Séminaire des nouveaux arrivants MSV — 2020-01-07

Analysis and Learning of Dynamical Biological Networks: A Summary of my Works

Analyse et apprentissage de réseaux biologiques dynamiques :
un résumé de mes travaux

Maxime FOLSCHETTE

maxime.folschette@centralelille.fr
http://maxime.folschette.name/

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

Université de Lille
cnrs
centralelille
BioComputing

Introduction & Résumé

Analysis of the Dynamics

- **Centrale Nantes** — PhD thesis — 2011–2014
  - → **Efficient reachability analysis on large networks**
  - → **Dynamical patterns enumeration with answer set programming**
- **Univ Kassel** — postdoc — 2014–2015
  - → Complex patterns enumeration with polyadic μ-calculus

Learning Models from Data

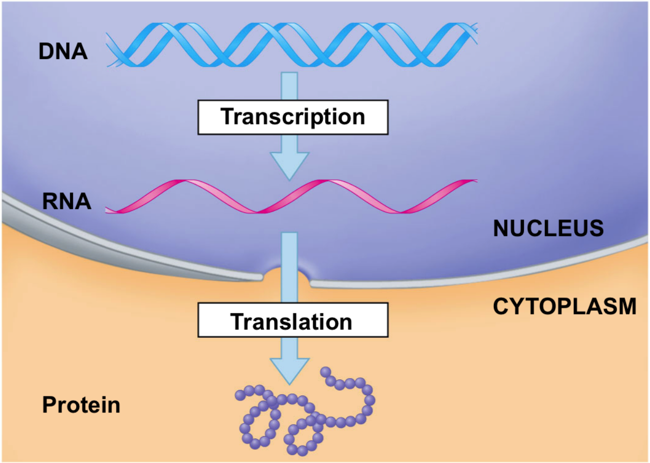
- **Univ Nice** — ATER — 2015–2016
  - → Inference of constraints on hybrid parameters
- **Univ Nantes** — ATER — 2016–2017
  - → **Learning models from time series data**

Learning New Knowledge from Models

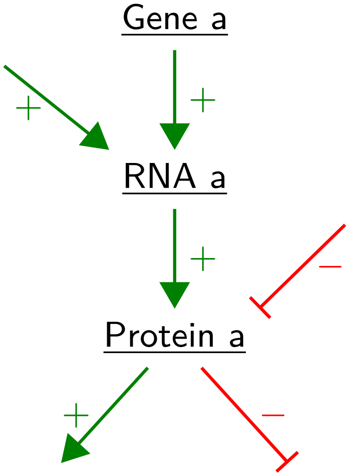
- **Univ Rennes** — postdoc — 2017–2018
  - → Computational model to study hepatocellular carcinoma progression
- **CNRS/LS2N** — postdoc — 2018–2019
  - → Integrate heterogeneous clinical, genetic, imaging data with semantic web in order to learn variables of interest
- **Centrale Lille** — **maître de conférences** — 2019 ⋮

Frameworks

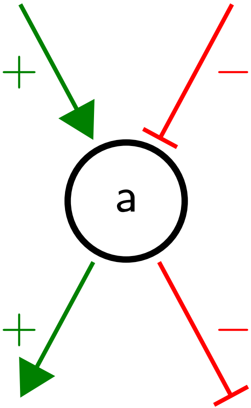
Preliminary Abstraction



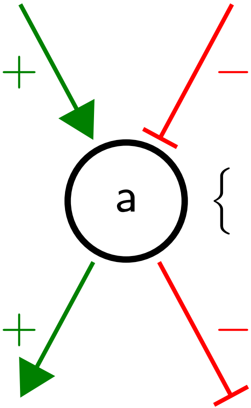
Preliminary Abstraction



Preliminary Abstraction

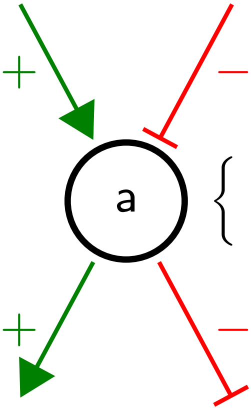


Preliminary Abstraction



$$\left\{ \begin{array}{l} 0 = \text{inactive} \\ 1 = \text{active} \end{array} \right\}$$

Preliminary Abstraction

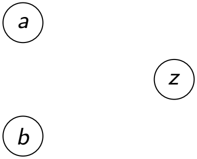


$$\left\{ \begin{array}{l} 0 = \text{complete degradation} \\ 1 = \text{traces} \\ 2 = \text{saturation} \end{array} \right\}$$

Discrete Networks / Thomas Modeling

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

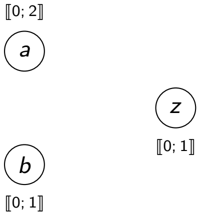
- A set of components $N = \{a, b, z\}$



Discrete Networks / Thomas Modeling

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

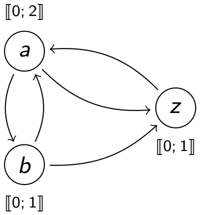
- A set of components $N = \{a, b, z\}$
- A discrete domain for each component $\text{dom}(a) = [\![0; 2]\!]$



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 $\text{dom}(a) = [\![0; 2]\!]$
- Discrete parameters / evolution functions $f^a : \mathcal{S} \to \text{dom}(a)$



a	f^b
0	**0**
1	**1**
2	**1**

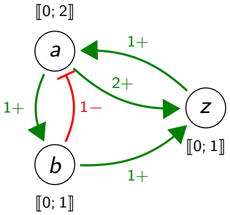
z	b	f^a
0	0	**1**
0	1	**0**
1	0	**1**
1	1	**2**

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**

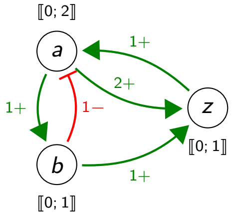
z	b	f^a
0	0	**1**
0	1	**0**
1	0	**1**
1	1	**2**

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 $\text{dom}(a) = [\![0; 2]\!]$
- 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**

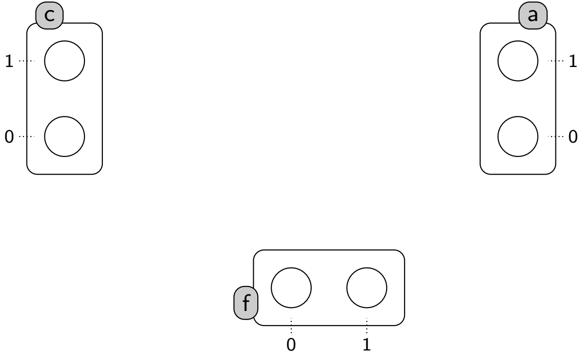
z	b	f^a
0	0	**1**
0	1	**0**
1	0	**1**
1	1	**2**

a	b	f^z
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**0**
2	0	**0**
2	1	**1**

Semantics = From this information, what is (are) the next state(s)?

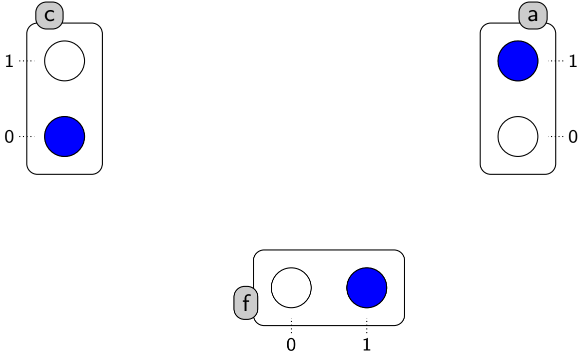
Asynchronous Automata Networks (AAN)

[Paulevé *et al.*, *Transactions on Computational Systems Biology*, 2011]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]



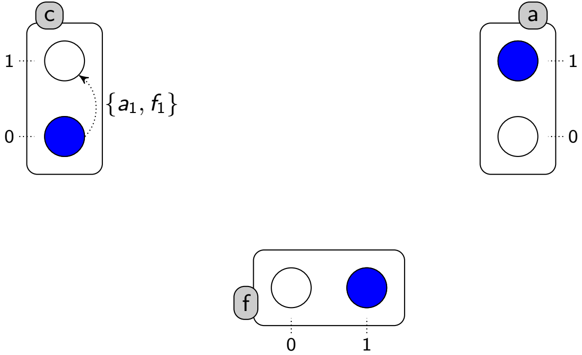
Asynchronous Automata Networks (AAN)

[Paulevé *et al.*, *Transactions on Computational Systems Biology*, 2011]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]



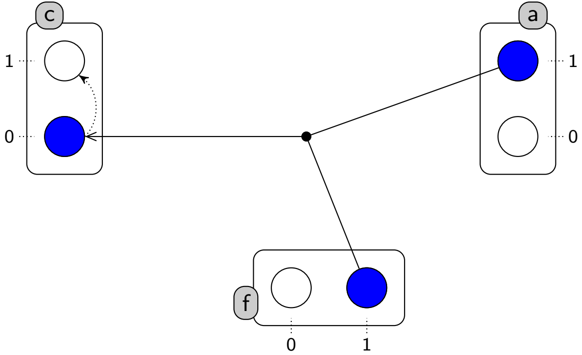
Asynchronous Automata Networks (AAN)

[Paulevé *et al.*, *Transactions on Computational Systems Biology*, 2011]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]



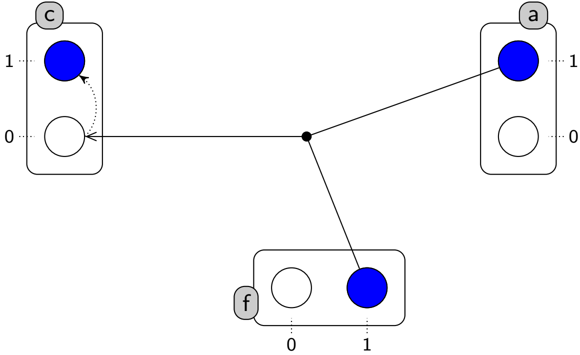
Asynchronous Automata Networks (AAN)

[Paulevé *et al.*, *Transactions on Computational Systems Biology*, 2011]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]



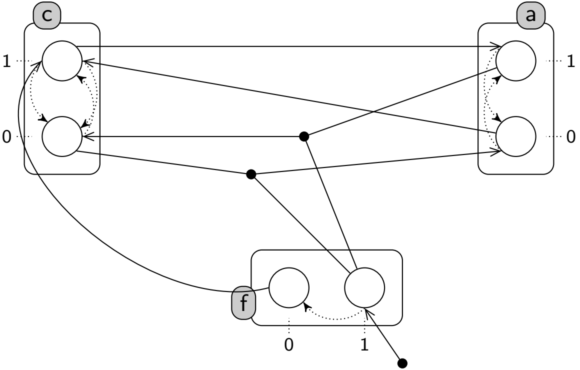
Asynchronous Automata Networks (AAN)

[Paulevé *et al.*, *Transactions on Computational Systems Biology*, 2011]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]



Asynchronous Automata Networks (AAN)

[Paulevé *et al.*, *Transactions on Computational Systems Biology*, 2011]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]
Model from [François *et al.*, *Molecular Systems Biology*, 2007]



Semantics = How to combine actions to compute the next state(s)?

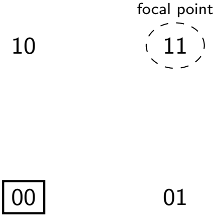
Semantics

10 11

00 01

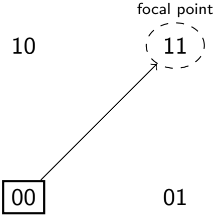
Synchronous Asynchronous Generalized

Semantics



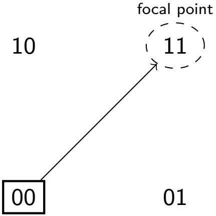
Synchronous Asynchronous Generalized

Semantics

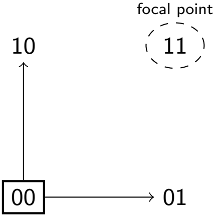


Synchronous **Asynchronous** **Generalized**

Semantics



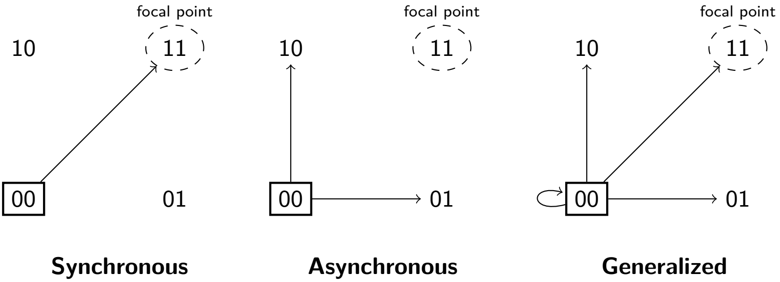
Synchronous



Asynchronous

Generalized

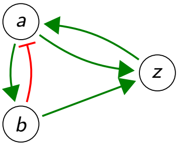
Semantics


focal point
10
11
00
01
Synchronous
focal point
10
11
00
01
Asynchronous
focal point
10
11
00
01
Generalized

State-graph

The state-graph depicts explicitly the whole dynamics

abz			
000	010	001	011
100	110	101	111
200	210	201	211

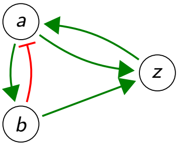


State-graph

The state-graph depicts explicitly the whole dynamics

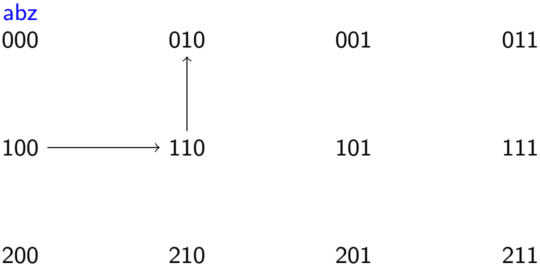
abz

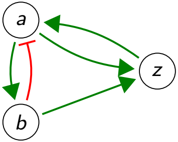
000	010	001	011
100 ⟶	110	101	111
200	210	201	211



State-graph

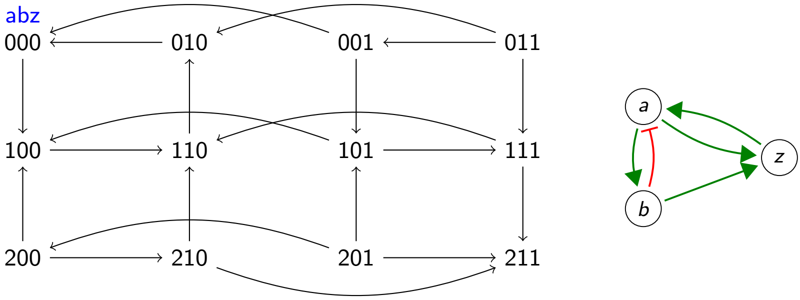
The state-graph depicts explicitly the whole dynamics





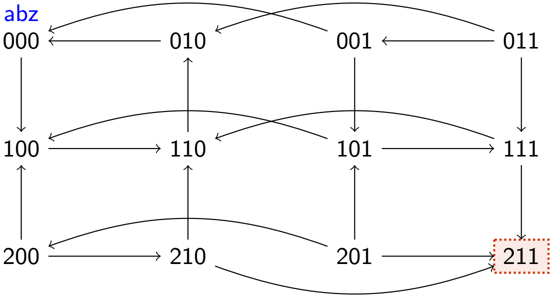
State-graph

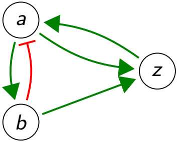
The state-graph depicts explicitly the whole dynamics



State-graph

The state-graph depicts explicitly the whole dynamics

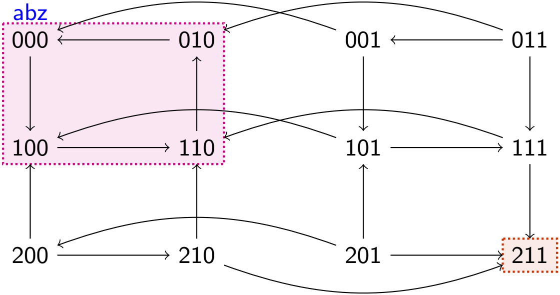


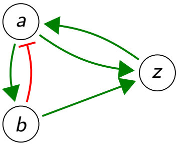


- **Stable state** = state with no successors

State-graph

The state-graph depicts explicitly the whole dynamics

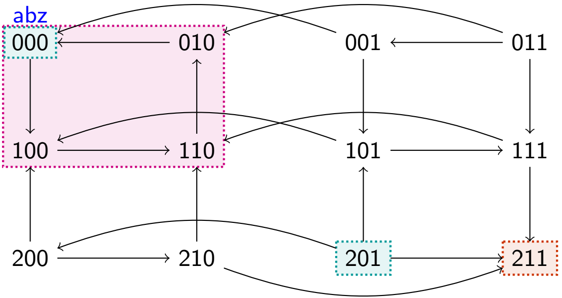


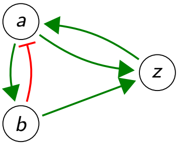


- **Stable state** = state with no successors
- **Complex attractor** = minimal loop or composition of loops from which the dynamics cannot escape

State-graph

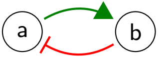
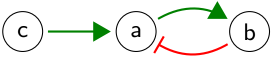
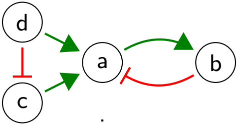
The state-graph depicts explicitly the whole dynamics





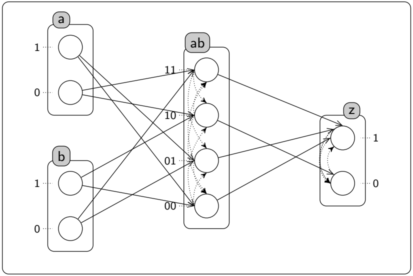
- **Stable state** = state with no successors
- **Complex attractor** = minimal loop or composition of loops from which the dynamics cannot escape
- **Reachability** = from **201**, can I reach **000**?

