2
v_1	1	1	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

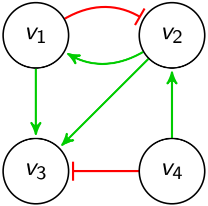
$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	T_0	T_1	T_2
v_1	1	1	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \land v_4$

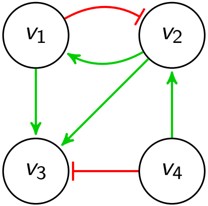
$f_{v_3} = v_1 \lor (v_2 \land \neg v_4)$

Time

Node	$\boldsymbol{T_0}$	$\boldsymbol{T_1}$	$\boldsymbol{T_2}$
$\boldsymbol{v_1}$	1	1	0
$\boldsymbol{v_2}$	1	0	1
$\boldsymbol{v_3}$	0	1	0
$\boldsymbol{v_4}$	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

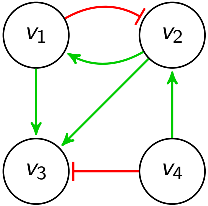
$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	$\boldsymbol{T}_0$	$\boldsymbol{T}_1$	$\boldsymbol{T}_2$
$\boldsymbol{v}_1$	1	1	0
$\boldsymbol{v}_2$	1	0	1
$\boldsymbol{v}_3$	0	1	0
$\boldsymbol{v}_4$	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



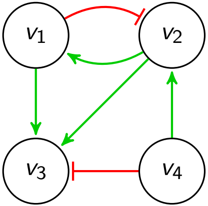
$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	$\boldsymbol{T}_0$	$\boldsymbol{T}_1$	$\boldsymbol{T}_2$
$\boldsymbol{v}_1$	1	1	0
$\boldsymbol{v}_2$	1	0	1
$\boldsymbol{v}_3$	0	1	0
$\boldsymbol{v}_4$	1	1	1



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

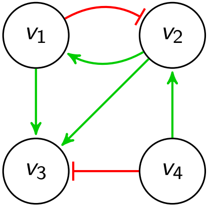
$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	$\boldsymbol{T_0}$	$\boldsymbol{T_1}$	$\boldsymbol{T_2}$
$\boldsymbol{v_1}$	1	1	0
$\boldsymbol{v_2}$	1	0	1
$\boldsymbol{v_3}$	0	1	0
$\boldsymbol{v_4}$	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \land v_4$

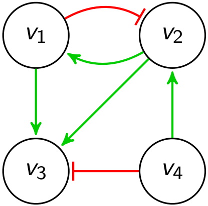
$f_{v_3} = v_1 \lor (v_2 \land \neg v_4)$

Time

Node	$\boldsymbol{T_0}$	$\boldsymbol{T_1}$	$\boldsymbol{T_2}$
$\boldsymbol{v_1}$	1	1	0
$\boldsymbol{v_2}$	1	0	1
$\boldsymbol{v_3}$	0	1	0
$\boldsymbol{v_4}$	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

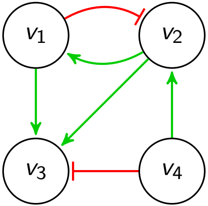
$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	T_0	T_1	T_2
v_1	1	1	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \land v_4$

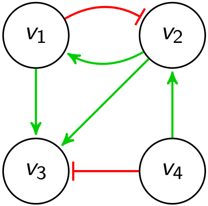
$f_{v_3} = v_1 \lor (v_2 \land \neg v_4)$

Time

Node	T_0	T_1	T_2
v_1	1	1	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

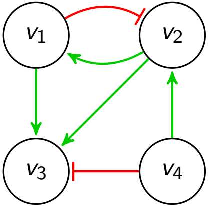
$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	$\boldsymbol{T_0}$	$\boldsymbol{T_1}$	$\boldsymbol{T_2}$
$\boldsymbol{v_1}$	1	1	0
$\boldsymbol{v_2}$	1	0	1
$\boldsymbol{v_3}$	0	1	0
$\boldsymbol{v_4}$	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

