Learning any memory-less discrete semantics for dynamical systems represented by logic programs

Tony Ribeiro[1,2,3], **Maxime Folschette**[4], Morgan Magnin[2,3], Katsumi Inoue[3]

[1]Independant researcher
[2]Université de Nantes, Centrale Nantes, CNRS, LS2N, F-44000 Nantes, France
[3]National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
[4]Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France

2023-09-15
21st International Conference on Computational Methods in Systems Biology (CMSB 2023)

Tony Ribeiro, Maxime Folschette, Morgan Magnin and Katsumi Inoue. **Learning any memory-less discrete semantics for dynamical systems represented by logic programs**. *Machine Learning* 111, Springer. November 2021. `https://doi.org/10.1007/s10994-021-06105-4`

SPRINGER LINK

Find a journal | Publish with us | Search

Home > Machine Learning > Article

Published: 24 November 2021

Learning any memory-less discrete semantics for dynamical systems represented by logic programs

Tony Ribeiro, Maxime Folschette, Morgan Magnin & Katsumi Inoue

Machine Learning **111**, 3593–3670 (2022) | Cite this article

1415 Accesses | **2** Citations | **1** Altmetric | Metrics

Abstract

Learning from interpretation transition (LFIT) automatically constructs a model of the

Contributions

- Algorithm to learn a discrete model from its stage graph
- Independant of the semantics for a defined class of memoryless semantics
- Heuristic for noisy/incomplete data

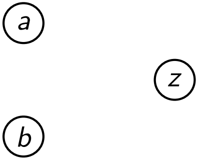
Outline

General Definitions

Discrete Networks / Thomas Modeling

[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

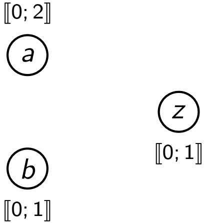
- A set of components $N = \{a, b, z\}$



Discrete Networks / Thomas Modeling

[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

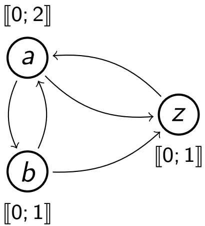
- A set of components $N = \{a, b, z\}$
- A discrete domain for each component $\text{dom}(a) = [\![0; 2]\!]$



Discrete Networks / Thomas Modeling

[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

- A set of components $N = \{a, b, z\}$
- A discrete domain for each component $\mathrm{dom}(a) = [\![0; 2]\!]$
- Discrete parameters / evolution functions $f_a : \mathcal{S} \to \mathrm{dom}(a)$



a	f_b
0	0
1	1
2	1

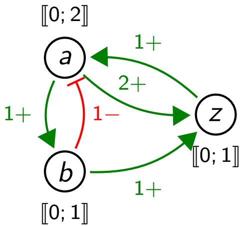
z	b	f_a
0	0	1
0	1	0
1	0	1
1	1	2

a	b	f_z
0	0	0
0	1	0
1	0	0
1	1	0
2	0	0
2	1	1

Discrete Networks / Thomas Modeling

[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

- A set of components $N = \{a, b, z\}$
- A discrete domain for each component $\text{dom}(a) = [\![0; 2]\!]$
- Discrete parameters / evolution functions $f_a : \mathcal{S} \to \text{dom}(a)$
- Signs & thresholds on the edges (redundant) $a \xrightarrow{2+} z$



a	f_b
0	0
1	1
2	1

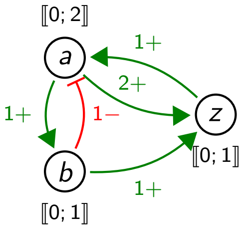
z	b	f_a
0	0	1
0	1	0
1	0	1
1	1	2

a	b	f_z
0	0	0
0	1	0
1	0	0
1	1	0
2	0	0
2	1	1

Discrete Networks / Thomas Modeling

[Kauffman, *Journal of Theoretical Biology*, 1969]
[Thomas, *Journal of Theoretical Biology*, 1973]

- A set of components $N = \{a, b, z\}$
- A discrete domain for each component $\mathrm{dom}(a) = [\![0; 2]\!]$
- Discrete parameters / evolution functions $f_a : \mathcal{S} \to \mathrm{dom}(a)$
- Signs & thresholds on the edges (redundant) $a \xrightarrow{2+} z$



a	f_b
0	0
1	1
2	1

z	b	f_a
0	0	1
0	1	0
1	0	1
1	1	2

a	b	f_z
0	0	0
0	1	0
1	0	0
1	1	0
2	0	0
2	1	1

Semantics = From this information, what are the possible next states?