Combinatorial explosion

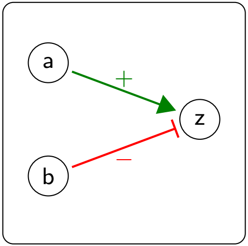
Model	Possible states
	4
	8
	16
⋮	⋮
(10)	1024
(20)	1048576
(100)	1267650600000000000000000000000

Translation of AAN models

[Folschette *et al.*, *Computational Methods in Systems Biology*, 2012]



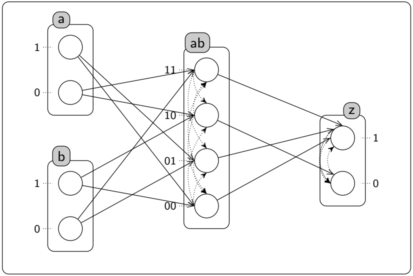
Before: **Process Hitting**
Efficient but recent



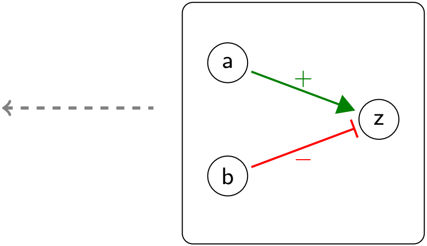
Thomas modeling
Widespread & readable

Translation of AAN models

[Folschette *et al.*, *Computational Methods in Systems Biology*, 2012]



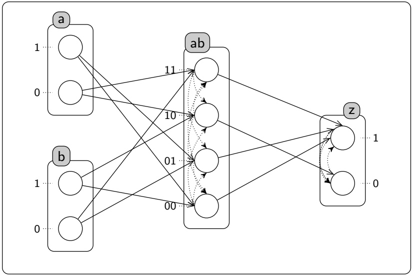
Before: **Process Hitting**
Efficient but recent



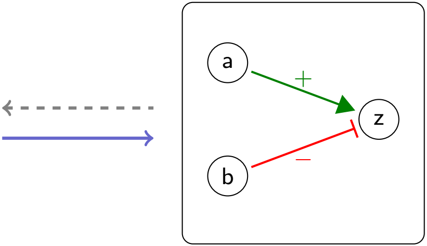
Thomas modeling
Widespread & readable

Translation of AAN models

[Folschette *et al.*, *Computational Methods in Systems Biology*, 2012]



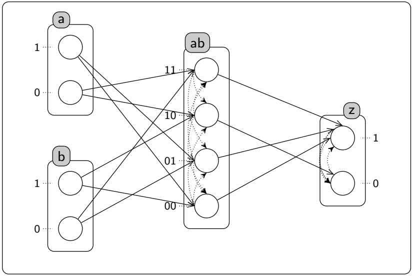
Before: **Process Hitting**
Efficient but recent

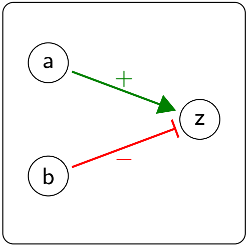


Thomas modeling
Widespread & readable

Towards AANs

[Folschette *et al.*, *CS2Bio'13*, 2013]



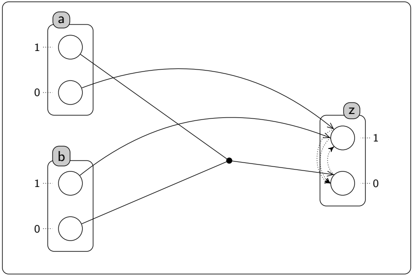


Before: **Process Hitting**
Loose behavior

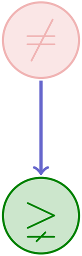
Thomas modeling
Expected behavior

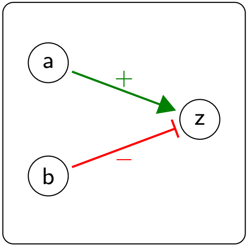
Towards AANs

[Folschette *et al.*, *CS2Bio'13*, 2013]



Now: **AANs**
Accurate behavior





Thomas modeling
Expected behavior

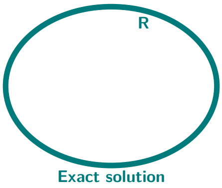
Analysis of the Dynamics

Efficient reachability analysis on large networks

Approximations of the Dynamics

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]

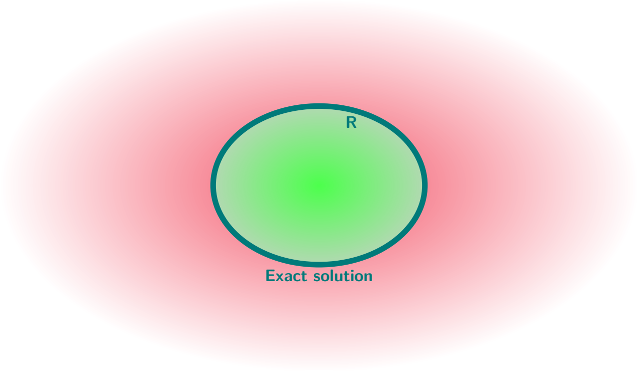
- Directly checking **R** is hard (**exponential**)
- Rather check **approximations** **P** and **Q** so that: **P** $\Rightarrow$ **R** $\Rightarrow$ **Q**
  so that computing **P** and **Q** is faster (roughly **polynomial**)



Approximations of the Dynamics

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]

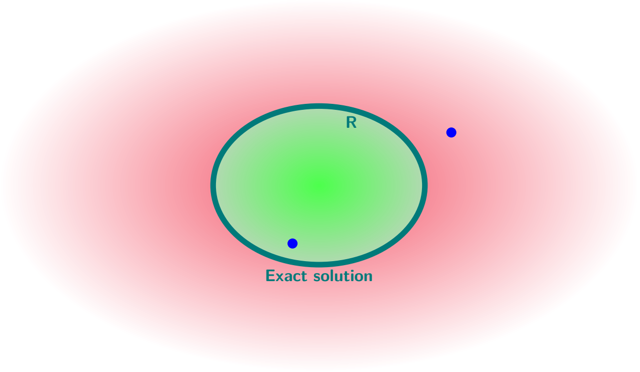
- Directly checking **R** is hard (**exponential**)
- Rather check **approximations** **P** and **Q** so that: **P** ⇒ **R** ⇒ **Q**
  so that computing **P** and **Q** is faster (roughly **polynomial**)



Approximations of the Dynamics

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]

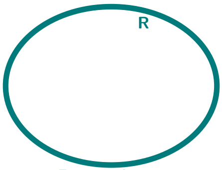
- Directly checking **R** is hard (**exponential**)
- Rather check **approximations P** and **Q** so that: **P** ⇒ **R** ⇒ **Q**
  so that computing **P** and **Q** is faster (roughly **polynomial**)



Approximations of the Dynamics

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]

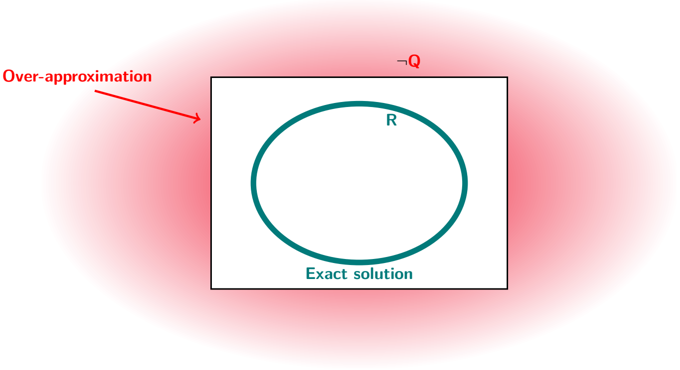
- Directly checking **R** is hard (**exponential**)
- Rather check **approximations** **P** and **Q** so that: **P** ⇒ **R** ⇒ **Q**
  so that computing **P** and **Q** is faster (roughly **polynomial**)



Approximations of the Dynamics

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]

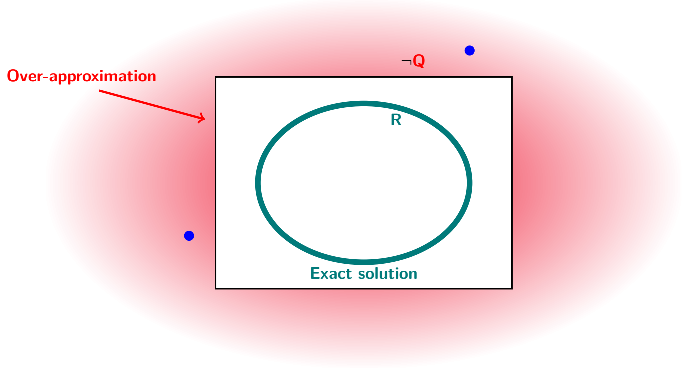
- Directly checking **R** is hard (**exponential**)
- Rather check **approximations** **P** and **Q** so that: **P** $\Rightarrow$ **R** $\Rightarrow$ **Q**
  so that computing **P** and **Q** is faster (roughly **polynomial**)



Approximations of the Dynamics

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]

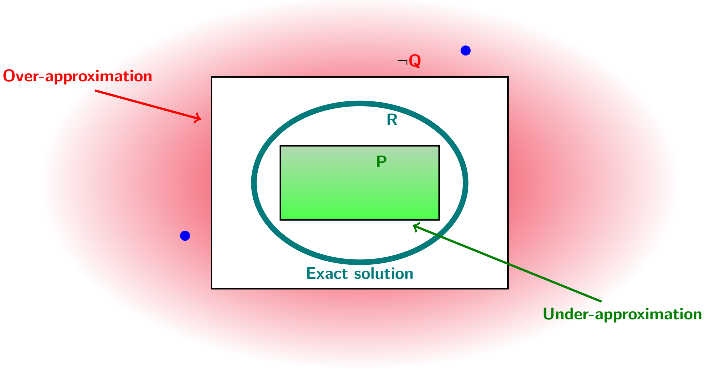
- Directly checking **R** is hard (**exponential**)
- Rather check **approximations** **P** and **Q** so that: **P** ⇒ **R** ⇒ **Q**
  so that computing **P** and **Q** is faster (roughly **polynomial**)



Approximations of the Dynamics

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]

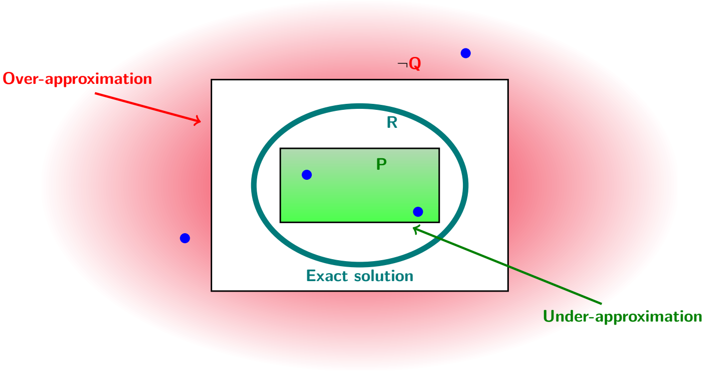
- Directly checking **R** is hard (**exponential**)
- Rather check **approximations** **P** and **Q** so that: **P** ⇒ **R** ⇒ **Q**
  so that computing **P** and **Q** is faster (roughly **polynomial**)



Approximations of the Dynamics

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]

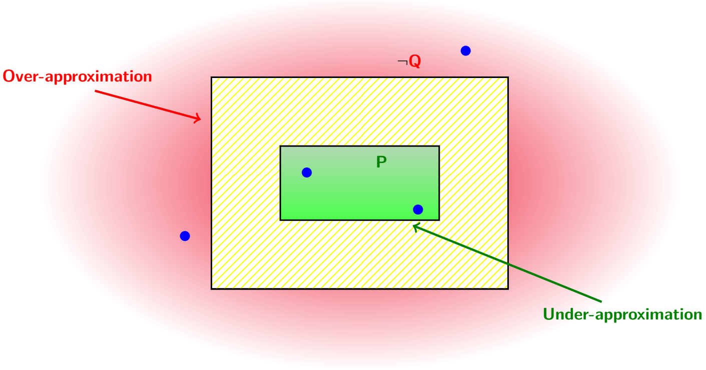
- Directly checking **R** is hard (**exponential**)
- Rather check **approximations** **P** and **Q** so that: **P** ⇒ **R** ⇒ **Q**
  so that computing **P** and **Q** is faster (roughly **polynomial**)



Approximations of the Dynamics

[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]

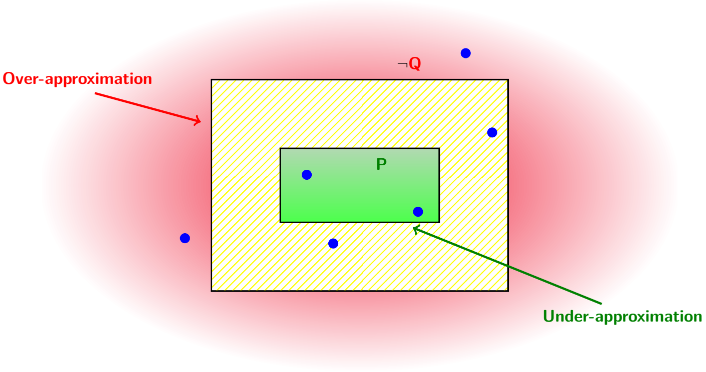
- Directly checking **R** is hard (**exponential**)
- Rather check **approximations** **P** and **Q** so that: **P** ⇒ **R** ⇒ **Q**
  so that computing **P** and **Q** is faster (roughly **polynomial**)



Approximations of the Dynamics

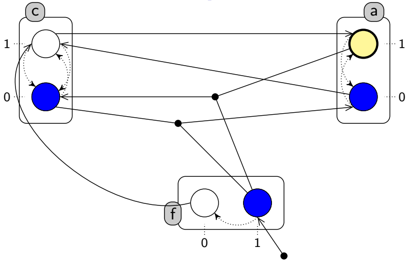
[Paulevé *et al.*, *Mathematical Structures in Computer Science*, 2012]
[Folschette *et al.*, *Theoretical Computer Science*, 2015a]

- Directly checking **R** is hard (**exponential**)
- Rather check **approximations** **P** and **Q** so that: **P** ⇒ **R** ⇒ **Q**
  so that computing **P** and **Q** is faster (roughly **polynomial**)



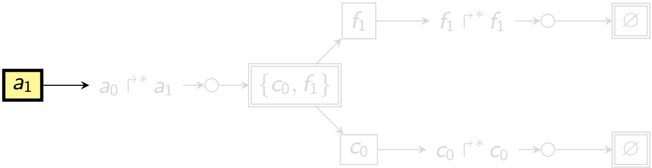
Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



Sufficient condition:

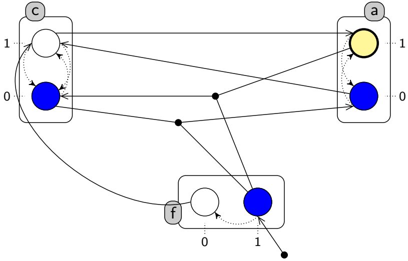
- No cycle
- No conflict
- All leaves are $\emptyset$



P is true $\Rightarrow$ **R** is true

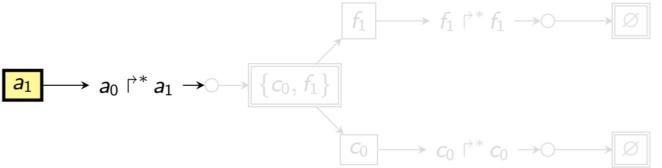
Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



Sufficient condition:

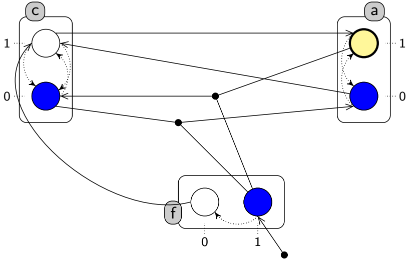
- No cycle
- No conflict
- All leaves are ∅



P is true ⇒ **R** is true

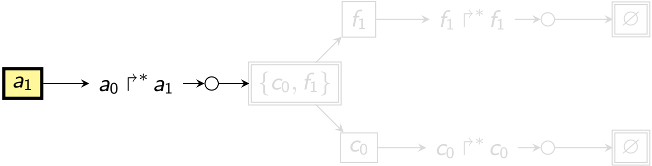
Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