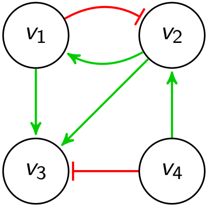
$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	T_0	T_1	T_2
v_1	1	1	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \land v_4$

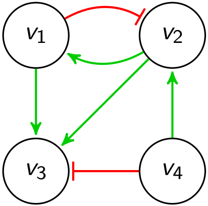
$f_{v_3} = v_1 \lor (v_2 \land \neg v_4)$

Time

Node	T_0	T_1	T_2
v_1	1	1	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



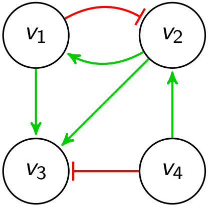
$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	T_0	T_1	T_2
v_1	1	1	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1



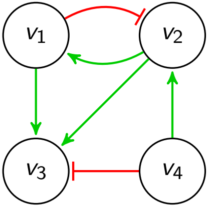
$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \land v_4$

$f_{v_3} = v_1 \lor (v_2 \land \neg v_4)$

Time

Node	$\boldsymbol{T_0}$	$\boldsymbol{T_1}$	$\boldsymbol{T_2}$
$\boldsymbol{v_1}$	1	1	0
$\boldsymbol{v_2}$	1	0	1
$\boldsymbol{v_3}$	0	1	0
$\boldsymbol{v_4}$	1	1	1



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \land v_4$

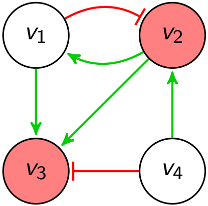
$f_{v_3} = v_1 \lor (v_2 \land \neg v_4)$

Time

Node	$\boldsymbol{T}_0$	$\boldsymbol{T}_1$	$\boldsymbol{T}_2$
$\boldsymbol{v}_1$	1	1	0
$\boldsymbol{v}_2$	1	0	1
$\boldsymbol{v}_3$	0	1	0
$\boldsymbol{v}_4$	1	1	1

Model Revision

Time-series Example - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \land v_4$

$f_{v_3} = v_1 \lor (v_2 \land \neg v_4)$

Time

Node	$\boldsymbol{T}_0$	$\boldsymbol{T}_1$	$\boldsymbol{T}_2$
$\boldsymbol{v}_1$	1	1	0
$\boldsymbol{v}_2$	1	0	1
$\boldsymbol{v}_3$	0	1	0
$\boldsymbol{v}_4$	1	1	1