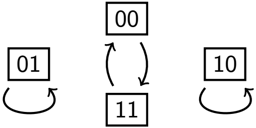
Semantics

$f_a = \neg b$ $\qquad$ $f_b = \neg a$

b	f_a
0	1
1	0

a	f_b
0	1
1	0

State transitions differ according to the update semantics used:



Synchronous

- **Synchronous**: all variables are updated

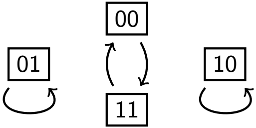
Semantics

$f_a = \neg b$ $f_b = \neg a$

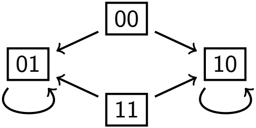
b	f_a
0	1
1	0

a	f_b
0	1
1	0

State transitions differ according to the update semantics used:



Synchronous



Asynchronous

- **Synchronous**: all variables are updated
- **Asynchronous**: only one variable is updated

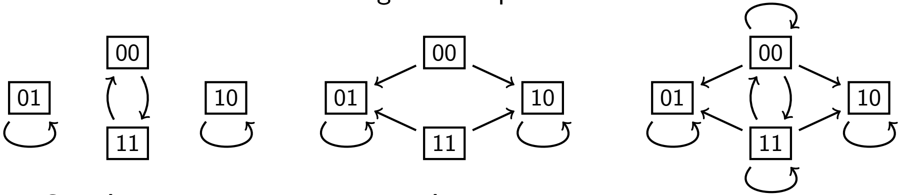
Semantics

$f_a = \neg b$ $\qquad$ $f_b = \neg a$

b	f_a
0	1
1	0

a	f_b
0	1
1	0

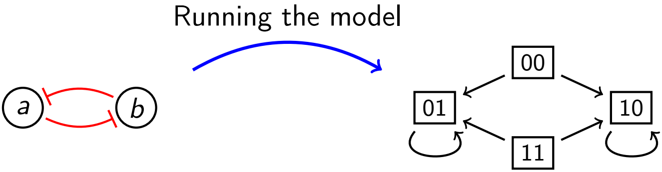
State transitions differ according to the update semantics used:



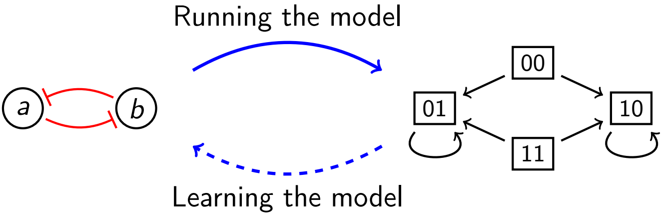
Synchronous **Asynchronous** **General**

- **Synchronous**: all variables are updated
- **Asynchronous**: only one variable is updated
- **General**: any number of variables can be updated