Sufficient condition:

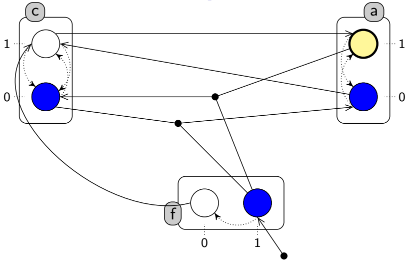
- No cycle
- No conflict
- All leaves are $\varnothing$



P is true $\Rightarrow$ **R** is true

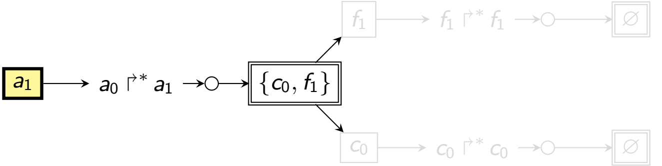
Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



Sufficient condition:

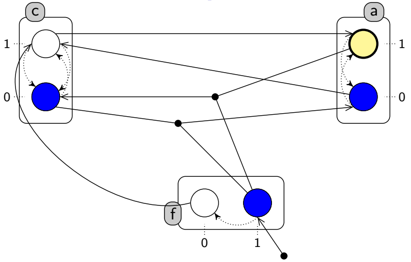
- No cycle
- No conflict
- All leaves are $\boxed{\varnothing}$



P is true ⇒ **R** is true

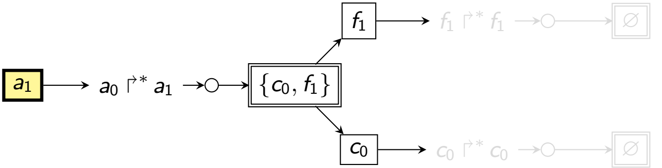
Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



Sufficient condition:

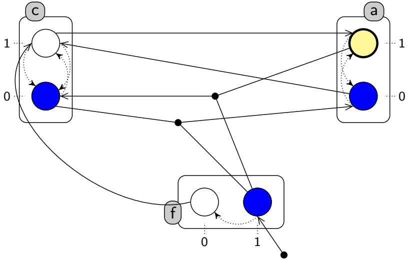
- No cycle
- No conflict
- All leaves are ∅



P is true ⇒ **R** is true

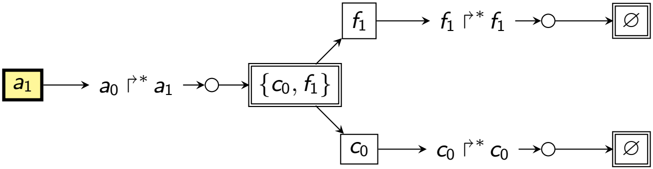
Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



Sufficient condition:

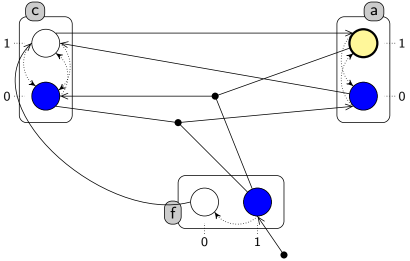
- No cycle
- No conflict
- All leaves are $\varnothing$



P is true $\Rightarrow$ **R** is true

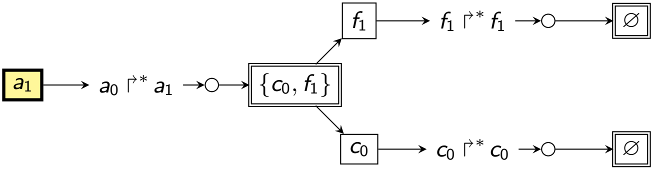
Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



Sufficient condition:

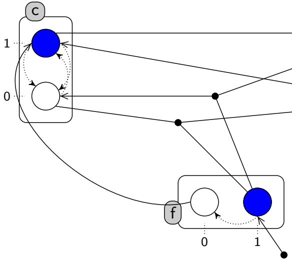
- No cycle
- No conflict
- All leaves are $\varnothing$



P is true $\Rightarrow$ **R** is true

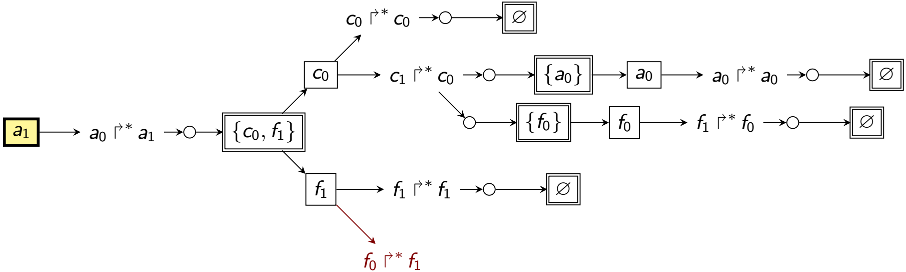
Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



Sufficient condition:

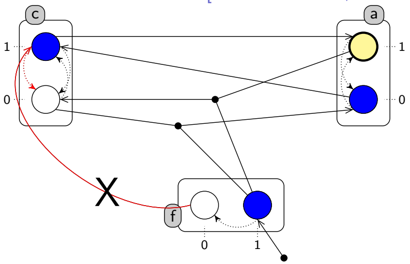
- No cycle
- No conflict
- ~~All leaves are ∅~~



P is false ⇒ Cannot conclude

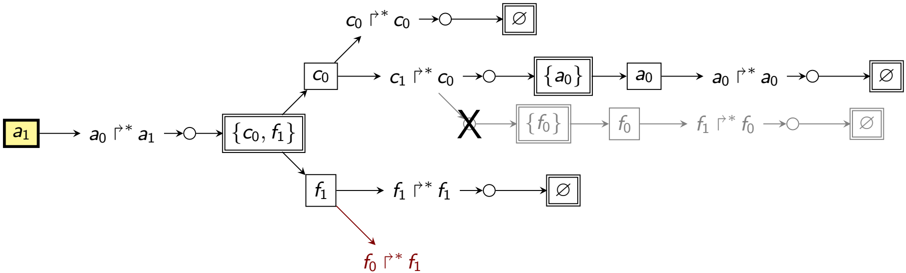
Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



Sufficient condition:

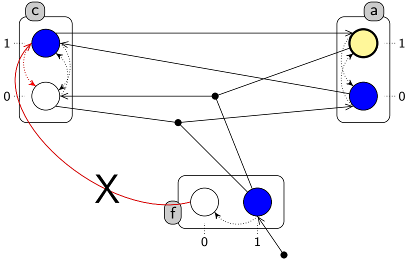
- No cycle
- No conflict
- ~~All leaves are ∅~~



P is false ⇒ Cannot conclude

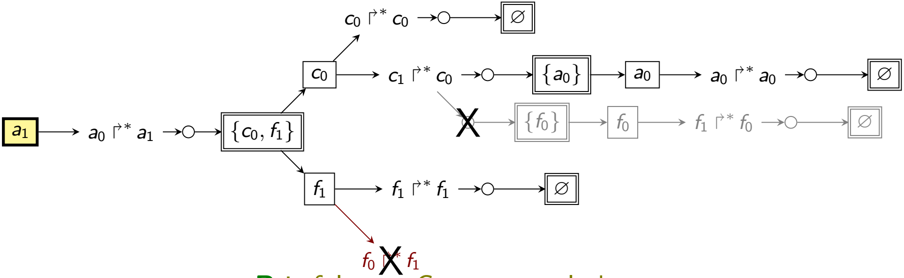
Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



Sufficient condition:

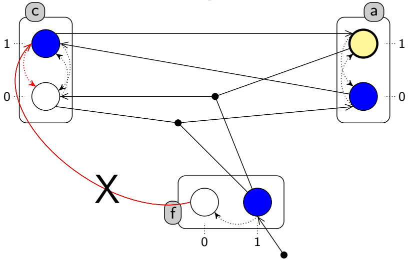
- No cycle
- No conflict
- All leaves are $\varnothing$



P is false ⇒ Cannot conclude

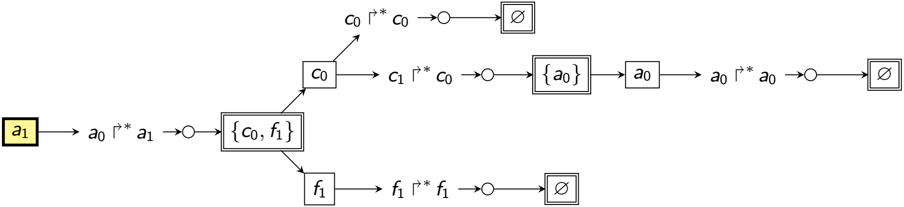
Abstract Interpretation (Under-approximation)

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]



Sufficient condition:

- No cycle
- No conflict
- All leaves are $\boxed{\varnothing}$



P is true ⇒ **R** is true

Implementation of the Abstract Interpretation

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]

Complexity:

- Computation of the local causality graph:
  - Polynomial in the number of automata
  - Exponential in the number of local states of each automata (usually low)
- Check of the sufficient condition:
  - Polynomial in the size of the abstract graph
- Enumeration of the subsets of solutions, if needed:
  - Exponential in the size of the abstract graph

Very efficient on biological networks

Model	Automata	Actions	States	libddd[1]	GINsim[2]	PINT[3]
egfr20	35	670	2^{64}		<1s	**0.02s**
tcrsig40	54	301	2^{73}		∞	**0.02s**
tcrsig94	133	1124	2^{194}	[>1min – ∞]		**0.03s**
egfr104	193	2356	2^{320}	[>1min – ∞]		**0.16s**

[1] LIP6/Move [Couvreur *et al.*, *Lecture Notes in Computer Science*, 2002]
[2] TAGC/IGC [Chaouiya, Naldi, Thieffry, *Methods in Molecular Biology*, 2012]
[3] Loïc Paulevé [Paulevé, *Computational Methods for Systems Biology*, 2017]

egfr20 : Epithelial Growth Factor Receptor (20 components) [Sahin *et al.*, 2009]
egfr104 : Epithelial Growth Factor Receptor (104 components) [Samaga *et al.*, 2009]
tcrsig40 : T-Cell Receptor (40 components) [Klamt *et al.*, 2006]
tcrsig94 : T-Cell Receptor (94 components) [Saez-Rodriguez *et al.*, 2007]

Implementation of the Abstract Interpretation

[Folschette *et al.*, *Theoretical Computer Science*, 2015b]

Complexity:

- Computation of the local causality graph:
  - Polynomial in the number of automata
  - Exponential in the number of local states of each automata (usually low)
- Check of the sufficient condition:
  - Polynomial in the size of the abstract graph
- Enumeration of the subsets of solutions, if needed:
  - Exponential in the size of the abstract graph

Very efficient on biological networks

Model	Automata	Actions	States	libddd[1]	GINsim[2]	PINT[3]
egfr20	35	670	2^{64}		$<$1s	**0.02s**
tcrsig40	54	301	2^{73}		∞	**0.02s**
tcrsig94	133	1124	2^{194}	[$>$1min – ∞]		**0.03s**
egfr104	193	2356	2^{320}	[$>$1min – ∞]		**0.16s**

[1] LIP6/Move [Couvreur *et al.*, *Lecture Notes in Computer Science*, 2002]
[2] TAGC/IGC [Chaouiya, Naldi, Thieffry, *Methods in Molecular Biology*, 2012]
[3] Loïc Paulevé [Paulevé, *Computational Methods for Systems Biology*, 2017]

egfr20 : Epithelial Growth Factor Receptor (20 components) [Sahin *et al.*, 2009]
egfr104 : Epithelial Growth Factor Receptor (104 components) [Samaga *et al.*, 2009]
tcrsig40 : T-Cell Receptor (40 components) [Klamt *et al.*, 2006]
tcrsig94 : T-Cell Receptor (94 components) [Saez-Rodriguez *et al.*, 2007]

Analysis of the Dynamics

Dynamical Patterns Enumeration with Answer Set Programming

Using Answer Set Programming for Model-Checking

Useful when:

- The abstract method is inconclusive
- Looking for complex patterns (attractors)
- Using a different update dynamics (synchronous)

Idea: Go back to an exhaustive analysis, but with heuristics
⇒ Answer Set Programming
⇒ Clingo grounder + solver (Potassco project)

Approach:

1) Describe the problem
2) Enumerate all candidate solutions
3) Filter out unwanted results (not part of the final solution)

Answer Set Programming Concepts

Answer Set Programming (ASP): Declarative & logic programming

Rule: $\underbrace{A_0}_{\text{head}} \leftarrow \underbrace{A_1,\ \dots,\ A_n,\ \texttt{not}\ A_{n+1},\ \dots,\ \texttt{not}\ A_m}_{\text{body}}.$

- $\texttt{not}\ A_i$ is true if there is no proof of A_i (negation by failure)
- If *body* is true, then *head* must be true (logical consequence)
- We search for minimal answer sets (there can be 0, 1, many)

Fact: $head \leftarrow \top.$ • *head* is always true

Constraint: $\bot \leftarrow body.$ • Invalidate this answer set if *body* is true

1) **Describe** the problem with facts and rules

$node(a).\ \ node(b).\ \ node(c).$

$edge(a, b).\ \ edge(b, c).\ \ edge(a, c).$

$edge(X, Y) \leftarrow edge(Y, X).$


```
Answer set 1: node(a) node(c) node(b)
              edge(a,b) edge(b,c) edge(a,c)
              edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming Concepts

Answer Set Programming (ASP): Declarative & logic programming

Rule: $\underbrace{A_0}_{\text{head}} \leftarrow \underbrace{A_1,\ \ldots,\ A_n,\ \texttt{not}\ A_{n+1},\ \ldots,\ \texttt{not}\ A_m}_{\text{body}}.$

- $\texttt{not}\ A_i$ is true if there is no proof of A_i (negation by failure)
- If *body* is true, then *head* must be true (logical consequence)
- We search for minimal answer sets (there can be 0, 1, many)

Fact: $head \leftarrow \top.$

- *head* is always true

Constraint: $\bot \leftarrow body.$

- Invalidate this answer set if *body* is true

1) **Describe** the problem with facts and rules

$node(a).\quad node(b).\quad node(c).$

$edge(a,b).\quad edge(b,c).\quad edge(a,c).$

$edge(X,Y) \leftarrow edge(Y,X).$


```
Answer set 1: node(a) node(c) node(b)
              edge(a,b) edge(b,c) edge(a,c)
              edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming Concepts

Answer Set Programming (ASP): Declarative & logic programming

Rule: $\underbrace{A_0}_{\text{head}} \leftarrow \underbrace{A_1,\ \ldots,\ A_n,\ \texttt{not}\ A_{n+1},\ \ldots,\ \texttt{not}\ A_m}_{\text{body}}.$

- $\texttt{not}\ A_i$ is true if there is no proof of A_i (negation by failure)
- If *body* is true, then *head* must be true (logical consequence)
- We search for minimal answer sets (there can be 0, 1, many)

Fact: *head*.

- *head* is always true

Constraint: $\bot \leftarrow$ *body*.