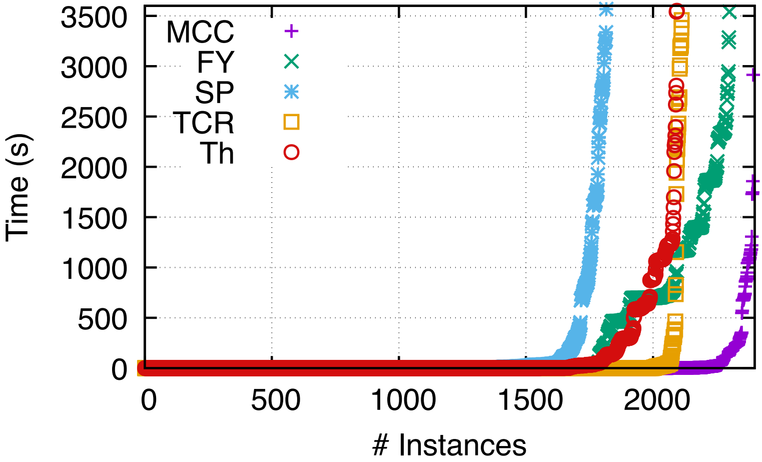
Evaluation

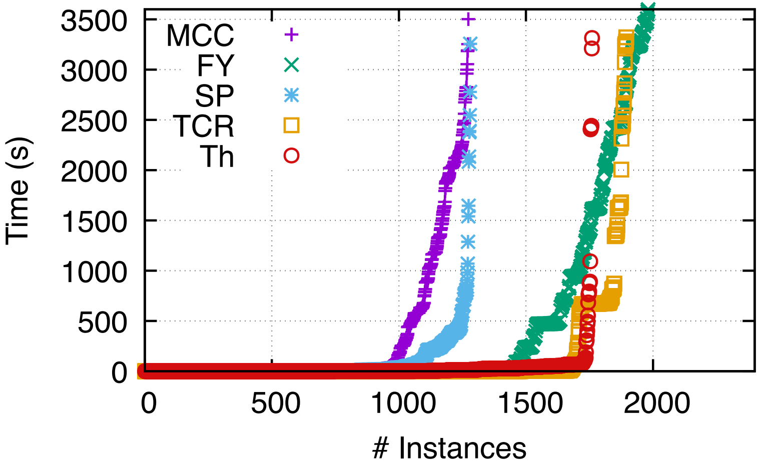
- 5 well-known biological models considered

Abbr.	Model	#N	#E	#SS	Reg.	Ref.
FY	Fission Yeast	10	27	12	5	[DB08]
SP	Segment Polarity	19	57	7	8	[SCT02]
TCR	TCR Signalisation	40	57	7	5	[Kla+06]
MCC	Mammalian Cell Cycle	10	35	1	6	[Fau+06]
Th	Th Cell Differentiation	23	35	3	5	[MX06]

Evaluation

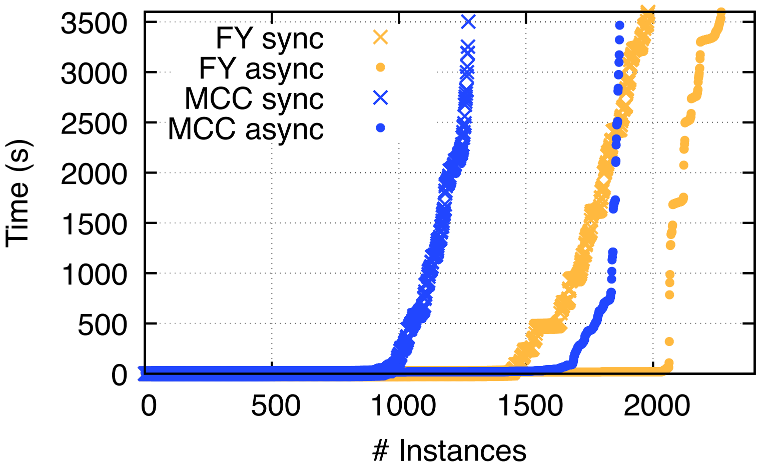
- Random changes were made according to probabilistic parameters:
  - F% : Change a **F**unction
  - E% : Flip the sign of an **E**dge
  - R% : **R**emove an existing edge
  - A% : **A**dd a missing edge
- Several configurations of these parameters were considered
- 100 instances were generated for each configuration for each model
- Timeout of 3600 seconds was considered


MCC
FY
SP
TCR
Th
Time (s)
3500
3000
2500
2000
1500
1000
500
0
0
500
1000
1500
2000
Instances


MCC
FY
SP
TCR
Th
Time (s)
3500
3000
2500
2000
1500
1000
500
0
0
500
1000
1500
2000
Instances

Results

Time-series Results - Synchronous vs Asynchronous



Results

- More connected networks have higher solving times
- Topologically corrupted models lead to higher solving times
  - Change the sign of edges
  - Remove edges
  - Add edges
- Asynchronous update scheme performs better than the synchronous update scheme:
  - 91,82% of instances solved considering asynchronous update scheme
  - 76,84% of instances solved considering synchronous update scheme