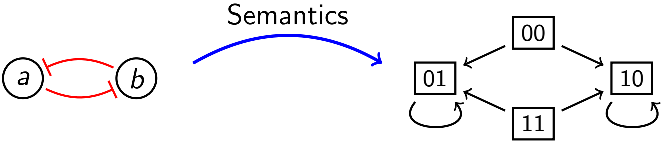
Learning from the State Graph



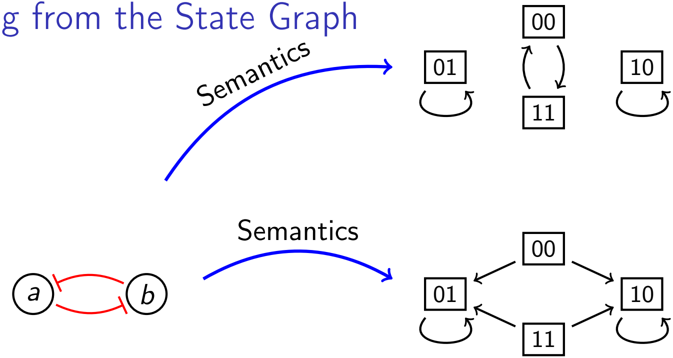
Learning from the State Graph



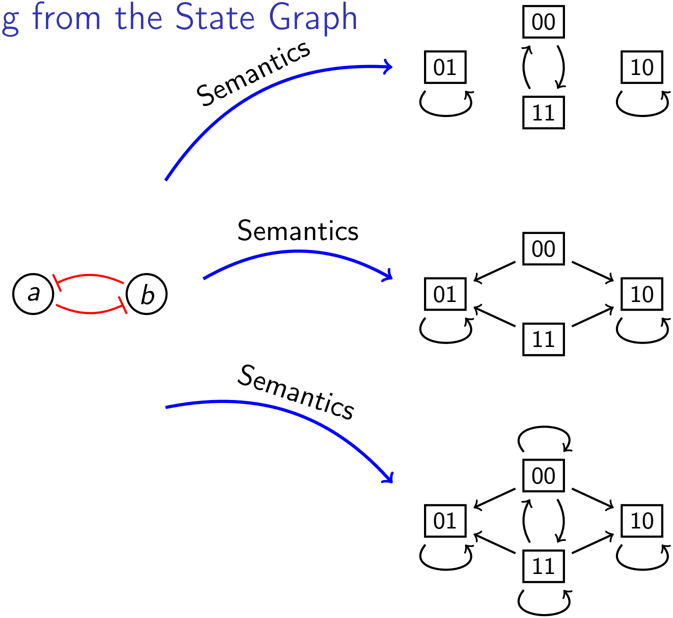
Learning from the State Graph



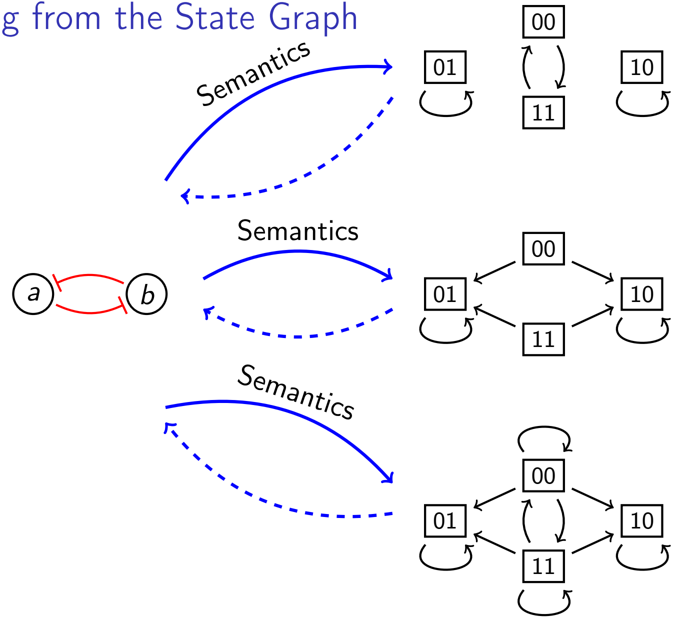
Learning from the State Graph



Learning from the State Graph



Learning from the State Graph



Logic Rules

LFIT learns a logic program, which is a set of logic rules.
It is an alternative representation of biological networks.

$$a_1 \leftarrow a_0, b_0, c_2.$$

If a and b are at level 0 and c is at level 2, then a can change its value to 1.

$$a_1 \leftarrow c_2.$$

Whenever c is at level 2, a can change its value to 1.

$$a_1 \leftarrow .$$

a can change its value to 1 anytime.

Logic Rules

LFIT learns a logic program, which is a set of logic rules.
It is an alternative representation of biological networks.

$$a_1 \leftarrow a_0, b_0, c_2.$$

If a and b are at level 0 and c is at level 2, then a can change its value to 1.

$$a_1 \leftarrow c_2.$$

Whenever c is at level 2, a can change its value to 1.

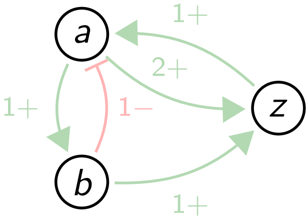
$$a_1 \leftarrow .$$

a can change its value to 1 anytime.

Semantics = From this information, what are the possible next states?