- Invalidate this answer set if *body* is true

1) **Describe** the problem with facts and rules

$node(a).\ \ node(b).\ \ node(c).$
$edge(a,b).\ \ edge(b,c).\ \ edge(a,c).$
$edge(X,Y) \leftarrow edge(Y,X).$


```
Answer set 1: node(a) node(c) node(b)
              edge(a,b) edge(b,c) edge(a,c)
              edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming Concepts

Answer Set Programming (ASP): Declarative & logic programming

Rule: $\underbrace{A_0}_{\text{head}} \leftarrow \underbrace{A_1,\ \ldots,\ A_n,\ \texttt{not}\ A_{n+1},\ \ldots,\ \texttt{not}\ A_m}_{\text{body}}.$

- not A_i is true if there is no proof of A_i (negation by failure)
- If *body* is true, then *head* must be true (logical consequence)
- We search for minimal answer sets (there can be 0, 1, many)

Fact: *head*.
- *head* is always true

Constraint: $\bot \leftarrow$ *body*.
- Invalidate this answer set if *body* is true

1) **Describe** the problem with facts and rules

$node(a).\quad node(b).\quad node(c).$

$edge(a,b).\quad edge(b,c).\quad edge(a,c).$

$edge(X,Y) \leftarrow edge(Y,X).$


```
Answer set 1: node(a) node(c) node(b)
              edge(a,b) edge(b,c) edge(a,c)
              edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming Concepts

Answer Set Programming (ASP): Declarative & logic programming

Rule: $\underbrace{A_0}_{\text{head}} \leftarrow \underbrace{A_1,\ \ldots,\ A_n,\ \texttt{not}\ A_{n+1},\ \ldots,\ \texttt{not}\ A_m}_{\text{body}}.$

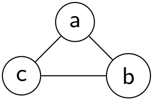
- $\texttt{not}\ A_i$ is true if there is no proof of A_i (negation by failure)
- If *body* is true, then *head* must be true (logical consequence)
- We search for minimal answer sets (there can be 0, 1, many)

Fact: *head*.
- *head* is always true

Constraint: $\bot \leftarrow$ *body*.
- Invalidate this answer set if *body* is true

1) Describe the problem with facts and rules

$node(a).\ \ node(b).\ \ node(c).$
$edge(a,b).\ \ edge(b,c).\ \ edge(a,c).$
$edge(X,Y) \leftarrow edge(Y,X).$




```
Answer set 1: node(a) node(c) node(b)
              edge(a,b) edge(b,c) edge(a,c)
              edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming Concepts

Answer Set Programming (ASP): Declarative & logic programming

Rule: $\underbrace{A_0}_{\text{head}} \leftarrow \underbrace{A_1,\ \ldots,\ A_n,\ \texttt{not}\ A_{n+1},\ \ldots,\ \texttt{not}\ A_m}_{\text{body}}.$

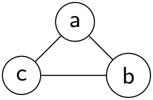
- not A_i is true if there is no proof of A_i (negation by failure)
- If *body* is true, then *head* must be true (logical consequence)
- We search for minimal answer sets (there can be 0, 1, many)

Fact: *head*.
- *head* is always true

Constraint: $\bot \leftarrow$ *body*.
- Invalidate this answer set if *body* is true

1) Describe the problem with facts and rules

$node(a).\ \ node(b).\ \ node(c).$
$edge(a,b).\ \ edge(b,c).\ \ edge(a,c).$
$edge(X,Y) \leftarrow edge(Y,X).$




```
Answer set 1: node(a) node(c) node(b)
              edge(a,b) edge(b,c) edge(a,c)
              edge(b,a) edge(c,b) edge(c,a)
```

Answer Set Programming Concepts

Enumeration: *atom* : *criterion*

- Enumerates all atoms of the form *atom* according to *criterion*

Cardinalities: *min* { *atom* : *criterion* } *max* ← *body*.

- Keep between *min* and *max* possibilities
- Creates as many answer sets as there are combinations

2) **Enumerate** of all candidate solutions using cardinalities

$color(red).\quad color(green).\quad color(blue).$

$1\ \{\ attrib(X,C) : color(C)\ \}\ 1 \leftarrow node(X).$

```
Answer set 1: attrib(b,red) attrib(c,red) attrib(a,red)
Answer set 2: attrib(b,red) attrib(c,red) attrib(a,blue)
Answer set 3: attrib(b,red) attrib(c,green) attrib(a,blue)
   ⋮ (27 answer sets)
```

Answer Set Programming Concepts

Enumeration: *atom* : *criterion*

- Enumerates all atoms of the form *atom* according to *criterion*

Cardinalities: $min\ \{\ atom : criterion\ \}\ max \leftarrow body.$

- Keep between *min* and *max* possibilities
- Creates as many answer sets as there are combinations

2) **Enumerate** of all candidate solutions using cardinalities

$color(red).\quad color(green).\quad color(blue).$

$1\ \{\ attrib(X, C) : color(C)\ \}\ 1 \leftarrow node(X).$

```
Answer set 1: attrib(b,red) attrib(c,red) attrib(a,red)
Answer set 2: attrib(b,red) attrib(c,red) attrib(a,blue)
Answer set 3: attrib(b,red) attrib(c,green) attrib(a,blue)
    ⋮  (27 answer sets)
```

Answer Set Programming Concepts

Enumeration: *atom* : *criterion*

- Enumerates all atoms of the form *atom* according to *criterion*

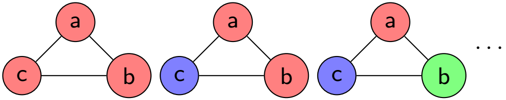
Cardinalities: *min* { *atom* : *criterion* } *max* ← *body*.

- Keep between *min* and *max* possibilities
- Creates as many answer sets as there are combinations

2) Enumerate of all candidate solutions using cardinalities

color(*red*). *color*(*green*). *color*(*blue*).
1 { *attrib*(*X*, *C*) : *color*(*C*) } 1 ← *node*(*X*).

```
Answer set 1: attrib(b,red) attrib(c,red) attrib(a,red)
Answer set 2: attrib(b,red) attrib(c,red) attrib(a,blue)
Answer set 3: attrib(b,red) attrib(c,green) attrib(a,blue)
   ⋮ (27 answer sets)
```

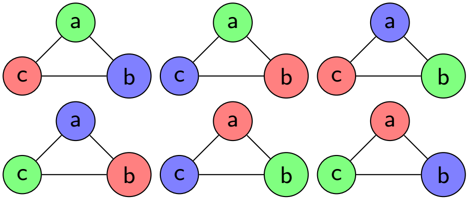


Answer Set Programming Concepts

3) Filter out the undesired candidates using constraints

$\bot \leftarrow attrib(X, C), attrib(Y, C), edge(X, Y).$