Conclusion

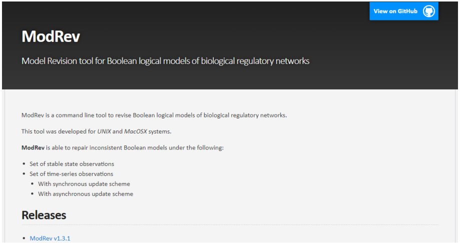
- Proposed a model revision approach for Boolean logical models of biological regulatory networks

- Considering both:
  - Stable state observations
  - Time-series observations

- Considering different dynamics:
  - Synchronous update scheme
  - Asynchronous update scheme

Conclusion

- Model revision approach computes optimum sets of repair operations
  - Optimisation criterion defined
- Model revision approach implemented as a tool: ModRev
  - Assess whether a Boolean logical model is consistent with a set of observations
  - Produces all optimum repair sets to repair an inconsistent model
  - ModRev available at https://filipegouveia.github.io/ModelRevisionASP/



Conclusion

- Approach successfully tested using five well-known biological models
- Able to solve 84,89% of the (108 000) instances
  - Most of the instances solved under 60 seconds
- Degree of connectivity plays a big role on the model revision procedure
- The dimension of the regulatory functions has the biggest impact on the performance
  - Number of monotone non-degenerate Boolean functions increases exponentially

Discussion and Future Work

- Heuristics could be used to reduce the number of solutions

- Study the solutions produced to determine common repair operations

- Consider different approaches:
  - Different ASP encodings
  - Use other logic-based approaches

- Facilitate the interoperability of ModRev with other tools
  - BioLQM toolkit

Thank you!

MODREV https://filipegouveia.github.io/ModelRevisionASP/

Acknowledgements:

FCT Fundação para a Ciência e a Tecnologia

inesc id lisboa

TÉCNICO LISBOA

References

[CMC19] José ER Cury, Pedro T Monteiro, and Claudine Chaouiya. “Partial Order on the set of Boolean Regulatory Functions”. In: *arXiv preprint arXiv:1901.07623* (2019).

[DB08] Maria I. Davidich and Stefan Bornholdt. “Boolean Network Model Predicts Cell Cycle Sequence of Fission Yeast”. In: *PLoS ONE* 3.2 (Feb. 2008). Ed. by Gustavo Stolovitzky, e1672.

[Fau+06] A. Fauré et al. “Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle”. In: *Bioinformatics* 22.14 (July 2006), e124–e131.

[GLM18] Filipe Gouveia, Inês Lynce, and Pedro T. Monteiro. “Model Revision of Logical Regulatory Networks Using Logic-Based Tools”. en. In: *Technical Communications of the 34th International Conference on Logic Programming (ICLP 2018)*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany, 2018. DOI: 10.4230/OASICS.ICLP.2018.23.

[GLM19] Filipe Gouveia, Inês Lynce, and Pedro T. Monteiro. “Model Revision of Boolean Regulatory Networks at Stable State”. In: *Bioinformatics Research and Applications*. Springer International Publishing, 2019, pp. 100–112. DOI: 10.1007/978-3-030-20242-2_9.

[GLM20a] Filipe Gouveia, Inês Lynce, and Pedro T. Monteiro. “ModRev - Model Revision Tool for Boolean Logical Models of Biological Regulatory Networks”. In: *Computational Methods in Systems Biology*. Springer International Publishing, 2020, pp. 339–348. DOI: 10.1007/978-3-030-60327-4_18.

[GLM20b] Filipe Gouveia, Inês Lynce, and Pedro T. Monteiro. “Revision of Boolean Models of Regulatory Networks Using Stable State Observations”. In: *Journal of Computational Biology* 27.2 (Feb. 2020), pp. 144–155. DOI: 10.1089/cmb.2019.0289.

[Kla+06] Steffen Klamt et al. “A methodology for the structural and functional analysis of signaling and regulatory networks”. In: *BMC Bioinformatics* 7.1 (2006), p. 56.