Discrete Models as Logic Programs

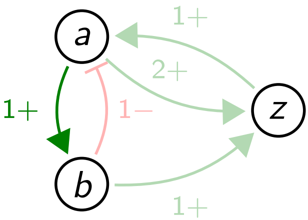
Discrete model:



Logic program:

Discrete Models as Logic Programs

Discrete model:

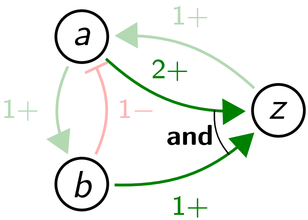


Logic program:

$b_1 \leftarrow a_1.$
$b_1 \leftarrow a_2.$
$b_0 \leftarrow a_0.$

Discrete Models as Logic Programs

Discrete model:



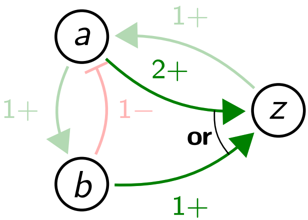
Logic program:

$b_1 \leftarrow a_1.$
$b_1 \leftarrow a_2.$
$b_0 \leftarrow a_0.$

$z_1 \leftarrow a_2 \wedge b_1.$
$z_0 \leftarrow a_0.$
$z_0 \leftarrow a_1.$
$z_0 \leftarrow b_0.$

Discrete Models as Logic Programs

Discrete model:



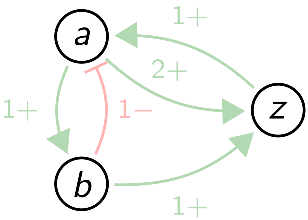
Logic program:

$b_1 \leftarrow a_1.$
$b_1 \leftarrow a_2.$
$b_0 \leftarrow a_0.$

$z_1 \leftarrow a_2.$
$z_1 \leftarrow b_1.$
$z_0 \leftarrow a_1 \wedge b_0.$
$z_0 \leftarrow a_0 \wedge b_0.$

Discrete Models as Logic Programs

Discrete model:



$<$

Expressivity

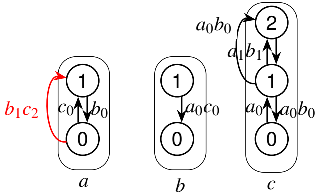
Logic program:

$b_1 \leftarrow a_1.$
$b_1 \leftarrow a_2.$
$b_0 \leftarrow a_0.$

$z_1 \leftarrow a_2.$
$z_1 \leftarrow b_1.$
$z_0 \leftarrow a_1 \wedge b_0.$
$z_0 \leftarrow a_0 \wedge b_0.$

AANs as Logic Programs

Asynchronous automata network:



Picture: [Soh et al., CMSB'2023]

Logic program:

$b_1 \leftarrow a_1.$

$b_1 \leftarrow a_2.$

$b_0 \leftarrow a_0.$

$z_1 \leftarrow a_2.$

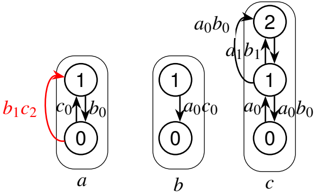
$z_1 \leftarrow b_1.$

$z_0 \leftarrow a_1 \wedge b_0.$

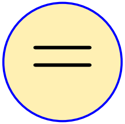
$z_0 \leftarrow a_0 \wedge b_0.$

AANs as Logic Programs

Asynchronous automata network:



Picture: [Soh et al., CMSB'2023]



Expressivity

Logic program:

$b_1 \leftarrow a_1.$

$b_1 \leftarrow a_2.$

$b_0 \leftarrow a_0.$

$z_1 \leftarrow a_2.$

$z_1 \leftarrow b_1.$

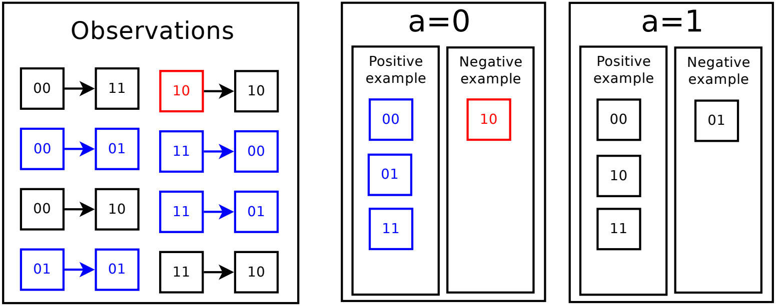
$z_0 \leftarrow a_1 \wedge b_0.$

$z_0 \leftarrow a_0 \wedge b_0.$

Learning From Interpretation Transition (LFIT)

Learning Algorithm Intuition: Classification Problem

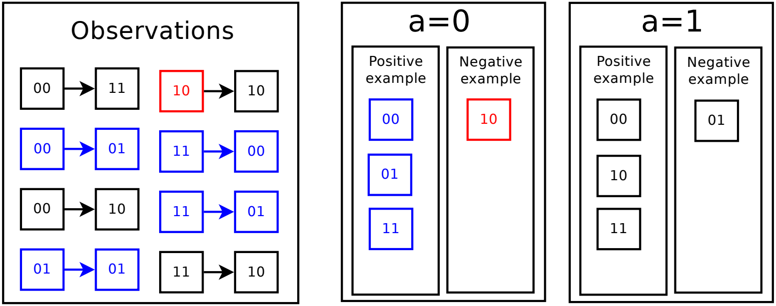
Learn applicable rules: conditions so that a variable **can** take a certain value in next state.