```
Answer set 1: attrib(b,green) attrib(c,blue) attrib(a,red)
Answer set 2: attrib(b,green) attrib(c,red) attrib(a,blue)
Answer set 3: attrib(b,blue) attrib(c,green) attrib(a,red)
Answer set 4: attrib(b,blue) attrib(c,red) attrib(a,green)
Answer set 5: attrib(b,red) attrib(c,green) attrib(a,blue)
Answer set 6: attrib(b,red) attrib(c,blue) attrib(a,green)
```



Steady States

[Ben Abdallah *et al.*, *IEEE Int. Conf. on Bioinformatics and Biomedicine*, 2015]

Steady States Enumeration (fixed points)

1) Describe the raw model with facts (automata, actions, playability)
2) Enumerate all possible states
3) Filter out states where at least one action is playable

Note: Identical for both synchronous and asynchronous semantics
→ Consistent with existing results on steady states

Reachability & Attractors

[Ben Abdallah *et al.*, *IEEE Int. Conf. on Bioinformatics and Biomedicine*, 2015]
[Ben Abdallah *et al.*, *Algorithms for Molecular Biology*, 2017]

Reachability analysis (reaching a given state)

1) Describe the raw model with facts
(automata, actions, initial states, targets)

2) Develop the dynamics:

- **[a]** describe playability with rules
- **[b]** enumerate potential futures with cardinalities and constraints

3) Filter out paths that don't contain the target state

Attractors Enumeration (find all smallest terminal components)

3) Filter out paths that are not cyclic and that can be escaped

Note: **[a]** can be adapted to any semantics

→ Already tested with synchronous & asynchronous
→ Other possible: general, with delay, with memory...

Conclusion on ASP for Model-Checking

[Ben Abdallah *et al.*, *IEEE Int. Conf. on Bioinformatics and Biomedicine*, 2015]
[Ben Abdallah *et al.*, *Algorithms for Molecular Biology*, 2017]

Pros:
- Very flexible (programming language)
- Complexity handled by the solver

Cons:
- Incremental approach (size of the paths)
- Still computational

Models		**Stable states**	**Reachability analysis**		
Name	Size	ASP	libddd[1]	GINsim[2]	ASP
egfr20	20	**0.017s**	1min 55s	2min 32s	**12s**
tcrsig40	40	**0.021s**	∞	∞	**4min 28s**

[1] LIP6/Move [Couvreur *et al.*, *Lecture Notes in Computer Science*, 2002]
[2] TAGC/IGC [Chaouiya, Naldi, Thieffry, *Methods in Molecular Biology*, 2012]

egfr20 : Epithelial Growth Factor Receptor (20 components) [Sahin *et al.*, 2009]

tcrsig40 : T-Cell Receptor (40 components) [Klamt *et al.*, 2006]

Conclusion on ASP for Model-checking

[Ben Abdallah *et al.*, *Algorithms for Molecular Biology*, 2017]

Models (Size)	n	**Attractors enumeration** — **asynchronous scheme** Δt (ms)	$\exists$?**A**	**synchronous scheme** Δt (ms)	$\exists$?**A**
Lambda phage (4)	2	14	**yes**	14	**yes**
	10	1,352	no	842	no
	20	15,656	no	14,452	no
Tcrsig (40)	2	26	no	25	no
	6	353	no	288	**yes**
	10	2,420	no	1,841	no
	20	85,599	no	27,078	no
FGF (59)	2	38	no	36	no
	10	2,080	no	1,953	no
	20	30,861	no	29,838	no
T-helper (101)	2	180	no	125	**yes**
	4	782	no	1,064	no
	6	4,271	no	2,372	**yes**
	9	26,443	no	7,042	**yes**
	12	107,358	no	28,520	**yes**
	20	4,230,836 $\sim$ 1h17	no	187,105 $\sim$ 3min	no

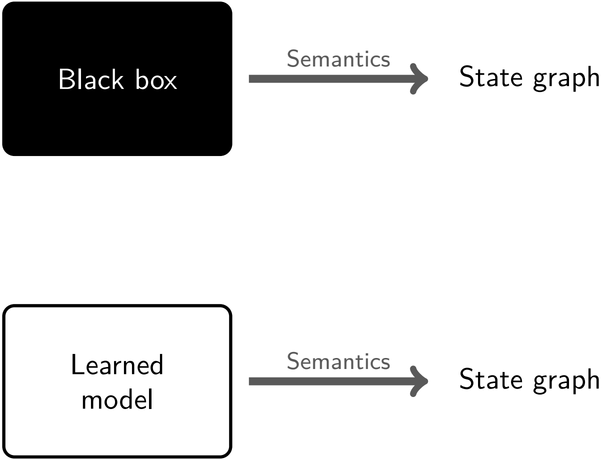
Lambda phage: Lysis/lysogenization decision in bacteriophage lambda [Thieffry & Thomas, 1995]
FGF: Drosophila FGF signaling pathway [Mbodj *et al.*, 2013]
T-helper: T-helper cell differentiation [Abou-Jaoudé *et al.*, 2014]

Learning Models from Data

Learning Models from Time Series Data

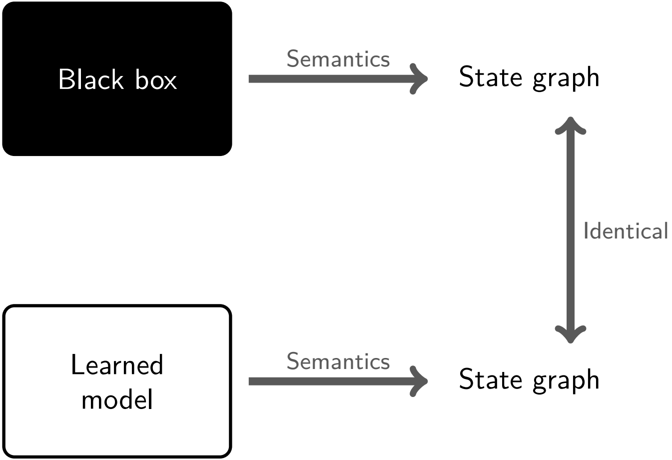
Learning Models from Execution Traces

[Ribeiro *et al.*, *Inductive Logic Programming*, 2018] (ACEDIA)
[Ribeiro *et al.*, *Inductive Logic Programming*, 2017] (GULA)



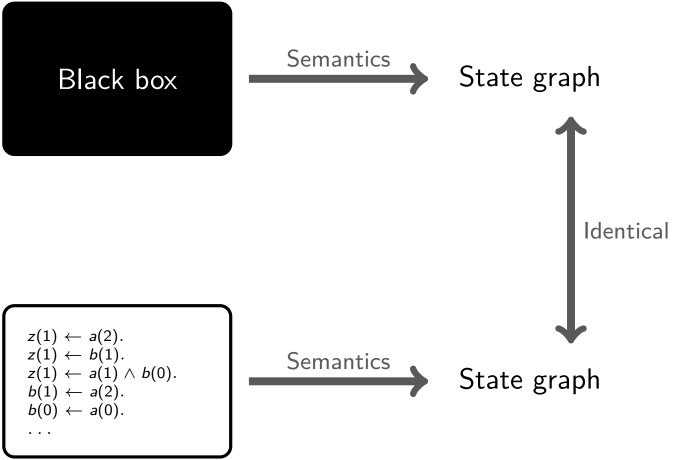
Learning Models from Execution Traces

[Ribeiro *et al.*, *Inductive Logic Programming*, 2018] (ACEDIA)
[Ribeiro *et al.*, *Inductive Logic Programming*, 2017] (GULA)



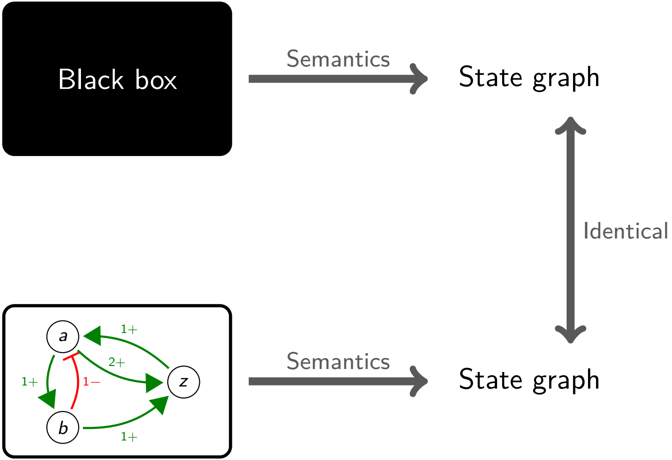
Learning Models from Execution Traces

[Ribeiro *et al.*, *Inductive Logic Programming*, 2018] (ACEDIA)
[Ribeiro *et al.*, *Inductive Logic Programming*, 2017] (GULA)



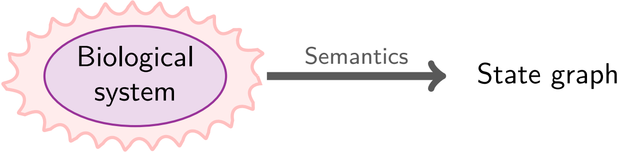
Learning Models from Execution Traces

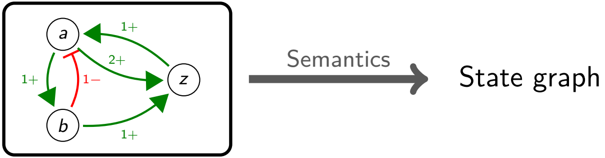
[Ribeiro *et al.*, *Inductive Logic Programming*, 2018] (ACEDIA)
[Ribeiro *et al.*, *Inductive Logic Programming*, 2017] (GULA)



Learning Models from Execution Traces

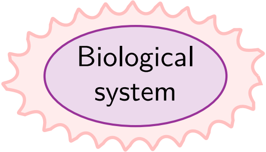
[Ribeiro *et al.*, *Inductive Logic Programming*, 2018] (ACEDIA)
[Ribeiro *et al.*, *Inductive Logic Programming*, 2017] (GULA)



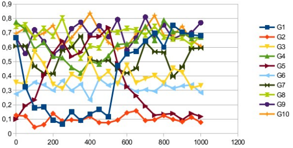


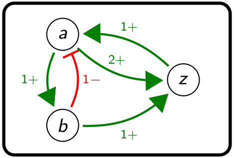
Learning Models from Execution Traces

[Ribeiro *et al.*, *Inductive Logic Programming*, 2018] (ACEDIA)
[Ribeiro *et al.*, *Inductive Logic Programming*, 2017] (GULA)



Semantics



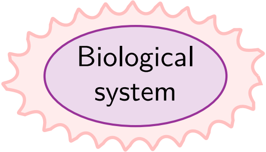


Semantics

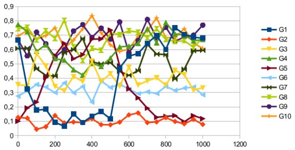
State graph

Learning Models from Execution Traces

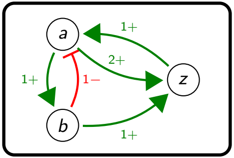
[Ribeiro *et al.*, *Inductive Logic Programming*, 2018] (ACEDIA)
[Ribeiro *et al.*, *Inductive Logic Programming*, 2017] (GULA)



Semantics



Equivalent (discretization)

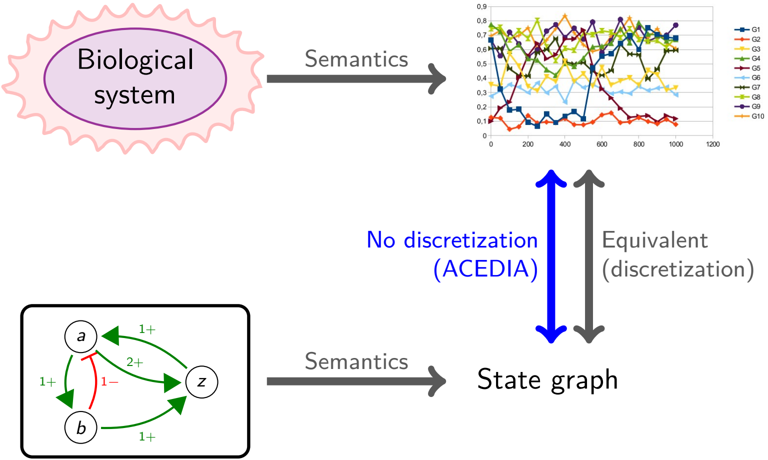


Semantics

State graph

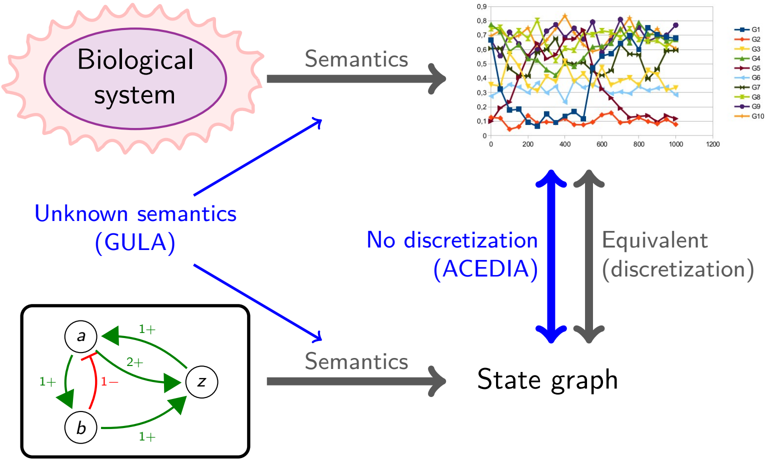
Learning Models from Execution Traces

[Ribeiro *et al.*, *Inductive Logic Programming*, 2018] (ACEDIA)
[Ribeiro *et al.*, *Inductive Logic Programming*, 2017] (GULA)



Learning Models from Execution Traces

[Ribeiro *et al.*, *Inductive Logic Programming*, 2018] (ACEDIA)
[Ribeiro *et al.*, *Inductive Logic Programming*, 2017] (GULA)



Logic Rule

$$\underbrace{x_0^{val_0}(t)}_{head} \leftarrow \underbrace{x_1^{val_1}(t-1) \ \wedge \ x_2^{val_2}(t-1) \ \wedge \ \ldots \ \wedge \ x_n^{val_n}(t-1)}_{body}.$$

→ When *body* is true, *head* is a potential outcome

Examples: $a^1 \leftarrow \{a^2, b^0, c^1\}.$ $b^1 \leftarrow \{c^1\}.$ $c^0 \leftarrow \emptyset.$ } all **match** $\langle a^2, b^0, c^1 \rangle$

A rule R **matches** a state s iff $body \subseteq s$

→ If the rule **matches** a state s then
there exists a successor state $s \rightarrow s'$ so that $head \in s'$

Semantics: same as for discrete networks
(synchronous, asynchronous, generalized)

Logic Rule

$$\underbrace{x_0^{val_0}(t)}_{head} \leftarrow \underbrace{\{\ x_1^{val_1}(t-1),\ x_2^{val_2}(t-1),\ \ldots,\ x_n^{val_n}(t-1)\ \}}_{body}.$$

→ When *body* is true, *head* is a potential outcome

Examples: $a^1 \leftarrow \{a^2, b^0, c^1\}.$ $b^1 \leftarrow \{c^1\}.$ $c^0 \leftarrow \emptyset.$ } all **match** $\langle a^2, b^0, c^1 \rangle$

A rule R **matches** a state s iff $body \subseteq s$

→ If the rule **matches** a state s then
there exists a successor state $s \rightarrow s'$ so that $head \in s'$

Semantics: same as for discrete networks
(synchronous, asynchronous, generalized)

Logic Rule

$$\underbrace{x_0^{val_0}}_{head} \leftarrow \underbrace{\{\ x_1^{val_1},\ x_2^{val_2},\ \dots,\ x_n^{val_n}\ \}}_{body}.$$

→ When *body* is true, *head* is a potential outcome

Examples: $a^1 \leftarrow \{a^2, b^0, c^1\}$. $b^1 \leftarrow \{c^1\}$. $c^0 \leftarrow \emptyset$. } all **match** $\langle a^2, b^0, c^1 \rangle$

A rule R **matches** a state s iff $body \subseteq s$

→ If the rule **matches** a state s then there exists a successor state $s \to s'$ so that $head \in s'$

Semantics: same as for discrete networks (synchronous, asynchronous, generalized)

Logic Rule

$$\underbrace{x_0^{val_0}}_{head} \leftarrow \underbrace{\{\ x_1^{val_1},\ x_2^{val_2},\ \ldots,\ x_n^{val_n}\ \}}_{body}.$$

→ When *body* is true, *head* is a potential outcome

Examples:
$a^1 \leftarrow \{a^2, b^0, c^1\}.$
$b^1 \leftarrow \{c^1\}.$
$c^0 \leftarrow \emptyset.$
all **match** $\langle a^2, b^0, c^1 \rangle$

A rule R **matches** a state s iff $body \subseteq s$

→ If the rule **matches** a state s then
there exists a successor state $s \rightarrow s'$ so that $head \in s'$

Semantics: same as for discrete networks
(synchronous, asynchronous, generalized)

Logic Rule

$$\underbrace{x_0^{val_0}}_{head} \leftarrow \underbrace{\{\ x_1^{val_1},\ x_2^{val_2},\ \ldots,\ x_n^{val_n}\ \}}_{body}.$$

→ When *body* is true, *head* is a potential outcome

Examples:
$a^1 \leftarrow \{a^2, b^0, c^1\}.$
$b^1 \leftarrow \{c^1\}.$
$c^0 \leftarrow \varnothing.$
} all **match** $\langle a^2, b^0, c^1 \rangle$

A rule R **matches** a state s iff $body \subseteq s$

→ If the rule **matches** a state s then
there exists a successor state $s \rightarrow s'$ so that $head \in s'$

Semantics: same as for discrete networks
(synchronous, asynchronous, generalized)

Logic Rule

$$\underbrace{x_0^{val_0}}_{head} \leftarrow \underbrace{\{\ x_1^{val_1},\ x_2^{val_2},\ \ldots,\ x_n^{val_n}\ \}}_{body}.$$

→ When *body* is true, *head* is a potential outcome

Examples:
$$\left.\begin{array}{l} a^1 \leftarrow \{a^2, b^0, c^1\}. \\ b^1 \leftarrow \{c^1\}. \\ c^0 \leftarrow \varnothing. \end{array}\right\} \text{ all } \textbf{match } \langle a^2, b^0, c^1 \rangle$$

A rule R **matches** a state s iff $body \subseteq s$

→ If the rule **matches** a state s then
there exists a successor state $s \rightarrow s'$ so that $head \in s'$

Semantics: same as for discrete networks
(synchronous, asynchronous, generalized)

Logic Rule

$$\underbrace{x_0^{val_0}}_{head} \leftarrow \underbrace{\{\, x_1^{val_1},\ x_2^{val_2},\ \ldots,\ x_n^{val_n} \,\}}_{body}.$$

→ When *body* is true, *head* is a potential outcome

Examples: $\left.\begin{array}{l} a^1 \leftarrow \{a^2, b^0, c^1\}. \\ b^1 \leftarrow \{c^1\}. \\ c^0 \leftarrow \emptyset. \end{array}\right\}$ all **match** $\langle a^2, b^0, c^1 \rangle$

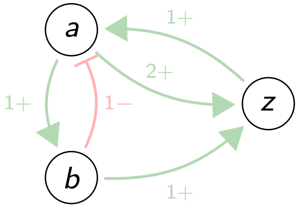
A rule R **matches** a state s iff *body* $\subseteq s$

→ If the rule **matches** a state s then
there exists a successor state $s \rightarrow s'$ so that *head* $\in s'$

Semantics: same as for discrete networks
(synchronous, asynchronous, generalized)

Model as a Logic Program

Discrete model:



+ Parameters
  or logic gates

Logic program:

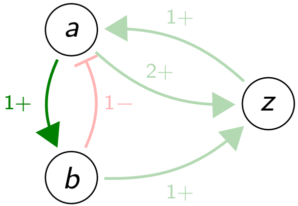
$b(1) \leftarrow a(1).$
$b(1) \leftarrow a(2).$
$b(0) \leftarrow a(0).$

$z(1) \leftarrow a(2) \wedge b(1).$
$z(0) \leftarrow a(0).$
$z(0) \leftarrow a(1).$
$z(0) \leftarrow b(0).$

etc...

Model as a Logic Program

Discrete model:



+ Parameters
or logic gates

Logic program:

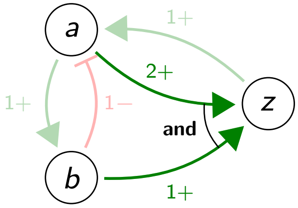
$b(1) \leftarrow a(1).$
$b(1) \leftarrow a(2).$
$b(0) \leftarrow a(0).$

$z(1) \leftarrow a(2) \wedge b(1).$
$z(0) \leftarrow a(0).$
$z(0) \leftarrow a(1).$
$z(0) \leftarrow b(0).$

etc...

Model as a Logic Program

Discrete model:



+ Parameters
or logic gates

Logic program:

$b(1) \leftarrow a(1).$

$b(1) \leftarrow a(2).$

$b(0) \leftarrow a(0).$

$z(1) \leftarrow a(2) \wedge b(1).$

$z(0) \leftarrow a(0).$

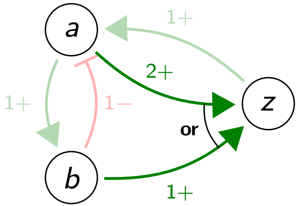
$z(0) \leftarrow a(1).$

$z(0) \leftarrow b(0).$

etc...

Model as a Logic Program

Discrete model:



+ Parameters
or logic gates

Logic program:

$b(1) \leftarrow a(1).$

$b(1) \leftarrow a(2).$

$b(0) \leftarrow a(0).$

$z(1) \leftarrow a(2).$

$z(1) \leftarrow b(1).$

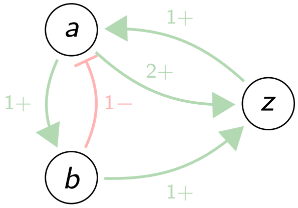
$z(0) \leftarrow a(1) \wedge b(0).$

$z(0) \leftarrow a(0) \wedge b(0).$

etc…

Model as a Logic Program

Discrete model:



+ Parameters
or logic gates

Logic program:

$b(1) \leftarrow a(1).$
$b(1) \leftarrow a(2).$
$b(0) \leftarrow a(0).$

$z(1) \leftarrow a(2).$