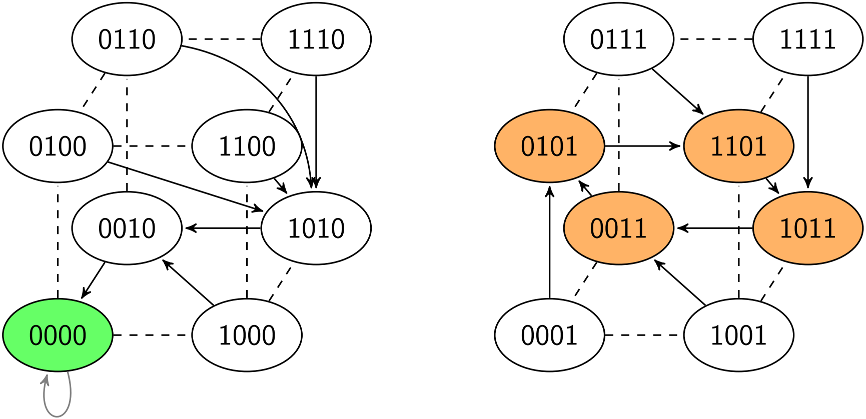
[KS08] Guy Karlebach and Ron Shamir. “Modelling and analysis of gene regulatory networks”. In: *Nature Reviews Molecular Cell Biology* 9.10 (2008), p. 770.

[MX06] Luis Mendoza and Ioannis Xenarios. “A method for the generation of standardized qualitative dynamical systems of regulatory networks”. In: *Theor. Biol. Med. Model.* 3.1 (2006), p. 13.

[SCT02] Lucas Sánchez, Claudine Chaouiya, and Denis Thieffry. “Segmenting the fly embryo: logical analysis of the role of the Segment Polarity cross-regulatory module”. In: *Int. J. Dev. Biol.* 52.8 (2002), pp. 1059–1075.

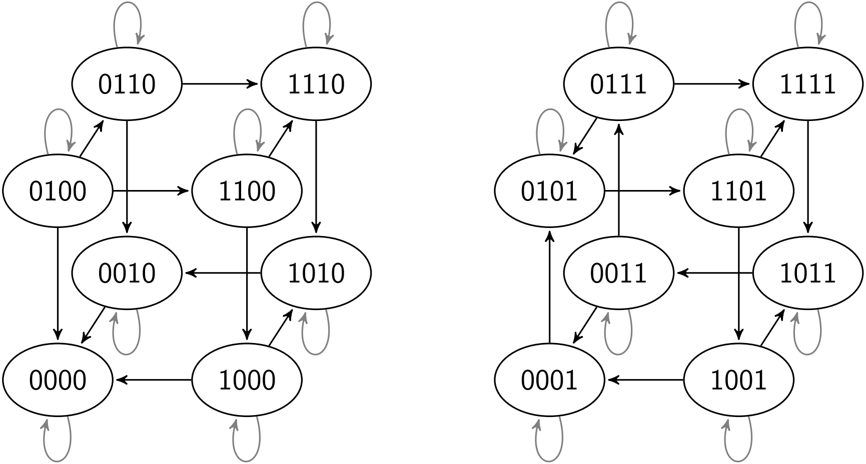
[Tho73] René Thomas. “Boolean formalization of genetic control circuits”. In: *J. Theor. Biol.* 42.3 (1973), pp. 563–585.