Equivalent to a **classification problem**: What is a typical state where a can take value 0 in the next state ? Here: when a_0 or b_1 is present.

Learning Algorithm Intuition: Classification Problem

Learn applicable rules: conditions so that a variable **can** take a certain value in next state.



Equivalent to a **classification problem**: What is a typical state where a can take value 0 in the next state ? Here: when a_0 or b_1 is present.

That is: $a_0 \leftarrow a_0.$ $a_0 \leftarrow b_1.$

Presentation of GULA

GULA = General Usage LFIT Algorithm

Input: a set of transitions $(s_1 \rightarrow s_2)$

Output: a logic program that respects:

- **Consistency**: the program allows no negative examples
- **Realization**: the program covers all positive examples
- **Completeness**: the program covers all the state space
- **Minimality** of the rules (most general conditions)

Compatible with semantics that use the specifications of the model
This includes the synchronous, asynchronous and general semantics

Method: start from most general rules and **specialize** iteratively

Specialization by Minimal Refinements

Suppose:

- a and b have two levels $\{0, 1\}$ and c has three levels $\{0, 1, 2\}$
- the current program contains the following rules regarding a_1:

$$a_1 \leftarrow c_2. \qquad\qquad a_1 \leftarrow b_1.$$

- from state $\langle a_1, b_0, c_2 \rangle$, a_1 is never observed in the next states.

Minimal refinement to make the rules inapplicable in this state:

Specialization by Minimal Refinements

Suppose:

- a and b have two levels $\{0, 1\}$ and c has three levels $\{0, 1, 2\}$
- the current program contains the following rules regarding a_1:

$$a_1 \leftarrow c_2. \qquad\qquad a_1 \leftarrow b_1.$$

- from state $\langle a_1, b_0, c_2 \rangle$, a_1 is never observed in the next states.

Minimal refinement to make the rules inapplicable in this state:

$$a_1 \leftarrow a_0, c_2.$$
$$a_1 \leftarrow b_1, c_2.$$
$$a_1 \leftarrow c_2, c_0.$$
$$a_1 \leftarrow c_2, c_1.$$

$$a_1 \leftarrow b_1.$$

(No change)

Specialization by Minimal Refinements

Suppose:

- a and b have two levels $\{0, 1\}$ and c has three levels $\{0, 1, 2\}$
- the current program contains the following rules regarding a_1:

$$a_1 \leftarrow c_2. \qquad\qquad a_1 \leftarrow b_1.$$

- from state $\langle a_1, b_0, c_2 \rangle$, a_1 is never observed in the next states.

Minimal refinement to make the rules inapplicable in this state:

$$a_1 \leftarrow a_0, c_2.$$
$$a_1 \leftarrow b_1, c_2.$$
$$a_1 \leftarrow c_2, c_0.$$
$$a_1 \leftarrow c_2, c_1.$$

$$a_1 \leftarrow b_1.$$

Specialization by Minimal Refinements

Suppose:

- a and b have two levels $\{0,1\}$ and c has three levels $\{0,1,2\}$
- the current program contains the following rules regarding a_1:

$$a_1 \leftarrow c_2. \qquad\qquad a_1 \leftarrow b_1.$$

- from state $\langle a_1, b_0, c_2\rangle$, a_1 is never observed in the next states.

Minimal refinement to make the rules inapplicable in this state:

$$a_1 \leftarrow a_0, c_2.$$
$$a_1 \leftarrow b_1, c_2.$$

$$a_1 \leftarrow b_1.$$

(More general)

Specialization by Minimal Refinements

Suppose:

- a and b have two levels $\{0, 1\}$ and c has three levels $\{0, 1, 2\}$
- the current program contains the following rules regarding a_1:

$$a_1 \leftarrow c_2. \qquad\qquad a_1 \leftarrow b_1.$$

- from state $\langle a_1, b_0, c_2 \rangle$, a_1 is never observed in the next states.