$z(1) \leftarrow b(1).$
$z(0) \leftarrow a(1) \wedge b(0).$
$z(0) \leftarrow a(0) \wedge b(0).$

etc...

GULA: Algorithm

- Start from the most general program: $\{x^{val} \leftarrow \varnothing. \mid x \in V \wedge val \in \mathrm{dom}(x)\}$
- For each state s, for each rule R that allows a behavior not observed after s:
  $\rightarrow$ Make minimal revisions on R (add an atom not in s)
- Remove all rules that are not the most general

Formally proved with transitions generated in **synchronous**, **asynchronous** and **generalized** semantics; should also work for a wider class of semantics

But what if the semantics “hides” some parts of the program?

$\rightarrow$ We should learn the semantics too! (In progress...)

GULA: Algorithm

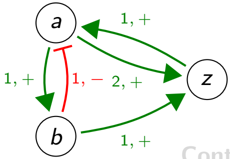
- Start from the most general program: $\{x^{val} \leftarrow \varnothing. \mid x \in V \land val \in \mathrm{dom}(x)\}$
- For each state s, for each rule R that allows a behavior not observed after s:
  $\rightarrow$ Make minimal revisions on R (add an atom not in s)
- Remove all rules that are not the most general

Formally proved with transitions generated in **synchronous**, **asynchronous** and **generalized** semantics; should also work for a wider class of semantics

But what if the semantics "hides" some parts of the program?

$\rightarrow$ We should learn the semantics too! (In progress...)

(Continuum) Logic Program



Discrete partitioning:

$$a = [\![0; 2]\!]$$
$$b = [\![0; 1]\!]$$
$$z = [\![0; 1]\!]$$

Continuous partitioning:

$$a = [0 - 0.3 - 0.6 - 1]$$
$$b = [0 \quad - \quad 0.5 \quad - \quad 1]$$
$$z = [0 \quad - \quad 0.8 - 1]$$

Discrete Logic Program:

$$b(1) \leftarrow a(1).$$
$$b(1) \leftarrow a(2).$$
$$b(0) \leftarrow a(0).$$

$$z(1) \leftarrow a(2) \wedge b(1).$$

$$\vdots$$

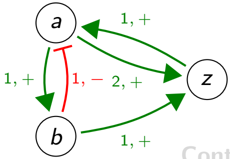
Continuum Logic Program:

$$b([0.5, 1]) \leftarrow a([0.3, 0.6]).$$
$$b([0.5, 1]) \leftarrow a([0.6, 1]).$$
$$b([0, 0.5]) \leftarrow a([0, 0.3]).$$

$$z([0.8, 1]) \leftarrow a([0.3, 0.6]) \wedge b([0.5, 1]).$$

$$\vdots$$

(Continuum) Logic Program



Discrete partitioning:

$$a = \llbracket 0; 2 \rrbracket$$
$$b = \llbracket 0; 1 \rrbracket$$
$$z = \llbracket 0; 1 \rrbracket$$

Continuous partitioning:

$$a = [0 - 0.3 - 0.6 - 1]$$
$$b = [0 - 0.5 - 1]$$
$$z = [0 - 0.8 - 1]$$

Discrete Logic Program:

$$b(1) \leftarrow a(1).$$
$$b(1) \leftarrow a(2).$$
$$b(0) \leftarrow a(0).$$

$$z(1) \leftarrow a(2) \wedge b(1).$$

$$\vdots$$

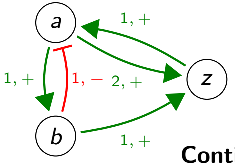
Continuum Logic Program:

$$b([0.5, 1]) \leftarrow a([0.3, 0.6]).$$
$$b([0.5, 1]) \leftarrow a([0.6, 1]).$$
$$b([0, 0.5]) \leftarrow a([0, 0.3]).$$

$$z([0.8, 1]) \leftarrow a([0.3, 0.6]) \wedge b([0.5, 1]).$$

$$\vdots$$

(Continuum) Logic Program



Discrete partitioning:

$$a = [\![0; 2]\!]$$
$$b = [\![0; 1]\!]$$
$$z = [\![0; 1]\!]$$

Continuous partitioning:

$$a = [0 - 0.3 - 0.6 - 1]$$
$$b = [0 \quad - \quad 0.5 \quad - \quad 1]$$
$$z = [0 \quad - \quad 0.8 - 1]$$

Discrete Logic Program:

$$b(1) \leftarrow a(1).$$
$$b(1) \leftarrow a(2).$$
$$b(0) \leftarrow a(0).$$

$$z(1) \leftarrow a(2) \wedge b(1).$$

⋮

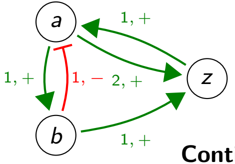
Continuum Logic Program:

$$b([0.5, 1]) \leftarrow a([0.3, 0.6]).$$
$$b([0.5, 1]) \leftarrow a([0.6, 1]).$$
$$b([0, 0.5]) \leftarrow a([0, 0.3]).$$

$$z([0.8, 1]) \leftarrow a([0.3, 0.6]) \wedge b([0.5, 1]).$$

⋮

(Continuum) Logic Program



Discrete partitioning:

$$a = [\![0;2]\!]$$
$$b = [\![0;1]\!]$$
$$z = [\![0;1]\!]$$

Continuous partitioning:

$$a = [0 - 0.3 - 0.6 - 1]$$
$$b = [0 \quad - \quad 0.5 \quad - \quad 1]$$
$$z = [0 \quad - \quad 0.8 - 1]$$

Discrete Logic Program:

$$b(1) \leftarrow a(1).$$
$$b(1) \leftarrow a(2).$$
$$b(0) \leftarrow a(0).$$

$$z(1) \leftarrow a(2) \wedge b(1).$$

$$\vdots$$

Continuum Logic Program:

$$b([0.5, 1]) \leftarrow a([0.3, 0.6]).$$
$$b([0.5, 1]) \leftarrow a([0.6, 1]).$$
$$b([0, 0.5]) \leftarrow a([0, 0.3]).$$

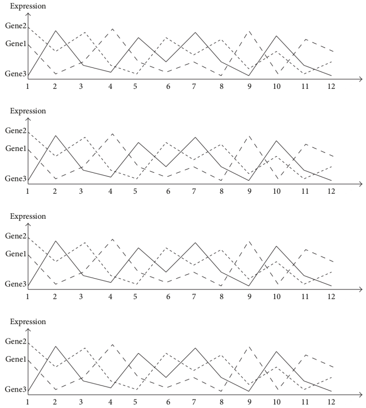
$$z([0.8, 1]) \leftarrow a([0.3, 0.6]) \wedge b([0.5, 1]).$$

$$\vdots$$

ACEDIA: Refinement of the Continuum Logic Program

INPUT:

A set of time series data



OUTPUT:

A continuum logic program

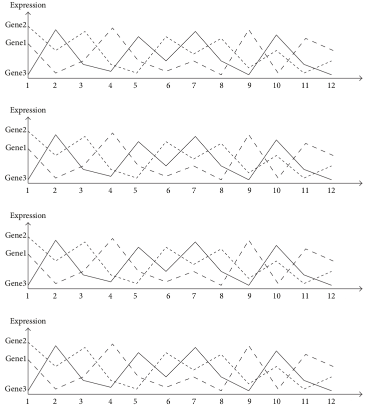
Equivalent to a regulatory network

$$p([0,0.5]) \leftarrow q([0,0.5]).$$
$$p([0.5,1]) \leftarrow q([0.5,1]).$$
$$q([0,0.5]) \leftarrow p([0,0.5]) \wedge r([0.5,1]).$$
$$q([0.5,1]) \leftarrow p([0.5,1]) \wedge r([0.5,1]).$$
$$r([0,0.5]) \leftarrow p([0.5,1]).$$
$$r([0.5,1]) \leftarrow p([0,0.5]).$$

- **Pros:** No discretization of the data
- **Cons:** Sensitive to noise, synchronous semantics only

ACEDIA: Refinement of the Continuum Logic Program

INPUT:
A set of time series data



OUTPUT:
A continuum logic program
Equivalent to a regulatory network

$p([0, 0.5]) \leftarrow q([0, 0.5])$.

$p([0.5, 1]) \leftarrow q([0.5, 1])$.

$q([0, 0.5]) \leftarrow p([0, 0.5]) \wedge r([0.3, 1])$.

$q([0.5, 1]) \leftarrow p([0.5, 1]) \wedge r([0.3, 1])$.

$r([0, 0.3]) \leftarrow p([0.5, 1])$.

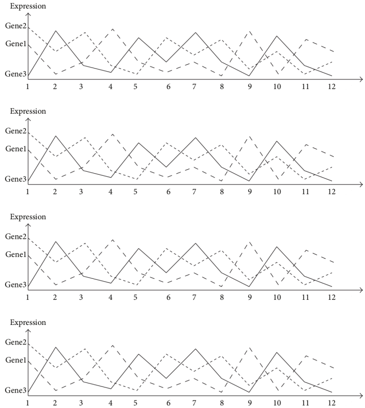
$r([0.3, 1]) \leftarrow p([0, 0.5])$.

- **Pros:** No discretization of the data
- **Cons:** Sensitive to noise, synchronous semantics only

ACEDIA: Refinement of the Continuum Logic Program

INPUT:

A set of time series data



OUTPUT:

A continuum logic program

Equivalent to a regulatory network

$$p([0,0.5]) \leftarrow q([0,0.5]).$$
$$p([0.5,1]) \leftarrow q([0.5,1]).$$
$$q([0,0.5]) \leftarrow p([0,0.5]) \wedge r([0.3,1]).$$
$$q([0.5,1]) \leftarrow p([0.5,1]) \wedge r([0.3,1]).$$
$$r([0,0.3]) \leftarrow p([0.5,1]).$$
$$r([0.3,1]) \leftarrow p([0,0.5]).$$

- **Pros:** No discretization of the data
- **Cons:** Sensitive to noise, synchronous semantics only

Conclusion: Learning from Time Series

Challenges:

- Discretization → ACEDIA
  A different discretization gives a different result
- Partial data → LUST
  Predict parts of the system
- Unknown semantics → GULA
  Measurment-dependent
- Heterogeneous "semantics" → Ongoing...
  Organism-dependent
- Changing behavior → Ongoing...
  Stochasticity
- Chronometry over chronology
  Learn time delays
- Learn from real data
  Avoid learning noise

Conclusion: Learning from Time Series

Challenges:

- Discretization → ACEDIA
  A different discretization gives a different result
- Partial data → LUST
  Predict parts of the system
- Unknown semantics → GULA
  Measurment-dependent
- Heterogeneous "semantics" → Ongoing...
  Organism-dependent
- Changing behavior → Ongoing...
  Stochasticity
- Chronometry over chronology
  Learn time delays
- Learn from real data
  Avoid learning noise

Conclusion: Learning from Time Series

Challenges:

- Discretization → ACEDIA
  A different discretization gives a different result
- Partial data → LUST
  Predict parts of the system
- Unknown semantics → GULA
  Measurment-dependent
- Heterogeneous "semantics" → Ongoing...
  Organism-dependent
- Changing behavior → Ongoing...
  Stochasticity
- Chronometry over chronology
  Learn time delays
- Learn from real data
  Avoid learning noise

Future works

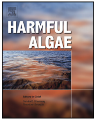
Harmful Algae 72 (2018) 1–13

ELSEVIER

Contents lists available at ScienceDirect

Harmful Algae

journal homepage: www.elsevier.com/locate/hal



Realized niche analysis of phytoplankton communities involving HAB: *Phaeocystis* spp. as a case study

Check for updates

Stéphane Karasiewicz[a,*], Elsa Breton[a], Alain Lefebvre[b], Tania Hernández Fariñas[c], Sébastien Lefebvre[a,d]

[a] *Univ. Lille, CNRS, Univ. Littoral Côte d'Opale, UMR 8187, LOG Laboratoire d'Océanologie et Géosciences, F 62930 Wimereux, France*
[b] *Ifremer, laboratoire Environnement et ressources du centre Manche Mer du Nord, 150 quai Gambetta, BP 699, 62321 Boulogne-sur-Mer, France*
[c] *Ifremer, Laboratoire Environnement Ressources de Normandie, Avenue du Général de Gaulle, BP 32, 14520 Port en Bessin, France*
[d] *Ifremer, Laboratoire Ressources Halieutiques, 150 Quai Gambetta BP 699, F-62321 Boulogne sur mer, France*

A R T I C L E I N F O

Article history:
Received 9 June 2017
Received in revised form 9 December 2017
Accepted 11 December 2017
Available online xxx

Keywords:
Harmful algae bloom
WitOMI

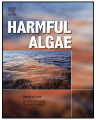
A B S T R A C T

The link between harmful algal blooms, phytoplankton community dynamics and global environmental change is not well understood. To tackle this challenging question, a new method was used to reveal how phytoplankton communities responded to environmental change with the occurrence of an harmful algae, using the coastal waters of the eastern English Channel as a case study. The great interannual variability in the magnitude and intensity of *Phaeocystis* spp. blooms, along with diatoms, compared to the ongoing gradual decrease in anthropogenic nutrient concentration and rebalancing of nutrient ratios; suggests that other factors, such as competition for resources, may also play an important role. A realized niche approach was used with the Outlying Mean Index analysis and the dynamics of the species' realized

Future works

Harmful Algae 72 (2018) 1–13

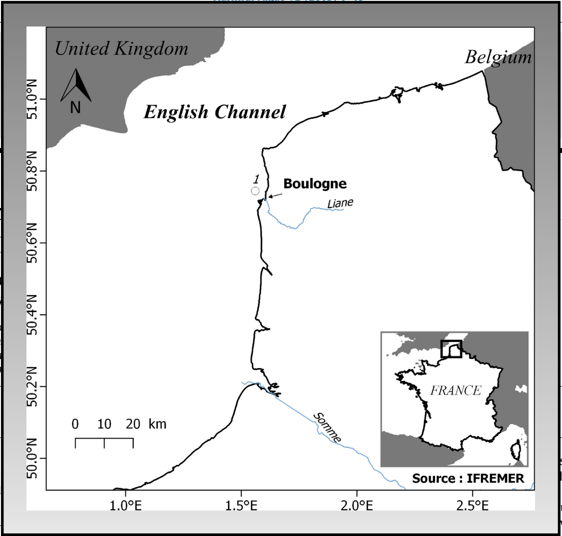
ELSEVIER



Realized nich… AB:
Phaeocystis s…

Stéphane Karasi…
Sébastien Lefebv…

[a] Univ. Lille, CNRS, Univ. Litto…
[b] Ifremer, laboratoire Environ…
[c] Ifremer, Laboratoire Environ…
[d] Ifremer, Laboratoire Ressou…



A R T I C L E I N F O

Article history:
Received 9 June 2017
Received in revised form 9…
Accepted 11 December 201…
Available online xxx

Keywords:
Harmful algae bloom
WitOMI

…s and global environmental …hod was used to reveal how … occurrence of an harmful …udy. The great interannual …with diatoms, compared to the ongoing gradual decrease in anthropogenic nutrient concentration and rebalancing of nutrient ratios; suggests that other factors, such as competition for resources, may also play an important role. A realized niche approach was used with the Outlying Mean Index analysis and the dynamics of the species' realized

Future works

SEANOE Sea scientific open data edition SEANOE

SRN dataset - Regional Observation and Monitoring Program for Phytoplankton and Hydrology in the eastern English Channel. 1992-2016.

Date	2017-09
Temporal extent	1992 -2016
Author(s)	SRN - Regional Observation and Monitoring program for Phytoplankton and Hydrology in the eastern English Channel
Contributor(s)	Lefebvre Alain, Blondel Camille, Duquesne Vincent, Hebert Pascale, Cordier Remy, Belin Catherine, Huguet Antoine, Durand Gaetane, Soudant Dominique
DOI	10.17882/50832
Publisher	SEANOE
Abstract	This SRN dataset includes long-term time series on marine phytoplankton and physico-chemical measures, since 1992, along the eastern English Channel coast. More precisely, samples were collected along transects offshore Dunkerque, Boulogne-sur-Mer and the bay of Somme. Data are complementary to REPHY and REPHYTOX datasets. Phytoplankton data essentially cover microscopic taxonomic identifications and counts, but also pigments measures (Chlorophyll-a and pheopigment). Physico-chemical measures include temperature, salinity, turbidity, suspended matters (organic, mineral), dissolved oxygen, dissolved inorganic nutrients (ammonium, nitrite+nitrate, phosphate, silicate).
Licence	CC BY