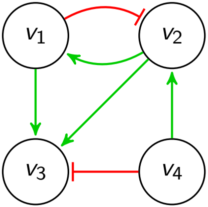
- Different update schemes:
  - Synchronous



Dynamics

- Different update schemes:
  - Asynchronous

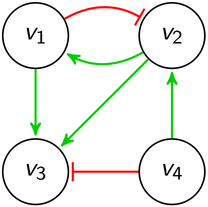




$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$



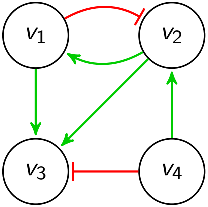
$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Nodes

- `vertex(v1).`
- `vertex(v2).`
- `vertex(v3).`
- `vertex(v4).`



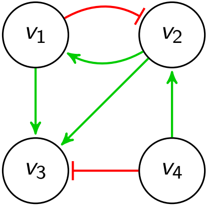
$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \land v_4$

$f_{v_3} = v_1 \lor (v_2 \land \neg v_4)$

Edges

- `edge(v1,v2,0).`
- `edge(v1,v3,1).`
- `edge(v2,v1,1).`
- `edge(v2,v3,1).`
- `edge(v4,v2,1).`
- `edge(v4,v3,0).`



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \land v_4$

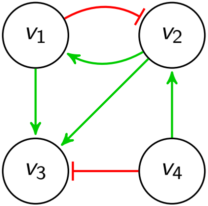
$f_{v_3} = v_1 \lor (v_2 \land \neg v_4)$

Functions

- `functionOr(v1, 1..1).`
- `functionAnd(v1, 1, v2).`

- `functionOr(v2, 1..1).`
- `functionAnd(v2, 1, v1).`
- `functionAnd(v2, 1, v4).`

- `functionOr(v3, 1..2).`
- `functionAnd(v3, 1, v1).`
- `functionAnd(v3, 2, v2).`
- `functionAnd(v3, 2, v4).`



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Observations

- `exp(E).`
- `obs_vlabel(E, V, S).`
  - `E` identifies the experiment
  - `V` identifies the vertex/node
  - `S` $\in 0, 1$ is the observed value
- `obs_vlabel(E, V, S, T).`
  - `E` identifies the experiment
  - `V` identifies the vertex/node
  - `S` $\in 0, 1$ is the observed value
  - `T` identifies the (integer) time step

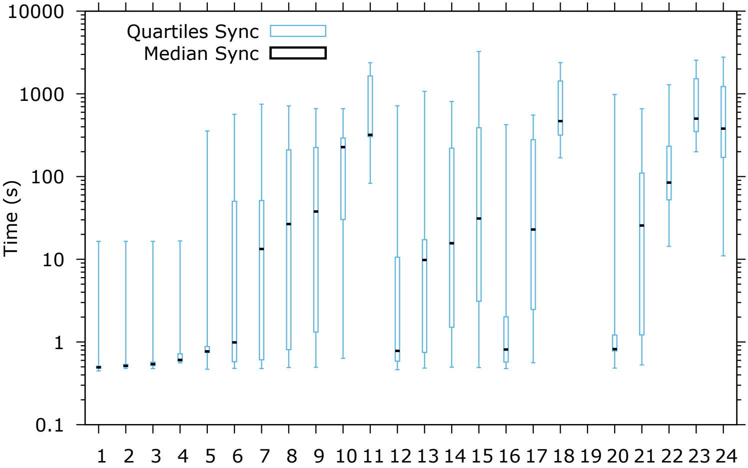
Configuration Parameters

- Random changes were made according to probabilistic parameters
  - F% : Change a **F**unction
  - E% : Flip the sign of an **E**dge
  - R% : **R**emove an existing edge
  - A% : **A**dd a missing edge

Conf.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
F	5	25	50	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	50	100	5	10
E	0	0	0	0	5	10	15	20	25	50	75	0	0	0	0	0	0	0	0	5	25	50	25	10
R	0	0	0	0	0	0	0	0	0	0	0	1	5	10	15	0	0	0	0	0	0	0	5	5
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	5	10	15	0	0	0	5	5

Results

Time-series Results By Configuration - SP - Synchronous (5 observations, 20 time steps)



Models

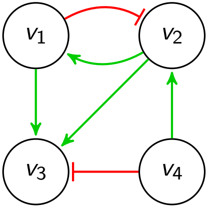
- 5 well-known biological models considered