Minimal refinement to make the rules inapplicable in this state:

$$a_1 \leftarrow a_0, c_2. \qquad\qquad a_1 \leftarrow b_1.$$

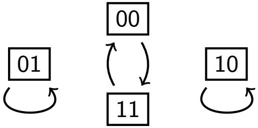
Example: Synchronous Semantics

$f_a = \neg b$

b	f_a
0	1
1	0

$f_b = \neg a$

a	f_b
0	1
1	0



Synchronous

$// \ f_a = \neg b$

$a_0 \leftarrow b_1$

$a_1 \leftarrow b_0$

$// \ f_b := \neg a$

$b_0 \leftarrow a_1$

$b_1 \leftarrow a_0$

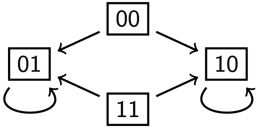
Example: Asynchronous Semantics

$f_a = \neg b$

b	f_a
0	1
1	0

$f_b = \neg a$

a	f_b
0	1
1	0



Asynchronous

// $f_a = \neg b$

$a_0 \leftarrow b_1$

$a_1 \leftarrow b_0$

// $f_b = \neg a$

$b_0 \leftarrow a_1$

$b_1 \leftarrow a_0$

// Default rules

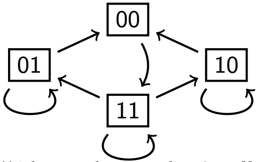
$a_0 \leftarrow a_0$

$a_1 \leftarrow a_1$

$b_0 \leftarrow b_0$

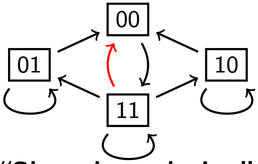
$b_1 \leftarrow b_1$

Learning the semantics with constraints



“Slow degradation”

Learning the semantics with constraints



"Slow degradation"

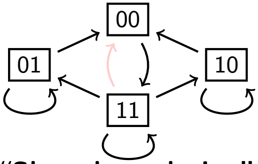
// $f_a = \neg b$
$a_0^t \leftarrow b_1^{t-1}.$
$a_1^t \leftarrow b_0^{t-1}.$

// $f_b = \neg a$
$b_0^t \leftarrow a_1^{t-1}.$
$b_1^t \leftarrow a_0^{t-1}.$

// Conservation rules
$a_1^t \leftarrow a_1^{t-1}.$
$b_1^t \leftarrow b_1^{t-1}.$

// Degradation
$a_0^t \leftarrow a_1^{t-1}.$
$b_0^t \leftarrow b_1^{t-1}.$

Learning the semantics with constraints



“Slow degradation”

// $f_a = \neg b$
$a_0^t \leftarrow b_1^{t-1}.$
$a_1^t \leftarrow b_0^{t-1}.$

// $f_b = \neg a$
$b_0^t \leftarrow a_1^{t-1}.$
$b_1^t \leftarrow a_0^{t-1}.$

// Conservation rules
$a_1^t \leftarrow a_1^{t-1}.$
$b_1^t \leftarrow b_1^{t-1}.$

// Degradation
$a_0^t \leftarrow a_1^{t-1}.$
$b_0^t \leftarrow b_1^{t-1}.$

// Constraints
$\bot \leftarrow a_0^t, b_0^t, a_1^{t-1}, b_1^{t-1}.$

Results

- **Algorithm that allows to learn the network**
  - Structure of the model
  - Under the form of a logic program
- **Directly works for a class of memoryless semantics**
  - Characterization of applicable semantics
- Usable to learn from other semantics as well
  - By using constraints to learn the semantics along with the model

Limitations:

- Exponential complexity
- What if the data is **incomplete** or **noisy**?

A Heuristic on LFIT

Weighted Likeliness/Unlikeliness Rules

- Use the algorithm twice to learn two logic programs:
  - likeliness rules: what is possible
  - unlikeliness rules: what is impossible
- Weight each rule by the number of observations it matches

Statistical overlay $\Rightarrow$ usable on **noisy datasets**

Likeliness rules	**Unlikeliness rules**
$(3, a_0 \leftarrow b_1)$	$(30, a_0 \leftarrow c_1)$
$(15, a_1 \leftarrow b_0)$	$(5, a_1 \leftarrow c_0)$
$\vdots$	$\vdots$

Using Weighted Likeliness/Unlikeliness Rules

Explainable predictions:

- Compare weights of applicable likeliness/unlikeliness rules
- Ratio of highest weights $\Rightarrow$ **probability** P
- Rules with highest weights $\Rightarrow$ **explanation** E

$$\text{predict} : (atom, state) \mapsto (P, E)$$

Likeliness rules	**Unlikeliness rules**
$(3, a_0 \leftarrow b_1)$	$(30, a_0 \leftarrow c_1)$
$(15, a_1 \leftarrow b_0)$	$(5, a_1 \leftarrow c_0)$

Using Weighted Likeliness/Unlikeliness Rules

Explainable predictions:

- Compare weights of applicable likeliness/unlikeliness rules
- Ratio of highest weights ⇒ **probability** P
- Rules with highest weights ⇒ **explanation** E

$$\text{predict} : (atom, state) \mapsto (P, E)$$

Likeliness rules

$(3, a_0 \leftarrow b_1)$

$(15, a_1 \leftarrow b_0)$

Unlikeliness rules

$(30, a_0 \leftarrow c_1)$

$(5, a_1 \leftarrow c_0)$

$\text{predict}(a_1, \langle a_1, b_0, c_0 \rangle) = (0.75, ((15, a_1 \leftarrow b_0), (5, a_1 \leftarrow c_0))) \Rightarrow$ Likely

Using Weighted Likeliness/Unlikeliness Rules

Explainable predictions:

- Compare weights of applicable likeliness/unlikeliness rules
- Ratio of highest weights $\Rightarrow$ **probability** P
- Rules with highest weights $\Rightarrow$ **explanation** E

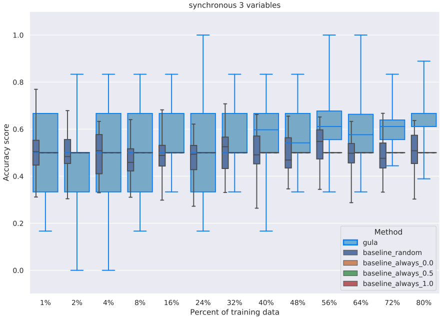
$$\text{predict} : (atom, state) \mapsto (P, E)$$

Likeliness rules	**Unlikeliness rules**
$(3, a_0 \leftarrow b_1)$	$(30, a_0 \leftarrow c_1)$
$(15, a_1 \leftarrow b_0)$	$(5, a_1 \leftarrow c_0)$

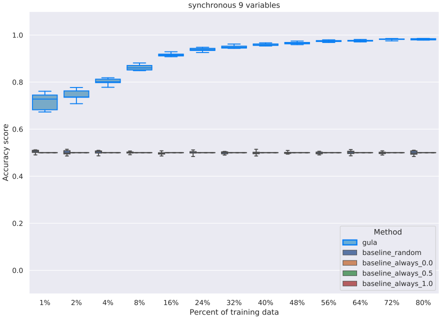
$\text{predict}(a_1, \langle a_1, b_0, c_0 \rangle) = (0.75, ((15, a_1 \leftarrow b_0), (5, a_1 \leftarrow c_0))) \Rightarrow$ Likely

$\text{predict}(a_0, \langle a_1, b_1, c_1 \rangle) = (0.09, ((3, a_0 \leftarrow b_1), (30, a_0 \leftarrow c_1))) \Rightarrow$ Unlikely

Prediction power



3 variables



9 variables

Training data = X% of transitions
Tested against unseen states (not in the training data)

Conclusion

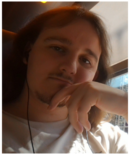
Conclusion

- **Learn** the network with LFIT (theory)
  - When not "learnable", learn the semantics with constraints
- **Heuristics** to tackle real data (practice)
  - Good results with 10% of the transitions

Outlooks:

- PRIDE: polynomial algorithm that "misses" some explanations
- Learn from the Most Permissive semantics
- Application to real data (marine phytoplankton)