Utilisation	Data are published without any warranty, express or implied. The user assumes all risk arising from his/her use of data. Data are intended to be research-quality and include estimates of data quality and accuracy, but it is possible that these estimates or the data themselves contain errors. It is the sole responsibility of the user to assess if the data are appropriate for his/her use, and to interpret the data, data quality, and data accuracy accordingly. Authors welcome users to ask questions and report problems.

Click to download the data DATA

Bloom of the Prymnesiophyceae Phaeocystis globosa in the harbour of Boulogne sur Mer (eastern English Channel, France)

Download metadata
TXT, RIS, XLS, RTF, BIBTEX

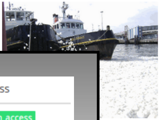
References

Future works

SEANOE Sea scientific open data edition SEANOE

SRN dataset - Regional Observation and Monitoring Program for Phytoplankton and Hydrology in the eastern English Channel. 1992-2016.

Click to download the data — DATA



ceae
harbour of
English Channel,

Date

Temporal exte

Author(s)

Contributor(s

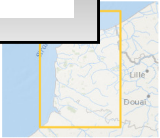
DOI

Publisher

Abstract

File	Size	Format	Processing	Access
Physico-chemical data	544 KB	CSV	Quality controlled data	Open access
Phytoplankton data	1 MB	CSV	Quality controlled data	Open access

collected along transects offshore Dunkerque, Boulogne-sur-Mer and the bay of Somme. Data are complementary to REPHY and REPHYTOX datasets. Phytoplankton data essentially cover microscopic taxonomic identifications and counts, but also pigments measures (Chlorophyll-a and pheopigment). Physico-chemical measures include temperature, salinity, turbidity, suspended matters (organic, mineral), dissolved oxygen, dissolved inorganic nutrients (ammonium, nitrite+nitrate, phosphate, silicate).



Licence — CC BY

Utilisation — Data are published without any warranty, express or implied. The user assumes all risk arising from his/her use of data. Data are intended to be research-quality and include estimates of data quality and accuracy, but it is possible that these estimates or the data themselves contain errors. It is the sole responsibility of the user to assess if the data are appropriate for his/her use, and to interpret the data, data quality, and data accuracy accordingly. Authors welcome users to ask questions and report problems.

Download metadata
TXT, RIS, XLS, RTF, BIBTEX

References

Thank you

Frameworks

- Thomas modeling, **asynchronous automata networks**

Analysis of the Dynamics

- **Efficient reachability analysis on large networks**
- **Dynamical patterns enumeration with answer set programming**
- Complex patterns enumeration with polyadic μ-calculus

Learning Models from Data

- Inference of constraints on hybrid parameters
- **Learning models from time series data**

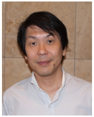
Learning New Knowledge from Models

- Computational model to study hepatocellular carcinoma progression
- Integrate heterogeneous clinical, genetic, imaging data with semantic web in order to learn variables of interest

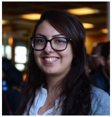
Collaborations

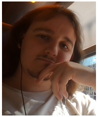

Olivier ROUX


Morgan MAGNIN


Katsumi INOUE


Loïc PAULEVÉ


Emna BEN ABDALLAH


Tony RIBEIRO

Martin LANGE

Jonathan BEHAEGEL

Jean-Paul COMET

Carito GUZIOLOWSKI

Nathalie THÉRET

Vincent LEGAGNEUX

Arnaud PORET

Lokmane CHEBOUBA

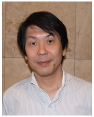
Alban GAIGNARD

Hala SKAF-MOLLI

Collaborations

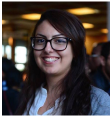
Olivier ROUX

Morgan MAGNIN

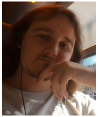


Katsumi INOUE

Loïc PAULEVÉ



Emna BEN ABDALLAH



Tony RIBEIRO

Martin LANGE

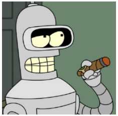
Jonathan BEHAEGEL

Jean-Paul COMET

Carito GUZIOLOWSKI

Nathalie THÉRET

Vincent LEGAGNEUX



Arnaud PORET

Lokmane CHEBOUBA

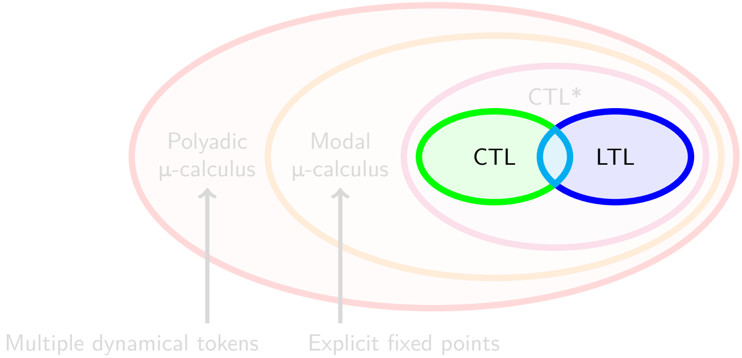
Alban GAIGNARD

Hala SKAF-MOLLI

Analysis of the Dynamics

Using µ-calculus for Complex Dynamical Patterns Enumeration

Polyadic μ-calculus

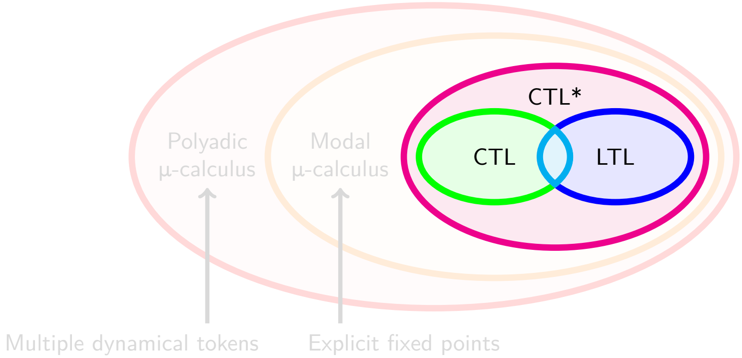


Polyadic (modal) μ-calculus allows to manipulate several tokens in parallel

$$\varphi = p_i \mid i \leftarrow j \mid i = j \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond_i \varphi \mid \Box_i \varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid X$$

- Modal operators: $\Box$ ("for all successors"), $\Diamond$ ("there exists a successor")
- Fixed points: μ (least fixed point), ν (greatest fixed point)
- Tokens (i, j) and their manipulation ($i = j$ and $i \leftarrow j$)

Polyadic μ-calculus

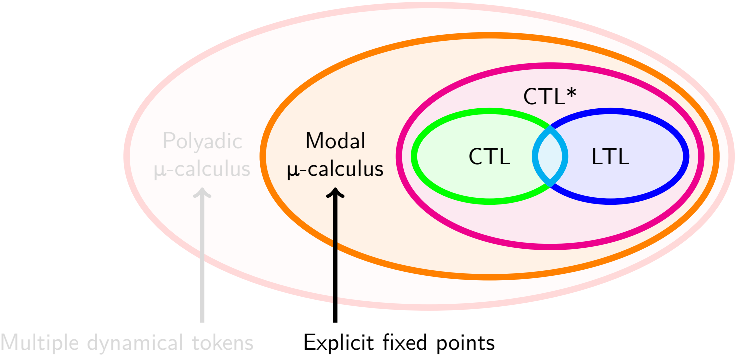


Polyadic (modal) μ-calculus allows to manipulate several tokens in parallel

$$\varphi = p_i \mid i \leftarrow j \mid i = j \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond_i \varphi \mid \Box_i \varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid X$$

- Modal operators: $\Box$ (“for all successors”), $\Diamond$ (“there exists a successor”)
- Fixed points: μ (least fixed point), ν (greatest fixed point)
- Tokens (i, j) and their manipulation ($i = j$ and $i \leftarrow j$)

Polyadic μ-calculus

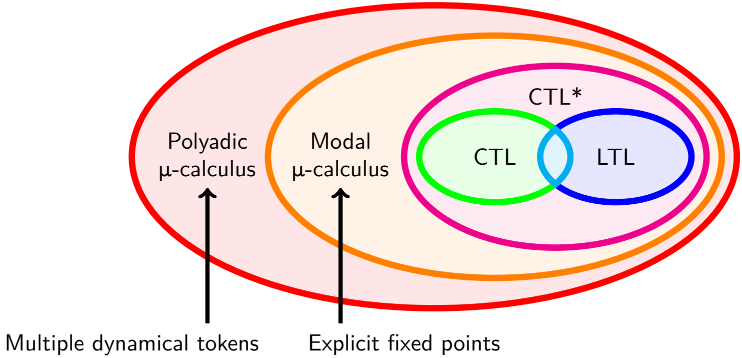


Polyadic (modal) μ-calculus allows to manipulate several tokens in parallel

$$\varphi = p_i \mid i \leftarrow j \mid i = j \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond_i \varphi \mid \Box_i \varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid X$$

- Modal operators: $\Box$ ("for all successors"), $\Diamond$ ("there exists a successor")
- Fixed points: μ (least fixed point), ν (greatest fixed point)
- Tokens (i, j) and their manipulation ($i = j$ and $i \leftarrow j$)

Polyadic μ-calculus



Polyadic (modal) μ-calculus allows to manipulate several tokens in parallel

$$\varphi = p_i \mid i \leftarrow j \mid i = j \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond_i \varphi \mid \Box_i \varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid X$$

- Modal operators: $\Box$ ("for all successors"), $\Diamond$ ("there exists a successor")
- Fixed points: μ (least fixed point), ν (greatest fixed point)
- Tokens (i, j) and their manipulation ($i = j$ and $i \leftarrow j$)

Learning Models from Biological Data

Learning Models from Time Series Data: Complements

GULA: Algorithm

- Start from the most general program:

$$P := \{x^{val} \leftarrow \varnothing. \mid x \in V \wedge val \in \mathrm{dom}(x)\}$$

- For each state s:
  $\rightarrow$ For each rule in conflict with the outcomes of s
  (that is, each rule R that allows a behavior not allowed after s)
  $\rightarrow$ Make minimal revisions on R to prevent this conflict

- Remove all rules that are not the most general

GULA: Minimal Revision of a Rule

The algorithm successively takes into account groups of transitions and performs **minimal modifications** on the program learned so far

Let R a rule in conflict with the current transitions, that is: there exists a state s so that R **matches** s, but $\forall s' \in \mathcal{S}$ so that $s \to s'$, $head(R) \notin s'$
That is: R expresses a potential outcome for a variable which never happens in s

Least specialization of R by s:

$$R \;:=\; head \leftarrow body$$

$$L_{spe}(R, s) \;:=\; \{head \leftarrow body \cup \{x^{val}\} \mid x^{val} \notin s \land \forall val' \in \mathbb{N}, x^{val'} \notin body\}$$

Least revision of P by a set of transitions T:

$$L_{rev}(P, T) \;:=\; (P \setminus R_P) \cup \bigcup_{R \in R_P} L_{spe}(R, s)$$

Scope of GULA

This learning should be independent from the semantics!

Formally proved: Compatible with transitions generated in **synchronous**, **asynchronous** and **generalized** semantics

Expectation: Compatible with a wider class of "learnable" semantics

Scope of GULA

This learning should be independent from the semantics!

Formally proved: Compatible with transitions generated in **synchronous**, **asynchronous** and **generalized** semantics

Expectation: Compatible with a wider class of “learnable” semantics

Scope of GULA

This learning **should be** independent from the semantics!

Formally proved: Compatible with transitions generated in **synchronous**, **asynchronous** and **generalized** semantics

Expectation: Compatible with a wider class of **“learnable”** semantics

Limits of our Definition of Semantics

Formally, a semantics is a function that, to each program, associates a set of transitions (with no dead-end)

$$\{\text{Set of all programs}\} \;\rightarrow\; (\mathcal{S} \rightarrow \wp(\mathcal{S}) \setminus \varnothing)$$

Not constrained enough as it allows some unwanted cases:

- a semantics where all variables are always updated to 0, disregarding any actual rules
- a semantics which behaves differently on one specific program (exception)

→ The program can be "hidden" and thus cannot be learned

Outcomes:

- Semantics should take into account the given program
- We can also learn the semantics (for now, we give a characterization)

Limits of our Definition of Semantics

Formally, a semantics is a function that, to each program, associates a set of transitions (with no dead-end)

$$\{\text{Set of all programs}\} \ \rightarrow \ (\mathcal{S} \rightarrow \wp(\mathcal{S}) \setminus \varnothing)$$

Not constrained enough as it allows some unwanted cases:

- a semantics where all variables are always updated to 0, disregarding any actual rules
- a semantics which behaves differently on one specific program (exception)

$\rightarrow$ The program can be "hidden" and thus cannot be learned

Outcomes:

- Semantics should take into account the given program
- We can also learn the semantics (for now, we give a characterization)

Limits of our Definition of Semantics

Formally, a semantics is a function that, to each program, associates a set of transitions (with no dead-end)

$$\{\text{Set of all programs}\} \;\rightarrow\; (\mathcal{S} \rightarrow \wp(\mathcal{S}) \setminus \varnothing)$$

Not constrained enough as it allows some unwanted cases:

- a semantics where all variables are always updated to 0, disregarding any actual rules
- a semantics which behaves differently on one specific program (exception)

$\rightarrow$ The program can be "hidden" and thus cannot be learned

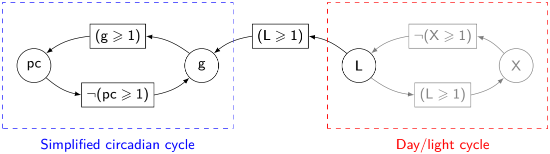
Outcomes:

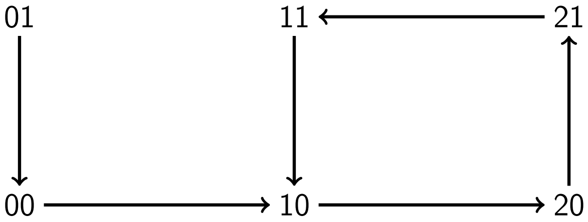
- Semantics should take into account the given program
- We can also learn the semantics (for now, we give a characterization)

Learning Models from Biological Data

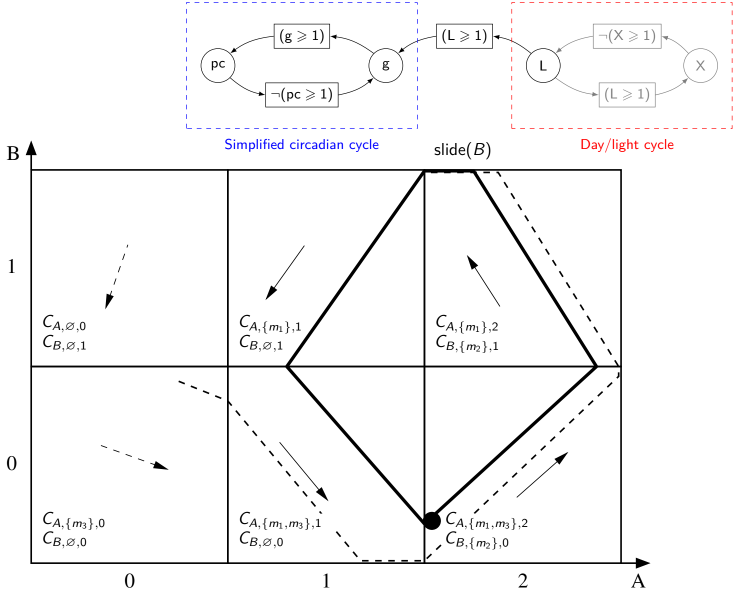
Inference of Constraints on Hybrid Parameters

Hybrid Thomas Modeling





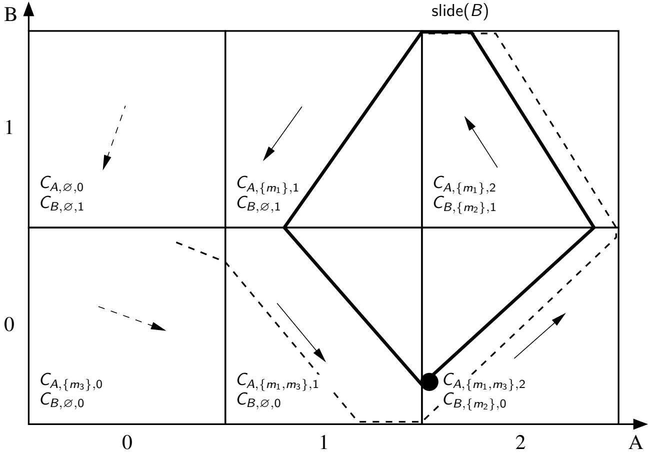
Hybrid Thomas Modeling



Hybrid Hoare Logic to Infer Parameters

[Behaegel *et al.*, *TIME'17*, 2017]

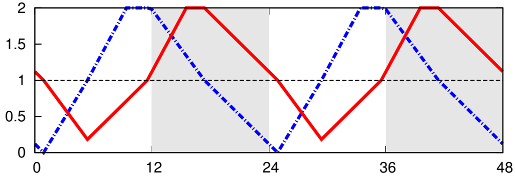
$$\left\{\begin{matrix} ??? \\ ??? \end{matrix}\right\} \begin{pmatrix} T_4 \\ \top \\ B+ \end{pmatrix} ; \begin{pmatrix} T_3 \\ \text{slide}^+(B) \\ A- \end{pmatrix} ; \begin{pmatrix} T_2 \\ \top \\ B- \end{pmatrix} ; \begin{pmatrix} T_1 \\ \top \\ A+ \end{pmatrix} \left\{\begin{matrix} D_0 \equiv (\eta_A = 2 \wedge \eta_B = 0) \\ H_0 \equiv (\pi_{\text{initial}} = \pi_{\text{final}}) \end{matrix}\right\}$$



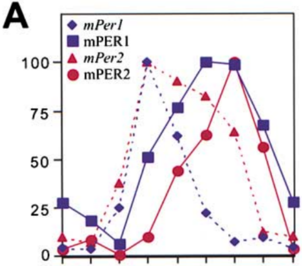
$$((((((((((\pi_g^{0\prime} = 0.12) \wedge ((\pi_{pc}^{0}{}' = 0.12) \wedge (\pi_L^{0\prime} = 0))) \wedge (((\pi_L^1 = 1) \wedge ((C_{L,\{m5\},0} > 0) \wedge (\pi_L^{1\prime} =$$