Abbr.	Model	#N	#E	#SS	Reg.	Ref.
FY	Fission Yeast	10	27	12	5	[DB08]
SP	Segment Polarity	19	57	7	8	[SCT02]
TCR	TCR Signalisation	40	57	7	5	[Kla+06]
MCC	Mammalian Cell Cycle	10	35	1	6	[Fau+06]
Th	Th Cell Differentiation	23	35	3	5	[MX06]

Average Number of Corruptions

(%)				FY	SP	TCR	MCC	Th
F	E	R	A					
5	0	0	0	1	1	1	1	1
25	0	0	0	2	4	3	2	3
50	0	0	0	3	7	5	4	5
100	0	0	0	6	15	10	8	11
0	5	0	0	2	3	3	2	2
0	10	0	0	3	6	6	3	4
0	15	0	0	5	9	8	5	6
0	20	0	0	6	12	12	7	7
0	25	0	0	7	14	14	8	9
0	50	0	0	14	28	28	17	17
0	75	0	0	20	42	43	26	26
0	0	1	0	2	3	3	2	2
0	0	5	0	5	7	7	5	4
0	0	10	0	6	11	11	7	7
0	0	15	0	8	15	16	10	10
0	0	0	1	3	7	30	3	8
0	0	0	5	7	26	112	7	34
0	0	0	10	12	46	193	12	58
0	0	0	15	16	63	269	17	78
25	5	0	0	3	7	5	4	5
50	25	0	0	10	22	19	12	14
100	50	0	0	19	43	39	26	28
5	25	5	5	15	44	128	17	44
10	10	5	5	11	36	118	13	39

Time-series Example 02 - Missing Value - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

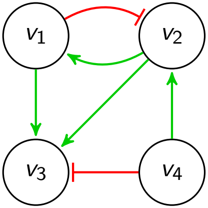
$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	$\boldsymbol{T_0}$	$\boldsymbol{T_1}$	$\boldsymbol{T_2}$
$\boldsymbol{v_1}$	1		0
$\boldsymbol{v_2}$	1	0	1
$\boldsymbol{v_3}$	0	1	0
$\boldsymbol{v_4}$	1	1	1

Model Revision

Time-series Example 02 - Missing Value - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \land v_4$

$f_{v_3} = v_1 \lor (v_2 \land \neg v_4)$

Time

Node	T_0	T_1	T_2
v_1	1		0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Time

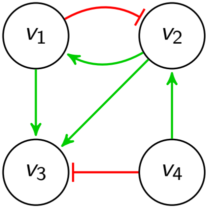
Node	T_0	T_1	T_2
v_1	1	1	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Time

Node	T_0	T_1	T_2
v_1	1	0	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Model Revision

Time-series Example 02 - Missing Value - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	T_0	T_1	T_2
v_1	1		0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Time

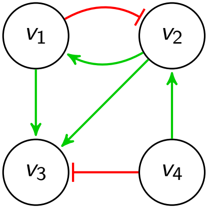
Node	T_0	T_1	T_2
v_1	1	1	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Time

Node	T_0	T_1	T_2
v_1	1	0	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Model Revision

Time-series Example 02 - Missing Value - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	T_0	T_1	T_2
v_1	1		0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Time

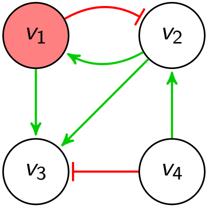
Node	T_0	T_1	T_2
v_1	1	1	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Time

Node	T_0	T_1	T_2
v_1	1	0	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1

Model Revision

Time-series Example 02 - Missing Value - Synchronous Update Scheme



$f_{v_1} = v_2$

$f_{v_2} = \neg v_1 \wedge v_4$

$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Time

Node	T_0	T_1	T_2
v_1	1	0	0
v_2	1	0	1
v_3	0	1	0
v_4	1	1	1