Thanks



Tony
RIBEIRO

Morgan
MAGNIN

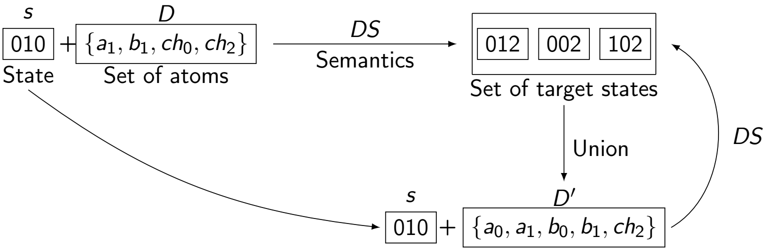
Katsumi
INOUE

Bibliography

- About GULA: Tony Ribeiro, Maxime Folschette, Morgan Magnin and Katsumi Inoue. **Learning any memory-less discrete semantics for dynamical systems represented by logic programs**. *Machine Learning* 111, Springer. November 2021. `https://doi.org/10.1007/s10994-021-06105-4`
- pyLFIT Python library: `https://github.com/Tony-sama/pylfit`
- About PRIDE: Tony Ribeiro, Maxime Folschette, Morgan Magnin and Katsumi Inoue. **Polynomial Algorithm For Learning From Interpretation Transition**. Poster at the *1st International Joint Conference on Learning & Reasoning*. October 2021, Online. `https://hal.science/hal-03347026v1`
- Application to phytoplankton: Omar Iken, Maxime Folschette and Tony Ribeiro. **Automatic Modeling of Dynamical Interactions Within Marine Ecosystems**. Poster in the *1st International Joint Conference on Learning & Reasoning*. October 2021, Online. `https://hal.science/hal-03347033v1`

Pseudo-idempotent semantics

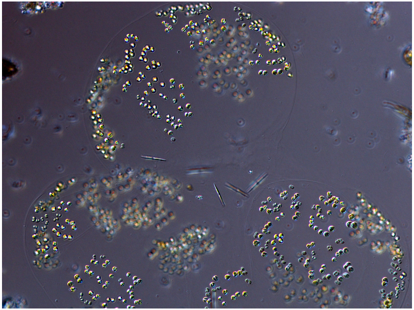
GULA can model observations from any pseudo-idempotent semantics.

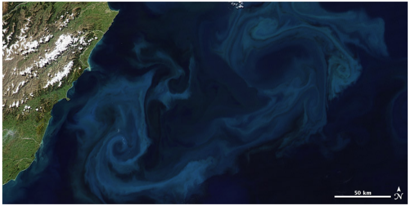


$$\longrightarrow DS(s, D) = DS\Big(s, \bigcup_{s' \in DS(s,D)} s'\Big)$$

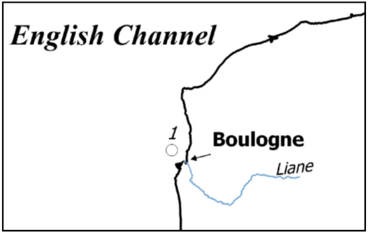
where DS is the dynamical semantics, and D is set of heads of rules of a multi-valued logic program that match the sate s.

Phytoplankton Blooms





SRN Dataset



https://www.seanoe.org/data/00397/50832/

Sampling location	Sampling date	Taxon	Value	Sampling depth
001-P-015	1992-05-18	CHLOROA	6.0	Surface (0-1m)
006-P-001	2019-12-02	Chaetoceros	1000.0	Surface (0-1m)
002-P-007	1994-05-25	Pleurosigma	100.0	Surface (0-1m)
002-P-030	2005-10-19	SALI	34.83	Surface (0-1m)
006-P-007	2015-09-28	Guinardia delicatula	11400.0	Surface (0-1m)

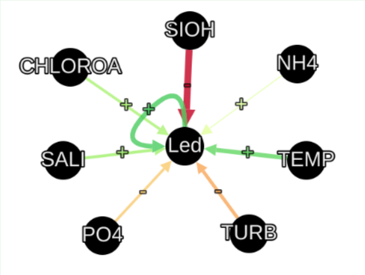
Environmental variables (7) Phytoplankton species (12)

Global Influences

Process: Search and count patterns in rules that characterize an activation/inhibition
Hypotheses: Monotonous influences & same threshold for all variables
Result: Score $[-1;+1]$ between each pair of variables (no threshold)

Influences on phytoplankton specie `Led`:

Variable	Positive	Negative	Global
`PO4`	$+0$	-58	-0.36
`SALI`	$+71$	-4	$+0.42$
`CHLOROA`	$+84$	-22	$+0.39$
`SIOH`	$+3$	-161	-0.98
`NH4`	$+25$	-5	$+0.12$
`TEMP`	$+106$	-5	$+0.63$
`TURB`	$+10$	-87	-0.48



$$\text{global\_influence}(\texttt{PO4} \to \texttt{Led}) = \frac{+0 + (-58)}{161} = -0.36$$