$$(\pi_L^1 - (C_{L,\{m5\},0} \times 6.6))))) \wedge ((\neg((C_{g,\varnothing,0} > 0) \wedge (\pi_g^{1\prime} > (\pi_g^1 - (C_{g,\varnothing,0} \times 6.6)))) \wedge (\neg((C_{pc,\varnothing,1} < 0) \wedge (\pi_{pc}^{1}{}' <$$
$$(\pi_{pc}^1 - (C_{pc,\varnothing,1} \times 6.6)))) \wedge \neg((C_{X,\varnothing,0} > 0) \wedge (\pi_X^{1\prime} > (\pi_X^1 - (C_{X,\varnothing,0} \times 6.6)))))) \wedge ((\pi_L^1 = (1 - \pi_L^{0\prime})) \wedge ((\pi_g^1 =$$
$$\pi_g^{0\prime}) \wedge ((\pi_{pc}^1 = \pi_{pc}^{0}{}') \wedge (\pi_X^1 = \pi_X^{0}{}'))))))) \wedge (((\pi_X^2 = 0) \wedge ((C_{X,\varnothing,1} < 0) \wedge (\pi_X^{2}{}' = (\pi_X^2 - (C_{X,\varnothing,1} \times 0.6))))) \wedge ((\neg((C_{g,\varnothing,0} >$$
$$0) \wedge (\pi_g^{2\prime} > (\pi_g^2 - (C_{g,\varnothing,0} \times 0.6)))) \wedge (\neg((C_{pc,\varnothing,1} < 0) \wedge (\pi_{pc}^{2}{}' < (\pi_{pc}^2 - (C_{pc,\varnothing,1} \times 0.6)))) \wedge \neg((C_{L,\varnothing,0} >$$
$$0) \wedge (\pi_L^{2\prime} > (\pi_L^2 - (C_{L,\varnothing,0} \times 0.6)))))) \wedge (((\pi_L^2 = 0) \wedge ((C_{L,\varnothing,0} < 0) \Rightarrow (\pi_L^{2\prime} < (\pi_L^2 - (C_{L,\varnothing,0} \times 0.6))))) \wedge ((\pi_X^2 =$$
$$(1 - \pi_X^{1}{}')) \wedge ((\pi_g^2 = \pi_g^{1\prime}) \wedge ((\pi_{pc}^2 = \pi_{pc}^{1}{}') \wedge (\pi_L^2 = \pi_L^{1\prime}))))))))\ntriangleleft \wedge (((\pi_g^3 = 0) \wedge ((C_{g,\varnothing,1} < 0) \wedge (\pi_g^{3\prime} =$$
$$(\pi_g^3 - (C_{g,\varnothing,1} \times 5.4))))) \wedge ((\neg((C_{pc,\{m2\},1} < 0) \wedge (\pi_{pc}^{3}{}' < (\pi_{pc}^3 - (C_{pc,\{m2\},1} \times 5.4)))) \wedge (\neg((C_{L,\varnothing,0} > 0) \wedge (\pi_L^{3\prime} >$$
$$(\pi_L^3 - (C_{L,\varnothing,0} \times 5.4)))) \wedge \neg((C_{X,\varnothing,1} < 0) \wedge (\pi_X^{3}{}' < (\pi_X^3 - (C_{X,\varnothing,1} \times 5.4)))))) \wedge (((\pi_{pc}^3 = 1) \wedge ((C_{pc,\{m2\},1} > 0) \Rightarrow$$
$$(\pi_{pc}^{3}{}' > (\pi_{pc}^3 - (C_{pc,\{m2\},1} \times 5.4))))) \wedge ((\pi_g^3 = (1 - \pi_g^{2\prime})) \wedge ((\pi_{pc}^3 = \pi_{pc}^{2}{}') \wedge ((\pi_L^3 = \pi_L^{2\prime}) \wedge (\pi_X^3 =$$
$$\pi_X^{2}{}')))))))) \wedge (((\pi_L^4 = 0) \wedge ((C_{L,\varnothing,1} < 0) \wedge (\pi_L^{4\prime} = (\pi_L^4 - (C_{L,\varnothing,1} \times 0.47))))) \wedge ((\neg((C_{g,\{m3\},1} < 0) \wedge (\pi_g^{4\prime} <$$
$$(\pi_g^4 - (C_{g,\{m3\},1} \times 0.47)))) \wedge (\neg((C_{pc,\{m2\},1} < 0) \wedge (\pi_{pc}^{4}{}' < (\pi_{pc}^4 - (C_{pc,\{m2\},1} \times 0.47)))) \wedge \neg((C_{X,\{m4\},1} <$$
$$0) \wedge (\pi_X^{4}{}' < (\pi_X^4 - (C_{X,\{m4\},1} \times 0.47)))))) \wedge ((\pi_L^4 = (1 - \pi_L^{3\prime})) \wedge ((\pi_g^4 = \pi_g^{3\prime}) \wedge ((\pi_{pc}^4 = \pi_{pc}^{3}{}') \wedge (\pi_X^4 =$$
$$\pi_X^{3}{}'))))))) \wedge (((\pi_{pc}^5 = 1) \wedge ((C_{pc,\{m2\},0} > 0) \wedge (\pi_{pc}^{5}{}' = (\pi_{pc}^5 - (C_{pc,\{m2\},0} \times 5.53))))) \wedge ((\neg((C_{g,\{m1,m3\},1} < 0) \wedge (\pi_g^{5\prime} <$$
$$(\pi_g^5 - (C_{g,\{m1,m3\},1} \times 5.53)))) \wedge (\neg((C_{L,\varnothing,1} < 0) \wedge (\pi_L^{5\prime} < (\pi_L^5 - (C_{L,\varnothing,1} \times 5.53)))) \wedge \neg((C_{X,\{m4\},1} < 0) \wedge (\pi_X^{5}{}' <$$
$$(\pi_X^5 - (C_{X,\{m4\},1} \times 5.53)))))) \wedge (((\pi_g^5 = 1) \wedge ((C_{g,\{m1,m3\},1} > 0) \Rightarrow (\pi_g^{5\prime} > (\pi_g^5 - (C_{g,\{m1,m3\},1} \times 5.53))))) \wedge ((\pi_{pc}^5 =$$
$$(1 - \pi_{pc}^{4}{}')) \wedge ((\pi_g^5 = \pi_g^{4\prime}) \wedge ((\pi_L^5 = \pi_L^{4\prime}) \wedge (\pi_X^5 = \pi_X^{4}{}')))))))) \wedge (((\pi_X^6 = 1) \wedge ((C_{X,\{m4\},0} > 0) \wedge (\pi_X^{6}{}' =$$
$$(\pi_X^6 - (C_{X,\{m4\},0} \times 0.6))))) \wedge ((\neg((C_{g,\{m1,m3\},1} < 0) \wedge (\pi_g^{6\prime} < (\pi_g^6 - (C_{g,\{m1,m3\},1} \times 0.6)))) \wedge (\neg((C_{pc,\{m2\},0} >$$
$$0) \wedge (\pi_{pc}^{6}{}' > (\pi_{pc}^6 - (C_{pc,\{m2\},0} \times 0.6)))) \wedge \neg((C_{L,\{m5\},1} < 0) \wedge (\pi_L^{6\prime} < (\pi_L^6 - (C_{L,\{m5\},1} \times 0.6)))))) \wedge ((\pi_X^6 =$$
$$(1 - \pi_X^{5}{}')) \wedge ((\pi_g^6 = \pi_g^{5}{}') \wedge ((\pi_{pc}^6 = \pi_{pc}^{5}{}') \wedge (\pi_L^6 = \pi_L^{5\prime}))))))) \wedge (((\pi_g^7 = 1) \wedge ((C_{g,\{m1,m3\},0} > 0) \wedge (\pi_g^{7\prime} =$$
$$(\pi_g^7 - (C_{g,\{m1,m3\},0} \times 4.5))))) \wedge ((\neg((C_{pc,\varnothing,0} > 0) \wedge (\pi_{pc}^{7}{}' > (\pi_{pc}^7 - (C_{pc,\varnothing,0} \times 4.5)))) \wedge (\neg((C_{L,\{m5\},1} < 0) \wedge (\pi_L^{7\prime} <$$
$$(\pi_L^7 - (C_{L,\{m5\},1} \times 4.5)))) \wedge \neg((C_{X,\{m4\},0} > 0) \wedge (\pi_X^{7\prime} > (\pi_X^7 - (C_{X,\{m4\},0} \times 4.5)))))) \wedge ((\pi_g^7 = (1 - \pi_g^{6\prime})) \wedge ((\pi_{pc}^7 = \pi_{pc}^{6}{}') \wedge$$
$$((\pi_L^7 = \pi_L^{6\prime}) \wedge (\pi_X^7 = \pi_X^{6}{}'))))))) \wedge (((\pi_{pc}^8 = 0) \wedge ((C_{pc,\varnothing,1} < 0) \wedge (\pi_{pc}^{8}{}' = (\pi_{pc}^8 - (C_{pc,\varnothing,1} \times 0.9))))) \wedge ((\neg((C_{g,\{m3\},0} >$$
$$0) \wedge (\pi_g^{8\prime} > (\pi_g^8 - (C_{g,\{m3\},0} \times 0.9)))) \wedge (\neg((C_{L,\{m5\},1} < 0) \wedge (\pi_L^{8\prime} < (\pi_L^8 - (C_{L,\{m5\},1} \times 0.9)))) \wedge \neg((C_{X,\{m4\},0} >$$
$$0) \wedge (\pi_X^{8}{}' > (\pi_X^8 - (C_{X,\{m4\},0} \times 0.9)))))) \wedge ((\pi_{pc}^8 = (1 - \pi_{pc}^{7}{}')) \wedge ((\pi_g^8 = \pi_g^{7\prime}) \wedge ((\pi_L^8 = \pi_L^{7\prime}) \wedge (\pi_X^8 = \pi_X^{7}{}')))))))$$

Results

- **Simplifications** of the constraints → Not very effective
- Using a non-linear solver: **AbSolute** → We obtain solutions
- Results checked with a simulation:



Simulation with 1 set of compatible values



Experiments

Learning New Knowledge from Models

Computational Model to Study Hepatocellular Carcinoma Progression

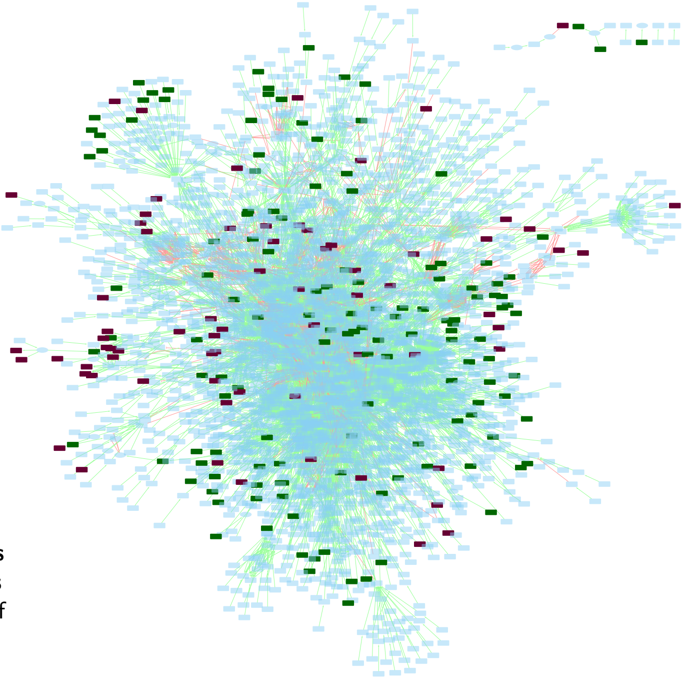
Graph content:

- 3'383 nodes
- 13'771 edges
  - 11'661 activations
  - 2'110 inhibitions

1913 genes from the differential expression
Only 209 are found in Kegg:

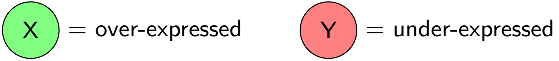
- 138 up-regulated
- 71 down-regulated
- 3174 new nodes

Nodes with up to:
92 incoming influences
79 outgoing influences
→ Nodes with a lot of impact on the network

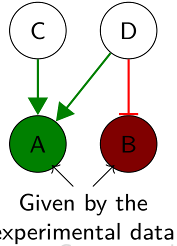


Graph Coloring

- Coloring = information attached to nodes about over- or under-expression



- Provenance = experimental (expression data) & computational (inference)



- Compute all colorings without inconsistencies
- **Prediction** = a node that is always colored the same

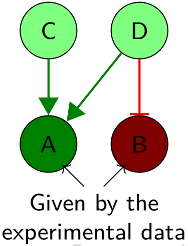
Here, only 1 prediction: D

- All computed by **Iggy** [Thiele *et al.*, *BMC Bioinformatics*, 2015] (Answer Set Programming)

Graph Coloring

- Coloring = information attached to nodes about over- or under-expression

- Provenance = experimental (expression data) & computational (inference)



- Compute all colorings without inconsistencies
- **Prediction** = a node that is always colored the same

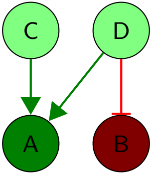
  Here, only 1 prediction: D

- All computed by **Iggy** [Thiele *et al.*, *BMC Bioinformatics*, 2015] (Answer Set Programming)

Graph Coloring

- Coloring = information attached to nodes about over- or under-expression

- Provenance = experimental (expression data) & computational (inference)



Consistent

- Compute all colorings without inconsistencies
- **Prediction** = a node that is always colored the same

Here, only 1 prediction: D

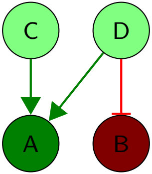
- All computed by **Iggy** [Thiele *et al.*, *BMC Bioinformatics*, 2015] (Answer Set Programming)

Graph Coloring

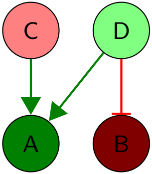
- Coloring = information attached to nodes about over- or under-expression

  X = over-expressed Y = under-expressed

- Provenance = experimental (expression data) & computational (inference)



Consistent



Consistent

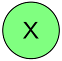
- Compute all colorings without inconsistencies
- **Prediction** = a node that is always colored the same

  Here, only 1 prediction: D

- All computed by **Iggy** [Thiele *et al.*, *BMC Bioinformatics*, 2015] (Answer Set Programming)

Graph Coloring

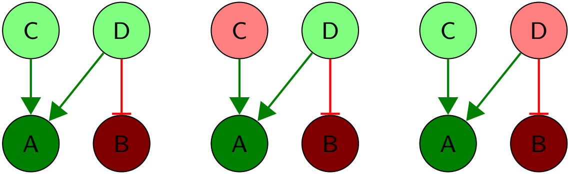
- Coloring = information attached to nodes about over- or under-expression



= over-expressed


= under-expressed

- Provenance = experimental (expression data) & computational (inference)



Consistent **Consistent** **Inconsistent**

- Compute all colorings without inconsistencies
- **Prediction** = a node that is always colored the same

  Here, only 1 prediction:

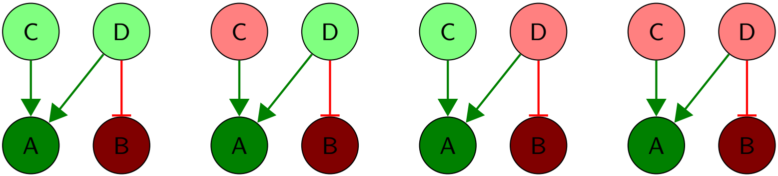
- All computed by **Iggy** [Thiele *et al.*, *BMC Bioinformatics*, 2015] (Answer Set Programming)

Graph Coloring

- Coloring = information attached to nodes about over- or under-expression

  X = over-expressed Y = under-expressed

- Provenance = experimental (expression data) & computational (inference)



Consistent **Consistent** **Inconsistent** **Inconsistent**

- Compute all colorings without inconsistencies
- **Prediction** = a node that is always colored the same

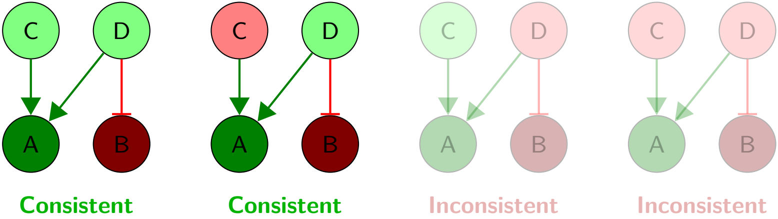
  Here, only 1 prediction: D

- All computed by **Iggy** [Thiele *et al.*, *BMC Bioinformatics*, 2015] (Answer Set Programming)

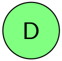
Graph Coloring

- Coloring = information attached to nodes about over- or under-expression

- Provenance = experimental (expression data) & computational (inference)



- Compute all colorings without inconsistencies
- **Prediction** = a node that is always colored the same

Here, only 1 prediction: 

- All computed by **Iggy** [Thiele *et al.*, *BMC Bioinformatics*, 2015] (Answer Set Programming)

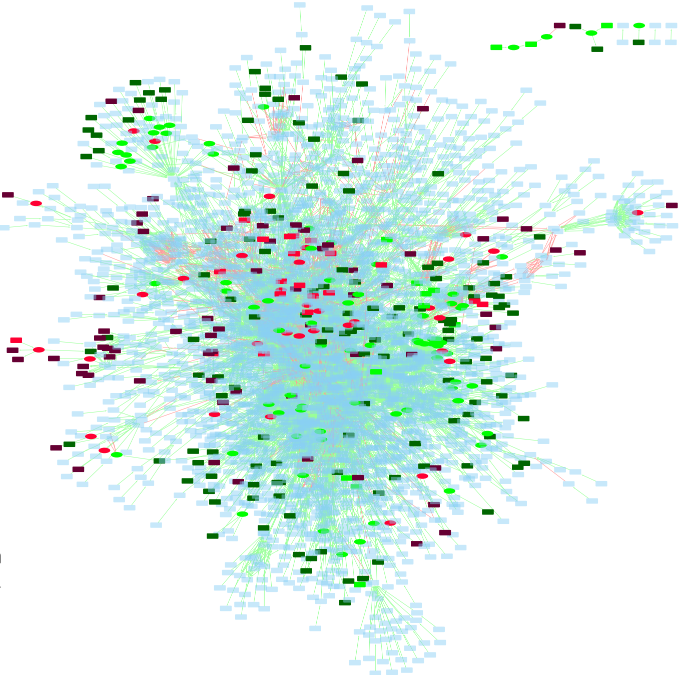
Knowledge from experiments:

- 138 up-regulated
- 71 down-regulated

Computational predictions:

- 92 predicted +
  - 24 non-trivial
- 54 predicted −
  - 33 non-trivial

70% more information compared to only knowledge from experiments



Prediction Results

New results compared to experimental data: complexes

Complexes predicted:

- NFKB1::BCL3 (+)
- NFKB2::RELB (+)
- JUND::NACA (−)

Results conflicting with experimental data

Predictions which are different from differential analysis:

- BAK1_gen, BMP4_gen, CREB1_prot, EIF4EBP2_prot, IGFBP3_gen, IGFBP3_prot, NR0B2_gen, NR0B2_prot, NR1H4_gen, NR1H4_prot, NR3C2_gen, NR3C2_prot, SESN3_gen, SESN3_prot, THBS1_gen, TNFRSF10A_gen, TP53_prot

Prediction Results

New results compared to experimental data: complexes

Complexes predicted:

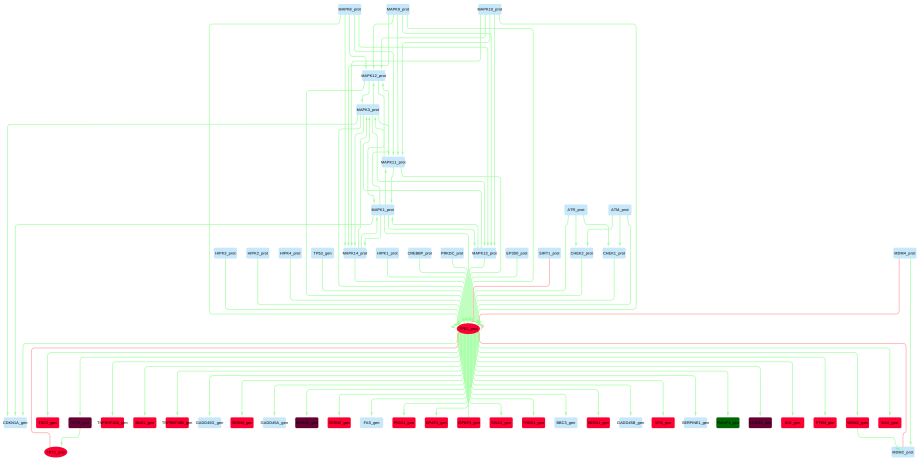
- NFKB1::BCL3 (+)
- NFKB2::RELB (+)
- JUND::NACA (−)

Results conflicting with experimental data

Predictions which are different from differential analysis:

- BAK1_gen, BMP4_gen, CREB1_prot, EIF4EBP2_prot, IGFBP3_gen, IGFBP3_prot, NR0B2_gen, NR0B2_prot, NR1H4_gen, NR1H4_prot, NR3C2_gen, NR3C2_prot, SESN3_gen, SESN3_prot, THBS1_gen, TNFRSF10A_gen, **TP53_prot**

Hub example: TP53_prot

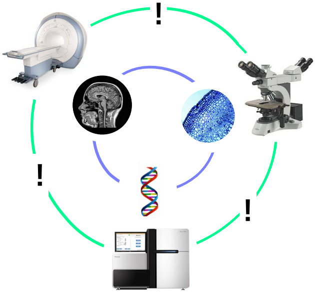


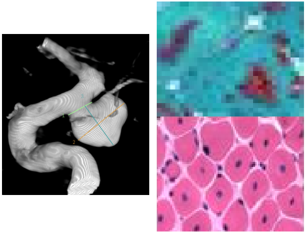
18 predictions directly depend of TP53_prot

Learning New Knowledge from Models

Create a Knowledge Graph of Clinical Data

INEX-MED Project



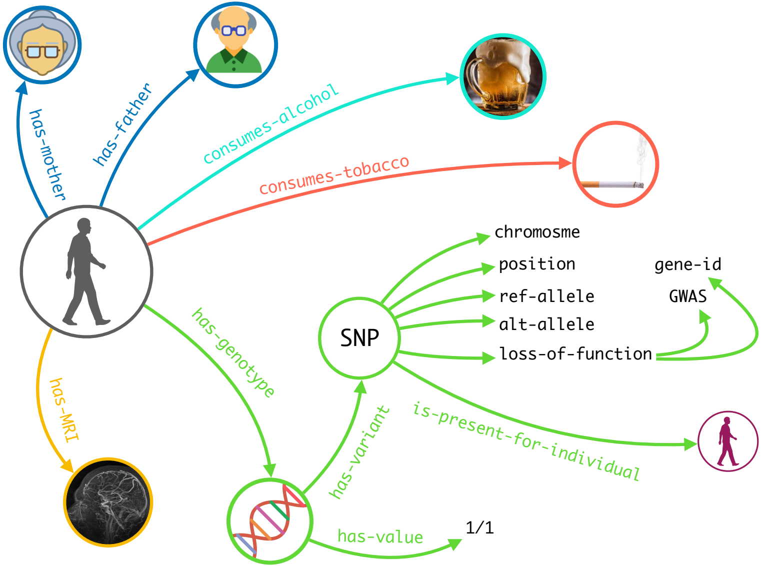


- Multiple data sources: clinical/diagnosis, imaging, microscopy, genomics
- Text/tabulated, not interoperable
- 2 use cases: **intracranial aneurysm** & **congenital myopathies**

Objectives:

- Create a general knowledge graph of **Linked Data**
- **SPARQL queries** on all sources
- **Machine learning** on the complete graph
- **FAIR** principles (Findability, Accessibility, Interoperability, Reusability)

Knowledge Graph



Project INEX-MED